

Programa del Diploma

# Guía de Informática

Primeros exámenes: 2010





Programa del Diploma

Material de ayuda al profesor de  
Informática  
Evaluación interna

Primeros exámenes: 2010

**Organización del Bachillerato Internacional**

Buenos Aires

Cardiff

Ginebra

Nueva York

Singapur

**Programa del Diploma  
Guía de Informática**

Versión en español del documento publicado en abril de 2004  
con el título *Computer science guide*

Publicada en abril de 2004  
Actualizada en septiembre de 2008

Bachillerato Internacional  
Peterson House, Malthouse Avenue, Cardiff Gate  
Cardiff, Wales GB CF23 8GL  
Reino Unido  
Tel.: +44 29 2054 7777  
Fax: +44 29 2054 7778  
Sitio web: <http://www.ibo.org>

© Organización del Bachillerato Internacional, 2004

**Glosario de términos informáticos**

Traducido y adaptado por el IB con autorización de Pearson Education Limited a partir del original en inglés

© British Informatics Society Ltd. 1997–98

© The British Computer Society 2002

El Bachillerato Internacional (IB) ofrece tres programas educativos exigentes y de calidad a una comunidad de colegios de todo el mundo, con el propósito de crear un mundo mejor y más pacífico.

El IB agradece la autorización para reproducir en esta publicación material protegido por derechos de autor. Cuando procede, se han citado las fuentes originales y, de serle notificado, el IB enmendará cualquier error u omisión con la mayor brevedad posible.

El uso del género masculino en esta publicación no tiene un propósito discriminatorio y se justifica únicamente como medio para hacer el texto más fluido. Se pretende que el español utilizado sea comprensible para todos los hablantes de esta lengua y no refleje una variante particular o regional de la misma.

Todos los derechos reservados. Esta publicación no puede reproducirse, almacenarse o distribuirse de forma total o parcial, en manera alguna ni por ningún medio, sin la previa autorización por escrito del IB, sin perjuicio de lo estipulado expresamente por la ley o por la política y normativa de uso de la propiedad intelectual del IB. Véase la página <http://www.ibo.org/es/copyright> del sitio web del IB para más información.

Los artículos promocionales y las publicaciones del IB pueden adquirirse en la tienda virtual del IB, disponible en <http://store.ibo.org>. Las consultas sobre pedidos deben dirigirse al departamento de marketing y ventas en Cardiff.

Tel.: +44 29 2054 7746  
Fax: +44 29 2054 7779  
Correo-e: [sales@ibo.org](mailto:sales@ibo.org)

# ÍNDICE

---

INTRODUCCIÓN	1
NATURALEZA DE LA ASIGNATURA	3
RECURSOS	4
MODELO CURRICULAR	5
OBJETIVOS GENERALES	6
OBJETIVOS ESPECÍFICOS	7
TÉRMINOS DE EXAMEN RELACIONADOS CON LOS OBJETIVOS ESPECÍFICOS	8
RESUMEN DEL PROGRAMA DE ESTUDIOS	10
DESCRIPCIÓN DETALLADA DEL PROGRAMA DE ESTUDIOS	12
ESTUDIO DE UN CASO	50
RESUMEN DE LA EVALUACIÓN	52
DESCRIPCIÓN DETALLADA DE LA EVALUACIÓN	54
DOMINIO	69
APÉNDICE 1	74
APÉNDICE 2	114
APÉNDICE 3	138
APÉNDICE 4	139

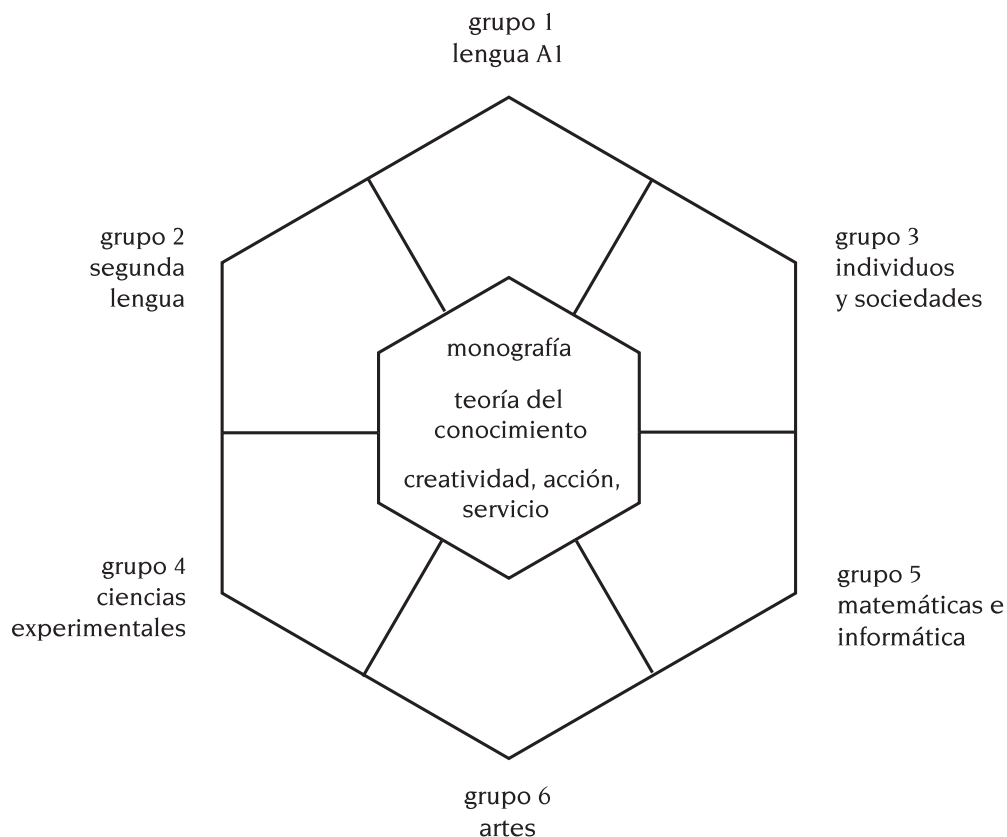


# INTRODUCCIÓN

---

El Programa del Diploma del Bachillerato Internacional es un curso pre-universitario exigente, diseñado para responder a las necesidades de estudiantes de secundaria altamente motivados, de edades comprendidas entre los 16 y los 19 años. El curso dura dos años y su amplio currículo prepara a los estudiantes para que cumplan con los requisitos de sistemas educativos de distintos países. Su modelo no se basa en el de ninguno en particular, sino que integra los mejores elementos de muchos de ellos. Puede cursarse en inglés, francés y español.

El modelo del programa se presenta en forma de hexágono, con seis áreas académicas en torno al centro. Las asignaturas se estudian simultáneamente y los estudiantes tienen la oportunidad de acceder a las dos grandes áreas tradicionales del saber, las humanidades y las ciencias.



Los alumnos aspirantes al Diploma deben seleccionar una asignatura de cada uno de los seis grupos de asignaturas. Por lo menos tres y no más de cuatro deben cursarse en el Nivel Superior (NS), y las demás en el Nivel Medio (NM). Se dedican 240 horas lectivas a los cursos de Nivel Superior y 150 a los de Nivel Medio. Al organizar los estudios de esta manera, se da a los estudiantes la posibilidad de explorar, en los dos años del programa, algunas disciplinas en profundidad y otras de modo más general. Este plan es el resultado de la búsqueda deliberada de un equilibrio entre la especialización precoz de ciertos sistemas nacionales y la universalidad preferida por otros.

El sistema de elección de asignaturas está concebido de tal manera que permite al estudiante con inclinaciones científicas aprender una lengua extranjera, y al lingüista nato familiarizarse con el trabajo de laboratorio. A la vez que se mantiene un equilibrio general, la flexibilidad de elegir asignaturas en el Nivel Superior permite al estudiante desarrollar áreas en las que está particularmente interesado y reunir los requisitos para el ingreso a la universidad.

Además del estudio de las seis asignaturas, los alumnos aspirantes al Diploma han de cumplir con otros tres requisitos. La Teoría del Conocimiento (TdC) es un curso interdisciplinario concebido para desarrollar un enfoque coherente del aprendizaje, que no sólo trascienda y unifique las diferentes áreas académicas sino que además estimule la apreciación de otras perspectivas culturales. La Monografía, de unas 4.000 palabras, ofrece a los estudiantes la oportunidad de investigar un tema de especial interés y les familiariza con la investigación independiente y el tipo de redacción académica que se espera de ellos en la universidad. La participación en el componente Creatividad, Acción y Servicio (CAS) del colegio anima a los estudiantes a tomar parte en actividades deportivas, artísticas y de servicio a la comunidad en el contexto local, nacional e internacional.

*Primeros exámenes: 2010*

# NATURALEZA DE LA ASIGNATURA

---

## Resolución de problemas

La informática conlleva la resolución de problemas mediante computadores. Por tanto, se requiere una comprensión total de la solución lógica de problemas, así como un conocimiento detallado del funcionamiento de los computadores. El éxito de un sistema informático depende de: un entendimiento total del problema que se debe solucionar; un uso adecuado del hardware, en función del conocimiento detallado de sus capacidades y limitaciones; el uso eficiente de los algoritmos y las estructuras de datos; un diseño lógico y minucioso; y la integración y pruebas cuidadosas de todos estos componentes. Los alumnos de Informática del Programa del Diploma del BI seguirán estrategias para la resolución de problemas que constantemente se reforzarán durante el trabajo en clase. En las fases iniciales del proceso se requerirá la identificación y definición de los problemas que se deben resolver mediante un sistema informático. El problema se descompone en partes que, a su vez, requieren una solución particular. A partir de esta definición de problema, el alumno construirá los algoritmos adecuados para crear una solución. Por tanto, cuando se utilicen computadores para solucionar problemas es necesario resaltar el uso de un enfoque lógico y un pensamiento analítico.

## Java

Se espera que los estudiantes adquieran el dominio de los aspectos de Java especificados. Entre los mecanismos adecuados se incluye la encapsulación, el polimorfismo y la herencia, aunque también son posibles otras aproximaciones estructuradas. El dominio de un aspecto o mecanismo concreto de la informática se define como la habilidad para utilizar dicho aspecto de forma adecuada para algún objetivo no trivial bien documentado. Este dominio se pondrá de manifiesto a través del trabajo enviado en el dossier de trabajo personal.

## Asignaturas

La asignatura de Informática para el Nivel Medio (NM) se centra en el desarrollo de software, en los fundamentos de los sistemas informáticos, y en la relación entre dichos sistemas y la sociedad. La asignatura para el Nivel Superior (NS) abarca todos estos elementos y, además, incluye matemáticas y lógica para informática, estructuras de datos y algoritmos avanzadas, otras cuestiones básicas de sistemas, y organización de archivos.

# RECURSOS

---

## Necesarios

Se consideran elementos **esenciales** para la enseñanza de la Informática:

- un computador personal (estación de trabajo) por alumno, durante el trabajo de programación, tanto en horario normal de clase como fuera de clase
- un compilador y un editor de Java, así como herramientas de depuración
- una impresora
- Internet.

## Recomendados

Los elementos **recomendados**, aunque no necesarios, son:

- una red
- equipamiento o dispositivos adicionales (por ejemplo, escáner o CD-ROM).

## No requeridos

**No se considera necesario** disponer de:

- dispositivos robóticos o de control
- herramientas CASE.

# MODELO CURRICULAR

---

Tanto los estudiantes del Nivel Medio (NM) como los del Nivel Superior (NS) deben estudiar un **tronco común** de material y demostrar el dominio de las técnicas de resolución de problemas y de varios aspectos de la informática mediante la realización de un **dossier de trabajo personal**. Además, los estudiantes de NS deben estudiar **unidades adicionales para NS**, que tienen dos funciones: ampliar algunas unidades del tronco común -al tratarlas con mayor profundidad- e introducir algunas unidades nuevas para proporcionar más conocimientos.

La existencia de un tronco común permitirá a los profesores enseñar ambos niveles conjuntamente, en algunas ocasiones y en el caso de que sea necesario. Este modelo curricular **no** implica que los estudiantes de NM y NS reciban clases conjuntamente. IBO **no** apoya la enseñanza conjunta de estudiantes de niveles diferentes porque no proporciona el mayor beneficio educacional para éstos; pero reconoce que esta estrategia puede resultar necesaria en algunos colegios.

<b>Unidades del tronco común</b> (todos los alumnos)
---

<b>Unidades adicionales para NS</b> (sólo alumnos del NS)
---

<b>Dossier de trabajo personal</b> (todos los alumnos)
--

## Horas lectivas

Las horas lectivas que deben asignarse a este modelo están conformes a los requisitos del Programa del Diploma: 150 horas para las asignaturas de NM y 240 horas para las de NS. Las horas se distribuyen de la siguiente manera:

Parte del modelo	Destinatarios	Horas de clase
Tronco común	todos los alumnos	125 horas
Unidades adicionales para NS	sólo los alumnos de NS	80 horas
Dossier de trabajo personal	alumnos de NM	25 horas
	alumnos de NS	35 horas

Las horas indicadas **no** incluyen el tiempo fuera del horario de clase que el alumno necesitará frente al computador (con el editor y el compilador adecuados) para poder desarrollar programas relacionados con el programa de estudios y el dossier de trabajo personal.

# OBJETIVOS GENERALES

---

Todas las asignaturas del Grupo 5 tienen como meta permitir a los alumnos:

- apreciar las perspectivas multiculturales e históricas de todas las asignaturas de este grupo
- disfrutar y llegar a apreciar la elegancia, las posibilidades y la utilidad de las asignaturas
- desarrollar el pensamiento lógico, crítico y creativo
- desarrollar una comprensión de los principios y la naturaleza de la asignatura
- emplear y perfeccionar sus capacidades de abstracción y generalización
- ejercitar la paciencia y la perseverancia en la resolución de problemas
- valorar las consecuencias derivadas de los avances tecnológicos
- aplicar destrezas a distintas situaciones y a la evolución de éstas
- comunicarse con claridad y confianza en diversos contextos.

# OBJETIVOS ESPECÍFICOS

---

Al finalizar las asignaturas de Informática NM o NS, se espera que los estudiantes hayan alcanzado los objetivos siguientes.

1.  **Demostrar comprensión de:** terminología, conceptos, procesos, estructuras, técnicas, principios, sistemas y consecuencias (importancia e implicaciones sociales) de la informática.
2.  **Aplicar y utilizar:** terminología, conceptos, procesos, estructuras, técnicas, principios y sistemas informáticos.
3.  **Analizar, discutir y evaluar:** terminología, conceptos, procesos, estructuras, técnicas, principios, sistemas y consecuencias (importancia e implicaciones sociales) de la informática.
4.  **Construir:** procesos, estructuras, técnicas y sistemas informáticos.

# TÉRMINOS DE EXAMEN RELACIONADOS CON LOS OBJETIVOS ESPECÍFICOS

---

Los términos que aquí se incluyen se aplican a los enunciados de evaluación y a las preguntas de los exámenes de Informática. Es aconsejable que los profesores se aseguren de que los estudiantes estén familiarizados con las definiciones. Asimismo, se puede orientar a los alumnos acerca del significado de un término en una pregunta concreta.

## Objetivo 1

- Defina* Dé el significado exacto de una palabra o frase de la forma más concisa posible.
- Dibuje* Represente mediante líneas trazadas con lápiz. Añada rótulos a menos que se diga lo contrario. (A veces, objetivo 2).
- Indique* Proporcione un nombre específico u otra respuesta breve. No es necesario ningún argumento o cálculo adicional.

## Objetivo 2

- Aplique* Utilice una idea, ecuación, principio, teoría o ley en una nueva situación. (A veces, objetivo 3).
- Calcule* Encuentre una respuesta exacta por medio de matemáticas u otros medios formales. Muestre las operaciones, a menos que se indique lo contrario. Se podrá utilizar “convierta”, “exprese” o “simplifique” para hacer referencia a formas específicas de cálculo. (A veces, objetivo 3).
- Describe* Proporcione una explicación detallada, incluyendo toda la información pertinente.
- Esboce* Dé una explicación breve o un resumen, incluyendo únicamente la información esencial.
- Estime* Encuentre una respuesta aproximada, normalmente por medio de métodos matemáticos.
- Identifique* Encuentre una respuesta entre varias posibilidades. (A veces, objetivo 3).
- Rastree* Haga un seguimiento y registre la acción de un algoritmo. (A veces, objetivo 3).

## Objetivo 3

- Analice* Interprete una información para llegar a unas conclusiones.
- Compare* Explique las semejanzas y diferencias entre dos (o más) elementos, haciendo referencia a cada uno de ellos. Las comparaciones se pueden presentar en una tabla.
- Discuta* Dé una explicación en la que se incluya, cuando sea posible, una gama de argumentos y valoraciones sobre la importancia relativa de varios factores o la comparación de hipótesis o ideas alternativas.
- Evalúe* Valore las implicaciones y limitaciones. (A veces, objetivo 2)
- Explique* Exponga con claridad, incluyendo las causas, razones o mecanismos.

## Objetivo 4

- Construya* Formule y/o reúna información de manera lógica.
- Determine* Encuentre la única respuesta posible. (A veces, objetivo 2)
- Diseñe* Produzca un plan, un objeto, una simulación o un modelo.
- Sugiera* Proponga una solución, una hipótesis u otra posible respuesta.

# RESUMEN DEL PROGRAMA DE ESTUDIOS

---

## Informática

Tronco común (alumnos de NS y NM) 125 h

Unidad 1: Ciclo de vida de los sistemas y desarrollo de software 35 h

1.1 Ciclo de vida de los sistemas 8 h

1.2 Análisis de sistemas 4 h

1.3 Diseño de sistemas 4 h

1.4 Importancia e implicaciones sociales de los sistemas informáticos 5 h

1.5 Ciclo de vida del software 2 h

1.6 Diseño de software 8 h

1.7 Documentación 4 h

Unidad 2: Construcción de programas en Java 50 h

Unidad 3: Fundamentos de los sistemas informáticos 37 h

3.1 Traductores de lenguajes 2 h

3.2 Arquitectura de computadores 12 h

3.3 Sistemas informáticos 5 h

3.4 Sistemas informáticos en red 8 h

3.5 Representación de datos 6 h

3.6 Errores 2 h

3.7 Software de utilidad 2 h

Estudio de un caso 3 h

**Dossier de trabajo personal**

Nivel Medio (NM)	25 h
------------------	------

Nivel Superior (NS)	35 h
---------------------	------

<b>Unidades adicionales para NS (sólo alumnos de NS)</b>	<b>80 h</b>
--	-------------

<b>Unidad 4: Matemáticas y lógica en informática</b>	<b>11 h</b>
--	-------------

4.1	Sistemas y representación de números	6 h
-----	--------------------------------------	-----

4.2	Lógica booleana	5 h
-----	-----------------	-----

<b>Unidad 5: Estructuras de datos abstractas y algoritmos</b>	<b>41 h</b>
---	-------------

5.1	Fundamentos	3 h
-----	-------------	-----

5.2	Estructuras de datos estáticas	8 h
-----	--------------------------------	-----

5.3	Estructuras de datos dinámicas	14 h
-----	--------------------------------	------

5.4	Los objetos en la resolución de problemas	6 h
-----	---	-----

5.5	Recursividad	6 h
-----	--------------	-----

5.6	Evaluación de algoritmos	4 h
-----	--------------------------	-----

<b>Unidad 6: Otras cuestiones básicas de sistemas</b>	<b>15 h</b>
---	-------------

6.1	Configuración del procesador	2 h
-----	------------------------------	-----

6.2	Almacenamiento en discos magnéticos	1 h
-----	-------------------------------------	-----

6.3	Sistemas operativos y utilidades	2 h
-----	----------------------------------	-----

6.4	Otras cuestiones básicas de redes	4 h
-----	-----------------------------------	-----

6.5	Comunicación computador/periféricos	6 h
-----	-------------------------------------	-----

<b>Unidad 7: Organización de archivos</b>	<b>10 h</b>
---	-------------

<b>Estudio de un caso</b>	<b>3 h</b>
---------------------------	------------

# DESCRIPCIÓN DETALLADA DEL PROGRAMA DE ESTUDIOS

---

## Formato del programa de estudios

Cada parte del programa de estudios proporciona la información siguiente:

- **Unidades:** Numeradas 1-3 (las del tronco común) y del 4-7 (las adicionales para NS).
- **Temas:** Numerados 1.1, 1.2 y así sucesivamente. Cada uno tiene unas horas lectivas estimadas.
- **Enunciados de evaluación:** Numerados 1.1.1, 1.1.2 y así sucesivamente.
- **Notas para los profesores:** Aparecen en una columna aparte.
- **Objetivos específicos de evaluación (Obj.):** Se indican mediante 1, 2, 3 o 4. (Véase *Objetivos específicos*)

## Enunciados de evaluación

Los enunciados de evaluación forman un programa de examen, **no** un programa de enseñanza, y tienen como fin establecer lo que los examinadores pueden evaluar mediante exámenes escritos. Cada enunciado se clasifica en función de los objetivos específicos de evaluación **1, 2, 3 o 4** para Informática. Dichos objetivos son importantes para lograr un equilibrio dentro del programa de estudios y los exámenes.

Los **términos de examen** son importantes porque ofrecen orientación a los alumnos y a los profesores sobre la profundidad y la amplitud de estudio necesarias. Es importante que los estudiantes conozcan el significado de dichos términos para entender exactamente el contenido de las preguntas de examen y lo que se espera de sus respuestas. (Véase *Términos de examen relacionados con los objetivos específicos*).

## Notas para los profesores

Las notas para los profesores acompañan a algunos enunciados de evaluación. Estas notas:

- pretenden aclarar la intención de los enunciados de evaluación
- ofrecen limitaciones para la profundidad y amplitud del tema
- pueden estipular lo que se desea y lo que no es necesario
- están diseñadas para asegurar que no haya sobrecarga de información.

## Programa de estudios

Se requiere que los profesores impartan: las **unidades del tronco común** (1-3) y sus temas a los alumnos del NM, y las **unidades del tronco común**, las **unidades adicionales para NS** (4-7) y sus temas a los alumnos del NS. Las habilidades asociadas con el desarrollo del dossier de trabajo personal se deben enseñar tanto a los alumnos de NM como a los de NS.

No es necesario enseñar las unidades en el orden en que aparecen en las secciones *Resumen del programa de estudios* y *Descripción detallada del programa de estudios*. Tampoco es necesario impartir las unidades del tronco común a los estudiantes del NS antes de impartir las unidades adicionales. Por tanto se recomienda a los profesores que planifiquen la enseñanza del temario y lo adapten a las necesidades de los estudiantes, de modo que se integren las unidades y el trabajo asociado con el dossier de trabajo personal.

## Distribución del tiempo

Las horas lectivas recomendadas para una asignatura de NM del Programa del Diploma son **150**; para NS, el número correspondiente de horas es de **240**. La distribución del tiempo propuesta en las secciones *Resumen del programa de estudios* y *Descripción detallada del programa de estudios* es aproximada, y sólo sugiere cómo se podría dividir el tiempo entre las diferentes unidades y el dossier de trabajo personal. Sin embargo, el tiempo exacto dedicado a cada unidad dependerá de varios factores, incluidos el conocimiento previo y el nivel de preparación de los estudiantes.

Se espera que en el Nivel Medio de Informática se dediquen **25** horas para trabajar en el dossier de trabajo personal; este número de horas aumenta hasta **35** en el Nivel Superior. Las horas indicadas no incluyen el tiempo fuera del horario de clase que el alumno necesitará frente al computador (con compilador/intérprete adecuados) para poder desarrollar programas relacionados con el programa de estudios y el dossier de trabajo personal. (Véase *Modelo curricular*).

## Uso de calculadoras

Se permite el uso de calculadoras en las clases, pero no en los exámenes.

## Material de ayuda al profesor

Se está produciendo una gran variedad de materiales de ayuda al profesor para complementar esta guía. En éstos se incluirá orientación para la corrección de dossiers de trabajo personal, y ejemplos de pruebas de examen y esquemas de calificación.

# Tronco común

## Unidad I: Ciclo de vida de los sistemas y desarrollo de software

Los alumnos deben comprender las tareas que realiza un analista de sistemas al considerar una situación que se pueda informatizar. En esta unidad se tratan éstas y otras tareas posteriores incluidas en el ciclo de vida de un sistema. Se espera que en el dossier de trabajo personal se refleje la comprensión y el dominio de estos aspectos.

Los alumnos deben aprender a analizar y resolver problemas, no sólo a escribir programas. El ciclo de vida del software consta de varias etapas, y se espera que los estudiantes participen, hasta un cierto nivel, en todas ellas. Un buen análisis de sistemas debe incluir investigación, obtención de datos, una planificación cuidadosa y una documentación minuciosa. Si el problema se analiza de forma adecuada, la implementación será más fácil y exitosa.

### Tema I.1: Ciclo de vida de los sistemas

8 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.1.1	Esboce el ciclo de vida de los sistemas en términos de las fases: análisis, diseño, implementación, funcionamiento y mantenimiento.	Existen otros modelos aceptables, siempre y cuando pongan énfasis en la naturaleza cíclica del proceso de resolución del problema.	2
1.1.2	Explique la importancia de la obtención de datos durante la fase de análisis.		3
1.1.3	Compare métodos de obtención de datos.	Ejemplos: entrevistas a usuarios y expertos en la materia, elaboración de cuestionarios, observación de los sistemas actuales y estudio de la documentación del usuario.	3
1.1.4	Describa la elaboración de una especificación de requisitos durante la fase de análisis.	Se puede incluir: definición de las entradas y salidas, una lista de herramientas, instalaciones, personal disponible para desarrollar la solución y una planificación para las fases siguientes del proyecto.	2
1.1.5	Esboce las características de un informe de viabilidad.	El informe de viabilidad puede elaborarse en la fase de análisis, en la de diseño, o en ambas. Se puede incluir: una breve descripción del sistema propuesto, los costos estimados, la responsabilidad económica, técnica y legal, y una posible fecha de finalización.	2

## Unidad I: Ciclo de vida de los sistemas y desarrollo de software (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.1.6	Compare las ventajas y desventajas de las soluciones alternativas en la fase de diseño. Entre éstas, se incluyen las soluciones hardware y software.	Es necesario tener en cuenta y evaluar varias soluciones posibles, haciendo preguntas como: ¿qué tipo de salida debe obtenerse?, ¿de dónde procederán los datos y cómo se introducirán?, ¿debería el sistema estar centralizado o en red?, ¿se necesita utilizar computadores?, ¿se debería utilizar software estándar?, ¿qué nivel de personalización se desea? Es necesario poner énfasis en la organización modular. También se debe tener en cuenta la interfaz entre el usuario y el computador.	3
1.1.7	Discuta los métodos para probar sistemas, la importancia de unas pruebas adecuadas y las implicaciones de un método de pruebas inadecuado.	Los alumnos deben ser capaces de proponer datos de prueba adecuados, incluyendo razones, durante las fases de diseño e implementación.	3
1.1.8	Esboce métodos de implementar nuevos sistemas.	Los métodos incluyen: la ejecución paralela de sistemas antiguos y sistemas nuevos, el cambio directo y la introducción de forma progresiva. Se deben tratar las implicaciones de capacitación y los posibles problemas durante la instalación.	2
1.1.9	Esboce las funciones y la importancia del mantenimiento de sistemas.	Las funciones que se deben tratar son las revisiones periódicas, la evaluación del rendimiento y la claridad en la documentación, para facilitar futuras modificaciones.	2

# Unidad 1: Ciclo de vida de los sistemas y desarrollo de software (continuación)

## Tema 1.2: Análisis de sistemas

4 h

Los alumnos deben aprender a investigar y analizar problemas en el nivel del sistema antes de empezar a pensar en una solución (algoritmos). También deben ser capaces de leer y construir diagramas de flujo de sistemas.

	<b>Enunciados de evaluación</b>	<b>Notas para la enseñanza</b>	<b>Obj.</b>
<b>1.2.1</b>	Explique la importancia de formular un problema de forma precisa.		<b>3</b>
<b>1.2.2</b>	Discuta los aspectos que se deben tener en cuenta en un problema concreto.	Los alumnos deben ser conscientes de la necesidad de actividades como entrevistas, cuestionarios y búsquedas bibliográficas para descubrir los aspectos pertinentes.	<b>3</b>
<b>1.2.3</b>	Identifique los resultados que debe producir una solución adecuada para resolver un problema concreto.		<b>2</b>
<b>1.2.4</b>	Identifique las partes de un problema que se pueden resolver adecuadamente mediante computadores.		<b>2</b>
<b>1.2.5</b>	Identifique las tres estructuras de control básicas de la programación: aceptación de datos, procesamiento y producción de una salida con los resultados.		<b>2</b>
<b>1.2.6</b>	Analice un problema mediante su descomposición en módulos.	Por ejemplo, los módulos pueden representar la entrada, el procesamiento y la salida de la solución al problema.	<b>3</b>

## Unidad I: Ciclo de vida de los sistemas y desarrollo de software (continuación)

### Tema 1.3: Diseño de sistemas

4 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.3.1	Indique las partes de un sistema.		1
1.3.2	Identifique las partes que debe almacenar y procesar un sistema.		2
1.3.3	Esboce métodos adecuados de captura de datos y presentación de salidas para un sistema.		2
1.3.4	Diseñe estructuras de datos apropiadas para almacenar datos en un sistema.		4
1.3.5	Indique los componentes de hardware apropiados para un sistema.		1
1.3.6	Esboce una interfaz adecuada entre un sistema y los usuarios.		2
1.3.7	Analice un diagrama de flujo de sistemas que represente un sistema entero.		3
1.3.8	Construya un diagrama de flujo de sistemas para representar un sistema entero.	Los símbolos que deben utilizar los alumnos aparecen en el apéndice 3.	4

## Unidad I: Ciclo de vida de los sistemas y desarrollo de software (continuación)

### Tema 1.4: Importancia e implicaciones sociales de los sistemas informáticos

5 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.4.1	Discuta las implicaciones sociales y económicas de la instalación de nuevos sistemas.	Véase el punto 1.1.8 para obtener información sobre los métodos de instalación que se deben tratar.	3
1.4.2	Discuta la importancia y las implicaciones sociales del uso extendido de los computadores en la sociedad.	La importancia social debe tratarse en relación con las consecuencias económicas, políticas, culturales y ambientales. Entre éstas, se incluyen: efectos sobre el empleo (cambios en el entorno laboral, nueva formación, etc.); computadores (hacking, virus, etc.); requisitos éticos y legales; almacenamiento de datos (protección de la privacidad y de datos, etc.); usuarios de software (copyright, licencias de software, etc.).	3
1.4.3	Discuta las tendencias actuales de los sistemas informáticos y las consecuencias de las mismas.		3

### Tema 1.5: Ciclo de vida del software

2 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.5.1	Esboce las principales fases del ciclo de vida del software.	En un modelo se incluye: análisis del sistema, conducente a un enunciado preciso del problema que se ha de resolver (especificación de requisitos); diseño de software; construcción de programas, incluidas las pruebas y la depuración; instalación y funcionamiento; mantenimiento. Existen otros modelos aceptables, siempre y cuando pongan énfasis en la naturaleza cíclica de la vida del software.	2

## Unidad I: Ciclo de vida de los sistemas y desarrollo de software (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.5.2	Explique por qué la producción de software es generalmente cíclica.	Los alumnos deben comprender que los sistemas informáticos se utilizan durante prolongados períodos de tiempo. El software de estos sistemas requiere mejoras periódicas. Después del diseño original y la implementación, se requieren nuevos análisis, rediseños y reestructuraciones para satisfacer las necesidades cambiantes. Esta dinámica continuará durante varios ciclos de análisis, diseño, implementación y uso.	3

### Tema 1.6: Diseño de software

8 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.6.1	Esboce los datos necesarios para resolver un problema con el que los alumnos no se hayan encontrado anteriormente, incluidos el formato de los archivos de datos y los requisitos de entrada y salida con el uso de interfaces de usuario adecuadas.	Por ejemplo, pantallas, formularios para OMR y formatos de informes.	2
1.6.2	Discuta las ventajas de la modularidad en el diseño de la solución para un problema.		3
1.6.3	Defina el término “creación de prototipos”.		1
1.6.4	Esboce la aproximación al diseño y desarrollo de sistemas mediante la creación de prototipos.	La creación de prototipos se puede realizar en diferentes niveles de complejidad. Para los objetivos de esta asignatura, la creación de prototipos se limita a la presentación de una solución preliminar que podría no ser funcional.	2
1.6.5	Discuta las ventajas que tiene para los usuarios finales y los diseñadores de sistemas la aproximación mediante creación de prototipos.	La creación de prototipos se puede emplear con los usuarios finales para obtener comentarios en la fase inicial del proceso de diseño. Los diseñadores de sistemas pueden emplear la creación de prototipos para buscar soluciones alternativas a un problema.	3

## Unidad I: Ciclo de vida de los sistemas y desarrollo de software (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.6.6	Esboce la eficiencia de una solución en términos de requisitos de almacenamiento, de requisitos de memoria, y de velocidad.	Se espera solamente un tratamiento cualitativo o un cálculo específico; la notación “O” u O Mayúscula sólo se requiere en el NS. (Véase 5.6 <i>Evaluación de algoritmos</i> ).	2
1.6.7	Esboce cómo se pueden probar y depurar programas.	Las pruebas requieren rastrear secciones de un algoritmo, incluyendo las respuestas a los errores (“ensayos”), así como el diseño de casos de prueba que se ejecutan posteriormente. Los alumnos deben ser capaces de proponer datos de prueba adecuados, así como de ofrecer razones. La depuración tiene los componentes de detección, diagnóstico y corrección de errores que aparezcan en las pruebas.	2
1.6.8	Describa la función de las herramientas en la construcción, prueba y depuración de programas.	Sería aconsejable que los alumnos utilizaran un entorno de desarrollo integrado (IDE), en el que se combine un editor, intérprete o compilador y herramientas de depuración; sin embargo, no se considera un requisito.	2

### Tema 1.7: Documentación

4 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
1.7.1	Esboce por qué es necesaria la documentación de cada fase del ciclo.		2
1.7.2	Explique las características de la documentación en el diseño, la programación y el mantenimiento, es decir, la documentación del sistema.	Es necesario que los alumnos documenten el proceso de resolución de problemas en función de los estándares descritos en las directrices para el dossier de trabajo personal. Los listados de los programas también deben estar completamente documentados.	3

## Unidad 1: Ciclo de vida de los sistemas y desarrollo de software (continuación)

	<b>Enunciados de evaluación</b>	<b>Notas para la enseñanza</b>	<b>Obj.</b>
<b>1.7.3</b>	Explique las características de la documentación destinada al usuario, es decir, la documentación del usuario.	Es necesario que los alumnos escriban instrucciones para el usuario final en función de los estándares descritos en las directrices para el dossier de trabajo personal. Los alumnos deben saber que es posible que se necesiten otros manuales de usuario (por ejemplo, sistemas de ayuda en línea y manuales de instalación en los que el usuario final no instala los sistemas), aunque no se exigirá que escriban este tipo de documentación.	<b>3</b>

## Unidad 2: Construcción de programas en Java

### Tema 2.1: Construcción de programas en Java

50 h

En este tema, la discusión del material tendrá una función fundamental en el desarrollo de los dossiers de trabajo personal. Aunque se han asignado 50 horas, debe tenerse en cuenta que algunas de las 25 horas asignadas como horas de contacto con el profesor se utilizarán en la discusión de estos aspectos. Para el lenguaje de alto nivel debe utilizarse la sintaxis de Java tal como se especifica en el apéndice 2.

	Enunciados de evaluación	Notas para la enseñanza	Obj.
2.1.1	<p>Aplique las siguientes estructuras de un lenguaje de alto nivel de forma correcta para implementar un diseño de software expresado en Java.</p> <ul style="list-style-type: none"><li>• Declare variables y tipos con el ámbito adecuado, distinguiendo entre identificadores privados y públicos.</li><li>• Defina y aplique objetos definidos por el usuario.</li><li>• Formatee la salida de forma que resulte fácil para el usuario.</li><li>• Construya y calcule expresiones aritméticas, relacionales y booleanas (únicamente and, or, not) mediante los operadores adecuados (&amp;&amp;,   , !) y teniendo en cuenta la precedencia.</li><li>• Construya y calcule el valor de las expresiones aritméticas de módulo mod y div utilizando los operadores adecuados (% , /) y teniendo en cuenta la precedencia.</li><li>• Implemente las restantes estructuras algorítmicas en Java: matrices, objetos, estructuras de selección (ramificación), operaciones con archivos, estructuras iterativas (bucles), centinelas e indicadores.</li><li>• Utilice subprogramas incorporados, incluyendo los de las clases de Java especificados en el apéndice 2.</li><li>• Defina y aplique métodos definidos por el usuario.</li><li>• Demuestre la comprensión de la firma de métodos.</li><li>• Demuestre la comprensión del uso de parámetros, incluyendo el paso de parámetros de primitivas y objetos y la devolución de valores.</li><li>• Demuestre la comprensión del ámbito de las identidades en Java, que se restringe a las palabras clave <code>private</code> y <code>public</code>.</li><li>• Defina “primitiva”, “clase”, “objeto”, “miembro dato”, “método”, “firma de un método” y “constructor”.</li></ul>		3

## Unidad 2: Construcción de programas en Java (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
2.1.2	Aplice los tipos y las estructuras de datos apropiados para resolver un problema desconocido hasta el momento.	Los tipos de datos requeridos son enteros, reales, caracteres y booleanos. Las estructuras de datos requeridas son cadenas de caracteres, matrices de una y dos dimensiones, registros y archivos.	3
2.1.3	Describa la naturaleza y función de los tipos y estructuras de datos presentados en 2.1.2.		2
2.1.4	Rastree algoritmos en Java.	Véase el apéndice 2. En las preguntas de los exámenes siempre se utilizará Java cuando sea necesario escribir código; por tanto, los alumnos deben ser capaces de entender algoritmos presentados en este lenguaje. Los algoritmos podrán ser los estándares del programa de estudios u otros de complejidad equivalente que los alumnos no hayan visto anteriormente. En los algoritmos se puede utilizar cualquiera de los tipos de datos y estructuras que se indican en 2.1.2.	2
2.1.5	Explique algoritmos escritos en Java con respecto a la eficiencia, corrección y adecuación para una tarea.	Véase nota en 2.1.4.	3
2.1.6	Construya algoritmos en Java.	Véase nota en 2.1.4.	4
2.1.7	Explique la necesidad de los métodos de búsqueda y ordenación.		3
2.1.8	Aplice algoritmos de búsqueda secuencial (lineal) y binaria, algoritmos de ordenación por selección y por el método de la burbuja para la resolución de problemas, incluyendo algunos que no se hayan estudiado anteriormente.	La búsqueda y la ordenación son buenos ejemplos para el estudio del diseño, desarrollo y análisis de algoritmos. Los alumnos deben ser capaces de discutir las circunstancias adecuadas para el uso de cada algoritmo. En los exámenes se podrán plantear descripciones de otros algoritmos para desarrollar.	3
2.1.9	Compare la eficiencia de los algoritmos específicos de búsqueda y ordenación mencionados en 2.1.8.		3

## Unidad 2: Construcción de programas en Java (continuación)

	<b>Enunciados de evaluación</b>	<b>Notas para la enseñanza</b>	<b>Obj.</b>
<b>2.1.10</b>	Discuta la eficiencia de algoritmos específicos de búsqueda y ordenación.	La notación O Mayúscula no se exige en el NM.	<b>3</b>
<b>2.1.11</b>	Describa errores de sintaxis, lógica y tiempo de ejecución.	Los errores de desbordamiento, subdesbordamiento y truncamiento pueden surgir durante el desarrollo de programas y, por tanto, podrán discutirse; sin embargo, no serán objeto de examen en el NM.	<b>2</b>

## Unidad 3: Fundamentos de los sistemas informáticos

En esta unidad se estudian los sistemas informáticos (hardware y software) y cómo interactúan.

### Tema 3.1: Traductores de lenguajes

2 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.1.1	Defina “sintaxis” y “semántica”.		1
3.1.2	Describa la función de los traductores de lenguajes de alto nivel.	Los traductores deben estar limitados a intérpretes y compiladores.	2
3.1.3	Esboce el uso de herramientas de desarrollo de software.	Ejemplos: sistemas de gestión de bases de datos, macros, herramientas CASE y traductores de lenguajes simples (los intérpretes y compiladores no son ejemplos adecuados en este contexto), editores HTML, editores de páginas web, editores de código, IDEs visuales.	2

### Tema 3.2: Arquitectura de computadores

12 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.2.1	Esboce la estructura de la unidad central de procesamiento (CPU), incluyendo las funciones de la unidad de control (CU), la unidad aritmético-lógica (ALU), la memoria principal y los buses de direcciones.	Se espera que los alumnos sean capaces de reproducir un diagrama básico en el que se ilustre la CPU, y sepan que cada ubicación de la memoria principal posee una única dirección.	2
3.2.2	Esboce el significado de los términos bit (b), byte (B) y sus derivados.	Los alumnos deben saber que en un computador todo se almacena y procesa en binario, de ahí la relación entre bits, bytes, etc. en potencias de 2. Por ejemplo, 1 kilobyte = $2^{10}$ bytes. Asimismo, deben familiarizarse con los prefijos “T”, “G”, “M”, “k” y su utilización en las medidas informáticas. Deben ser capaces de aplicar los prefijos “T”, “G”, “M” y “k” a los bits y bytes. Por ejemplo TB (terabytes), Gb (gigabits) y MB (megabytes).	2

## Unidad 3: Fundamentos de los sistemas informáticos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.2.3	Esboce el significado de los términos “palabra”, “registro” y “dirección”, así como su utilización en el almacenamiento de datos e instrucciones.	No se requiere el estudio de registros específicos.	2
3.2.4	Esboce los pasos que componen el ciclo de una instrucción de máquina: seleccionar, decodificar, ejecutar y almacenar.	Un modelo con un único procesador es suficiente. No se requiere el estudio de una CPU específica.	2
3.2.5	Esboce las características de la memoria principal y la diferencia entre la memoria volátil y no volátil.	Los alumnos deben comprender la función de las memorias RAM, ROM y caché, así como el tamaño habitual (en bytes). Es necesario comprender la forma en que se puede utilizar la memoria virtual para aumentar la memoria principal; sin embargo, no es necesario conocer detalles de paginación.	2
3.2.6	Esboce las características de la memoria secundaria y defina “acceso secuencial” y “acceso directo”.	Con respecto a la memoria secundaria, se debe hacer referencia a unidades de disco flash, CDs, DVDs y cintas. Los alumnos deben conocer el tipo de acceso de los medios de memoria secundaria anteriores. Además, deben saber proporcionar un ejemplo de aplicación de cada tipo y justificar su uso en dicha aplicación.	2
3.2.7	Esboce el papel que desempeña un microprocesador diseñado para ejecutar una o varias funciones (en un coche, una lavadora, etc.).	Los alumnos deben comprender la necesidad de la existencia de tipos diferentes de memoria en un microprocesador. Asimismo, deben ser capaces de citar al menos un ejemplo del uso de un microprocesador y de indicar las entradas y las salidas.	2
3.2.8	Discuta las características, ventajas, desventajas y aplicaciones de los dispositivos específicos de entrada y salida, así como de los medios que utiliza cada uno.	Los alumnos deben conocer las características de los elementos siguientes: <i>mouse</i> (ratón), teclado, pantalla táctil, reconocimiento óptico de caracteres (OCR), reconocimiento de caracteres de tinta magnética (MICR), escáneres (de página, de detección de marcas y código de barras), monitores LCD, reconocimiento de voz, sensores, cámaras digitales, tabletas gráficas, impresoras, trazadores de gráficos, monitores, robótica y sonido. No son necesarios detalles técnicos, a menos que se introduzcan en el estudio de un caso.	3

## Unidad 3: Fundamentos de los sistemas informáticos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.2.9	Esboce desarrollos recientes en el campo de la arquitectura de sistemas informáticos, incluyendo la arquitectura de procesadores, las tecnologías de memoria principal y los dispositivos de memoria secundaria.	No son necesarios detalles técnicos, a menos que se introduzcan en el Estudio de un caso.	2

### Tema 3.3: Sistemas informáticos

5 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.3.1	Defina el término “sistema operativo”.	No se requieren conocimientos sobre sistemas operativos específicos.	1
3.3.2	Esboce las funciones de los sistemas operativos.	En las funciones se incluyen: comunicación con los periféricos; coordinación del procesamiento concurrente de trabajos; gestión de memoria; monitorización, enumeración y seguridad de recursos; gestión de programas y datos; y proporcionar interfaces de usuario adecuadas.	2
3.3.3	Discuta las características de varios sistemas informáticos, incluyendo los sistemas monousuario y multiusuario, en entornos monotarea y multitarea.	Es necesario comprender los términos “multiacceso” y “multiprogramación”; sin embargo, no constituyen materia de examen los detalles sobre su administración.	3
3.3.4	Compare las características y aplicaciones de los diferentes tipos de computadores.	Se deben tener en cuenta los computadores personales, portátiles, centrales y los supercomputadores. Entre las características se deberían incluir: tamaño de las memorias principal y secundaria, dispositivos de entrada/salida (E/S), entorno (tamaño, comodidad, lugar de utilización), costo, usuarios (multi o mono), procesador (longitud de palabra, tamaño del bus, y frecuencia).	3
3.3.5	Esboce las características principales del procesamiento por lotes, en línea (interactivo) y en tiempo real.		2

## Unidad 3: Fundamentos de los sistemas informáticos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.3.6	Esboce algunas aplicaciones que utilicen cada uno de los métodos de procesamiento enumerados en 3.3.5: procesamiento por lotes (p.ej. procesamiento de nóminas y cheques bancarios); procesamiento interactivo (en línea) (p.ej. procesamiento de textos, juegos para el computador); procesamiento en tiempo real (p.ej. control del tráfico aéreo o monitorización de pacientes en la unidad de cuidados intensivos de un hospital).		2
3.3.7	Explique la relación entre los archivos maestros y los archivos de transacción.	Estos aspectos deberán relacionarse con los ejemplos de 3.3.6.	3
3.3.8	Discuta la fiabilidad del sistema, incluyendo las implicaciones de los fallos.	La necesidad y el uso de estrategias de copias de seguridad, los sistemas espejados y las utilidades se explican en 3.7.	3

### Tema 3.4: Sistemas informáticos en red

8 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.4.1	Defina “red de área local (LAN)”, “red de área ancha (WAN)”, “servidor” y “cliente”.		1
3.4.2	Explique las topologías de red básicas.	Los alumnos deben ser capaces de explicar e ilustrar las redes en estrella y en bus, así como los híbridos que incluyan a ambas redes.	3
3.4.3	Explique el hardware necesario para la interconexión de redes.	En el hardware se debe incluir enlaces de comunicación (cables, fibra óptica, microondas, etc.), hubs, conmutadores, nodos y encaminadores.	3

## Unidad 3: Fundamentos de los sistemas informáticos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.4.4	Defina los términos “protocolo estándar”, “integridad de datos” y “seguridad de datos” en el contexto de la transmisión de datos a través de una red.	Los alumnos deben saber que los protocolos estándares son un conjunto de reglas reconocidas internacionalmente para la transmisión de datos. También deben conocer la diferencia entre seguridad de datos e integridad de datos. Los alumnos no necesitan conocer detalles específicos o técnicos, como sistema de capas ISO (OSI), TCP/IP, etc.	1
3.4.5	Explique el software necesario para la interconexión de redes.	Los alumnos deben comprender la función del software de comunicaciones en la conexión de redes de área local y área ancha, así como la necesidad de trabajar con protocolos y seguridad de datos.	3
3.4.6	Describa los métodos adecuados para asegurar la integridad en la transmisión de datos.	Es necesario comprender los códigos de verificación de errores, tales como sumas de verificación (verificaciones de caracteres en bloque) y verificaciones de paridad. Es necesario comprender las razones para el uso de la retransmisión. Se debe tener en cuenta la calidad de las líneas de comunicación.	2
3.4.7	Describa métodos adecuados para garantizar la seguridad de los datos.	Los alumnos deben comprender el concepto de encriptación de datos, pero no es necesario ofrecer detalles algorítmicos. Deben comprender la necesidad y el uso de contraseñas, la seguridad física y los distintos niveles de acceso (permisos) para los diferentes usuarios.	2
3.4.8	Discuta la necesidad de la velocidad en la transmisión de datos y cómo se puede mejorar dicha velocidad.	Los alumnos deben saber que los archivos de documentos y gráficos se pueden enviar en diferentes formatos y que el formato seleccionado afecta a la velocidad de transmisión. Deben conocerse los formatos más comunes, como JPEG y BMP. Los principios de compresión de datos deben tenerse en cuenta, pero no es necesario conocer detalles sobre métodos de compresión.	3

## Unidad 3: Fundamentos de los sistemas informáticos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.4.9	Discuta las aplicaciones de red y las implicaciones del uso de redes para las organizaciones, incluyendo las comunicaciones internas, el correo y el comercio electrónicos, las conferencias y el procesamiento distribuido.	Es necesario tener en cuenta el uso de LANs, WANs públicas y privadas, e Internet.	3
3.1.10	Esboce las funciones de un navegador Web y un motor de búsqueda, incluyendo la visualización de una página HTML, el seguimiento de hipervínculos y la búsqueda mediante palabras clave.	No es necesario conocer nombres específicos de navegadores y motores de búsqueda.	2

### Tema 3.5: Representación de datos

6 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.5.1	Esboce el uso del sistema binario para la representación de datos.	Los alumnos deben comprender la relación entre el número de dígitos y el número de patrones disponibles ( $2^n$ , por ejemplo. La representación del color en 4 bits permite 16 colores; un bus de direcciones de 32 bits puede direccionar 4GB de RAM). Es necesario conocer las diferentes características de los códigos ASCII y Unicode, pero no se espera que los alumnos conozcan las representaciones específicas de caracteres.	2
3.5.2	Esboce la necesidad de formatos estándares para el almacenamiento de documentos y archivos.	Relacionar con 3.4.8 y 3.4.9.	2
3.5.3	Expresa números en las bases: decimal, binaria y hexadecimal.		2
3.5.4	Realice conversiones de enteros entre las bases especificadas en 3.5.3 (máximo 8 bits).		2

## Unidad 3: Fundamentos de los sistemas informáticos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.5.5	Aplice la notación binaria para la representación de enteros, tanto negativos como positivos, utilizando el método de complemento a dos.		2
3.5.6	Defina “datos analógicos” y “datos digitales”.		1
3.5.7	Esboce la necesidad de la interconversión de datos entre formatos analógicos y digitales para el procesamiento informático.	Los alumnos deben comprender la necesidad de la conversión de datos para el procesamiento; p. ej., los sensores y los módems.	2
3.5.8	Discuta dos aplicaciones que requieran la conversión de datos entre formatos analógicos y digitales, incluyendo sensores de temperaturas.	Los profesores tienen libertad para elegir la segunda aplicación. Otros ejemplos de software incluyen: reconocimiento de voz, detección de luz, procesamiento de imágenes y software de OCR.	3

### Tema 3.6: Errores

2 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.6.1	Describa las siguientes causas de errores, con referencia a una aplicación en cada caso: de entrada de datos, accidental, deliberada, de software y de hardware.		2
3.6.2	Esboce los métodos de detección y prevención de cada uno de los errores enumerados en 3.6.1.	Es necesario comprender la verificación y la validación. Deben explicarse los dígitos de verificación y el total de dispersiones. También es necesario comprender los operadores de módulo (mod, div) en la formación de dígitos de verificación.	2
3.6.3	Describa métodos de recuperación ante errores.	Se deben considerar las opciones de repetición de la entrada, retransmisión y recuperación a partir de copias de seguridad. No se requieren algoritmos de corrección de errores.	2

## Unidad 3: Fundamentos de los sistemas informáticos (continuación)

### Tema 3.7: Software de utilidad

2 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
3.7.1	Esboce la función o las funciones principales de las utilidades de software siguientes: compresores de datos, software antivirus, gestores de archivos y software de desfragmentación.	Las funciones necesarias de un gestor de archivos son: copiar, eliminar, formatear, buscar, crear carpetas/directorios, archivar, imprimir, realizar copias de seguridad, cambiar nombres y restaurar. El hecho de que los archivos no se almacenen contiguamente sólo debe estudiarse de forma somera para comprender por qué se requiere software de desfragmentación. No se requieren detalles técnicos.	2
3.7.2	Discuta la necesidad de cada una de las utilidades presentadas en 3.7.1.		3

# Unidades adicionales para el NS

## Unidad 4: Matemáticas y lógica en informática

Informática no es una asignatura de matemáticas. Sin embargo, los temas siguientes permiten al estudiante comprender los principios básicos de la arquitectura de computadores, entender las causas fundamentales de muchos errores comunes, diseñar circuitos simples y construir algunos algoritmos comunes que requieran técnicas matemáticas.

### Tema 4.1: Sistemas y representación de números

6 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
4.1.1	Realice cálculos en las bases especificadas en 3.5.3.	Para cálculos en las bases hexadecimal y binaria sólo es necesario conocer la suma.	3
4.1.2	Indique la mantisa y el exponente de un número binario con representación en punto flotante. Relacione lo anterior con la notación científica en decimal.	Para los números binarios negativos, tanto enteros como reales, sólo se necesita el método de complemento a dos.	1
4.1.3	Aplice la notación binaria para representar números reales.	Es necesario conocer la representación en punto fijo y flotante. Dada una representación específica, los alumnos deben ser capaces de calcular el rango de números en punto flotante normalizados. Se deben entender cuestiones como la necesidad de la normalización y la pérdida de precisión.	2
4.1.4	Discuta las ventajas y desventajas de las representaciones de enteros y en punto flotante.		3
4.1.5	Defina “error de truncamiento”, “error de subdesbordamiento” y “error de desbordamiento”.		1
4.1.6	Esboce tres situaciones, cada una de las cuales proporcione un ejemplo de cuándo y dónde se pueden producir alguno de los errores enumerados en 4.1.5. Cada situación debe mostrar un error diferente, es decir, deben describirse los tres errores.		2

## Unidad 4: Matemáticas y lógica en informática (continuación)

### Tema 4.2: Lógica booleana

5 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.										
4.2.1	Defina los operadores booleanos <b>and</b> , <b>or</b> , <b>not</b> , <b>nand</b> , <b>nor</b> y <b>xor</b> , mediante la representación de la tabla de verdad correspondiente.		1										
4.2.2	Construya expresiones booleanas mediante los operadores enumerados en 4.2.1.	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Operador</th> <th>Símbolo</th> </tr> </thead> <tbody> <tr> <td><b>and</b></td> <td>•</td> </tr> <tr> <td><b>or</b></td> <td>+</td> </tr> <tr> <td><b>not</b></td> <td>(barra horizontal sobre la variable)</td> </tr> <tr> <td><b>xor</b></td> <td>⊕</td> </tr> </tbody> </table> <p>Por ejemplo: <math>(A \oplus \overline{B}) \cdot (\overline{C + D})</math>.</p> <p>En palabras, se puede escribir como: (A <b>xor not</b> B) <b>and</b> (C <b>nor</b> D).</p>	Operador	Símbolo	<b>and</b>	•	<b>or</b>	+	<b>not</b>	(barra horizontal sobre la variable)	<b>xor</b>	⊕	4
Operador	Símbolo												
<b>and</b>	•												
<b>or</b>	+												
<b>not</b>	(barra horizontal sobre la variable)												
<b>xor</b>	⊕												
4.2.3	Calcule los valores de una expresión booleana utilizando tablas de verdad.	Se exigirá un máximo de tres entradas. Se debe incluir el uso de tablas de verdad para determinar si dos expresiones booleanas son lógicamente equivalentes.	3										
4.2.4	Convierta expresiones booleanas en formas más simples.	Se exigirá un máximo de tres entradas. Las conversiones se podrán hacer “algebraicamente” (mediante identidades como $x+1=1$ y las leyes de De Morgan) o mediante mapas de Karnaugh, diagramas de Venn o cualquier otro método adecuado.	2										
4.2.5	Construya un circuito lógico simple, utilizando puertas lógicas estándares, que se corresponda con una expresión booleana dada.		4										

## Unidad 4: Matemáticas y lógica en informática (continuación)

	<b>Enunciados de evaluación</b>	<b>Notas para la enseñanza</b>	<b>Obj.</b>
<b>4.2.6</b>	Construya una expresión booleana que se corresponda con un circuito lógico dado.		<b>4</b>
<b>4.2.7</b>	Explique la función de un circuito determinado.		<b>3</b>

## Unidad 5: Estructuras de datos abstractas y algoritmos

El lenguaje de programación Java proporciona algunas estructuras de datos estándares (como matrices o archivos) que son adecuadas para muchos problemas estándar. Otros problemas requieren tipos de datos más avanzados para representar estructuras más complejas, mejorar la eficiencia de los algoritmos o proporcionar una gestión de memoria más compleja.

Aunque Java aplica distintos tipos de clases de contenedor para la comodidad de los programadores, se espera que los alumnos desarrollen sus propios TDA a partir de los principios básicos.

Los alumnos del NS deben demostrar el dominio de algunas de estas técnicas en el dossier de trabajo personal; asimismo, deben ser capaces de utilizar cualquiera de estas técnicas en el examen. En esta unidad se amplían varios aspectos de las unidades 1 y 2.

### Tema 5.1: Fundamentos

3 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
5.1.1	Defina “operador” (unario y binario), “identificador”, “operando”, “parámetro real” (argumento), “parámetro formal”, “notación infija”, “notación postfija” y “notación prefija”.		1
5.1.2	Defina “pila”, “cola” y “árbol binario”.		1
5.1.3	Discuta las características y el uso adecuado de las pilas, incluyendo: almacenamiento de parámetros, manipulación de interrupciones, evaluación de expresiones aritméticas y almacenamiento de direcciones de retorno de subprogramas.		3
5.1.4	Discuta las características y el uso adecuado de las colas, incluyendo: colas de teclado, de impresión y simulaciones de colas de clientes.		3
5.1.5	Discuta las características y el uso adecuado de los árboles binarios, incluyendo: almacenamiento de claves de búsqueda, árboles de decisión y sistemas de archivos.		3

## Unidad 5: Estructuras de datos abstractas y algoritmos (continuación)

### Tema 5.2: Estructuras de datos estáticas

8 h

Las matrices se estudian con detalle en el tronco común. Este tema debe considerarse una extensión.

	Enunciados de evaluación	Notas para la enseñanza	Obj.
5.2.1	Rastree algoritmos que realicen una ordenación rápida sobre matrices lineales.		2
5.2.2	Construya algoritmos que realicen una ordenación rápida sobre matrices lineales.		4
5.2.3	Construya una tabla hash, incluyendo la generación de direcciones mediante aritmética de módulo y la manipulación de conflictos mediante la ubicación del siguiente espacio libre.	Se dará a los alumnos un algoritmo hash y un conjunto de claves o registros a los que se asignarán ubicaciones de memoria utilizando dicho algoritmo.	4
5.2.4	Rastree algoritmos que implementen una pila en una matriz.		2
5.2.5	Construya algoritmos que implementen una pila en una matriz.	Esto incluye: inicializar una pila, comprobar si la pila está vacía o llena, introducir o extraer un dato y mostrar el elemento de la parte superior. Todas las operaciones deben proteger ante los posibles desbordamientos y subdesbordamientos.	4
5.2.6	Rastree algoritmos que implementen una cola en una matriz.		2
5.2.7	Construya algoritmos que implementen una cola en una matriz.	Esto incluye: inicializar una cola, comprobar si la cola está vacía o llena, añadir un elemento en la parte posterior de una cola (añadir a la cola), eliminar un elemento de la parte delantera de una cola (quitar de la cola) y visualizar un elemento de la parte delantera de la cola. Los algoritmos deben incluir implementación lineal y circular. Todas las operaciones deben proteger ante los posibles desbordamientos y subdesbordamientos.	4

## Unidad 5: Estructuras de datos abstractas y algoritmos (continuación)

### Tema 5.3: Estructuras de datos dinámicas

14 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
5.3.1	Defina el concepto “referencia a objeto”.		1
5.3.2	Construya algoritmos que utilicen mecanismos de referencia.		4
5.3.3	Discuta las características y el uso adecuado de listas simples, doblemente enlazadas y enlazadas circularmente.		3
5.3.4	Esboce e ilustre el funcionamiento lógico de los enlaces.		2
5.3.5	Rastree algoritmos para implementar listas enlazadas.		2
5.3.6	Construya algoritmos para implementar listas enlazadas.	Esto incluye: inicializar, añadir y eliminar objetos, encontrar el objeto de la posición final, realizar una búsqueda lineal e insertar objetos en una lista. Todas las operaciones deben protegerse ante excepciones de punteros nulos.	4
5.3.7	Rastree algoritmos que implementen una pila dinámica mediante referencias.		2
5.3.8	Construya algoritmos que implementen una pila dinámica mediante referencias.	Los alumnos deben reconocer la diferencia entre este concepto anterior y la representación estática de pilas. Véanse las notas de 5.2.5.	4
5.3.9	Rastree algoritmos que implementen una cola dinámica mediante referencias.		2
5.3.10	Construya algoritmos que implementen una cola dinámica mediante referencias.	Los alumnos deben reconocer la diferencia entre este concepto y la representación estática de una cola. Véanse también las notas de 5.2.7.	4

## Unidad 5: Estructuras de datos abstractas y algoritmos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
5.3.11	Defina “padre”, “hijo-izquierdo”, “hijo-derecho” y “subárbol”.		1
5.3.12	Rastree algoritmos que implementen árboles binarios.		2
5.3.13	Construya algoritmos que implementen árboles binarios.	Esto incluye: inicializar, añadir objetos y recorrer (en orden previo, en orden, en orden posterior). Todos los recorridos de árboles deben implementarse de forma recursiva. Véase 5.5.	4
5.3.14	Esboce e ilustre la representación lógica de estructuras de datos dinámicas.		2

### Tema 5.4: Los objetos en la resolución de problemas

6 h

El ámbito de esta unidad se limita a las funciones ejemplificadas en Java. (Véase el apéndice 2).

	Enunciados de evaluación	Notas para la enseñanza	Obj.
5.4.1	Esboce las características de un objeto.	Esta explicación debería limitarse a la definición siguiente.  Un objeto es una combinación de datos y las operaciones que se pueden realizar en asociación con dichos datos. A cada parte de datos de un objeto se la conoce con el nombre de “miembro dato”, mientras que es posible denominar “métodos” a las operaciones. El estado actual de un objeto se almacena en sus miembros dato; sólo los métodos pueden modificar o acceder a dicho estado. Entre las categorías de operaciones más comunes se incluyen: construcción de objetos, operaciones que establecen (métodos mutadores) o devuelven (métodos accesorios) los miembros dato; operaciones únicas para los tipos de datos y operaciones que utiliza internamente el objeto.	2

## Unidad 5: Estructuras de datos abstractas y algoritmos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
5.4.2	Explique las características básicas y las ventajas de la encapsulación.	La encapsulación es la combinación de datos y las operaciones que actúan con dichos datos en una “unidad de programa” simple denominada objeto. La ventaja es que permite la ocultación de la información y los datos.	3
5.4.3	Explique las características básicas y las ventajas de la ocultación de información y datos.	Una vez encapsulados en un objeto, se pueden ocultar tanto los miembros dato como los detalles de la implementación de las funciones miembro. De esta forma se permite utilizar el objeto en un nivel abstracto.	3
5.4.4	Explique las características básicas y las ventajas del polimorfismo.	El polimorfismo describe la situación en la cual se puede aplicar la misma operación a objetos diferentes, donde cada objeto se comporta de la forma adecuada. No es necesario conocer los conceptos de plantillas, funciones miembro virtuales y sobrecarga de operadores. El polimorfismo permite que los objetos se utilicen de forma intuitiva; asimismo, simplifica la codificación mediante la generalización.	3
5.4.5	Explique las características básicas y las ventajas de la herencia.	La herencia permite que un objeto se derive de otro. El objeto derivado posee todos los miembros dato y funciones miembro del objeto original, así como cualquier miembro dato adicional o funciones miembro que se definan dentro del mismo. Incluso la funcionalidad previamente definida se puede volver a definir con la funcionalidad adecuada aplicada al objeto particular que la invoca. En Java, todas las clases son subclases de la clase del objeto. Cuando las funciones (incluidos los constructores) se vuelven a definir en un objeto derivado, éstas sobrescriben completamente la función original. La herencia en Java está limitada a la derivación de un objeto a partir de otro (un nivel de herencia). El lenguaje Java no admite herencia múltiple.	3
5.4.6	Rastree un algoritmo en el que se incluyan objetos.	Se debe incluir el registro del comportamiento y el estado de los objetos.	2

## Unidad 5: Estructuras de datos abstractas y algoritmos (continuación)

### Tema 5.5: Recursividad

6 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
5.5.1	Defina “recursividad”.		1
5.5.2	Discuta las ventajas y desventajas de la recursividad.	Los alumnos deben comprender que, en algunas aplicaciones, el uso de un procedimiento recursivo es breve y elegante y, por tanto, que una solución recursiva se adecua perfectamente a algunos algoritmos. Sin embargo, la recursividad no es adecuada para la mayoría de los algoritmos, debido a que los no recursivos son más eficientes.	3
5.5.3	Rastree algoritmos recursivos.	Todos los pasos y las llamadas se deben mostrar claramente. Puede que los alumnos necesiten dibujar un árbol.	2
5.5.4	Construya algoritmos recursivos.	Esta construcción se limita a un algoritmo que devuelva no más de un resultado y que contenga una o dos llamadas recursivas a sí mismo.	4
5.5.5	Implemente las siguientes estructuras: clases autorreferenciadas y recursividad.		3

## Unidad 5: Estructuras de datos abstractas y algoritmos (continuación)

### Tema 5.6: Evaluación de algoritmos

4 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
5.6.1	Indique la eficiencia de los siguientes algoritmos en notación O Mayúscula: una búsqueda lineal es $O(n)$ , una ordenación por el método de la burbuja es $O(n^2)$ , una ordenación rápida es $O(n \log n)$ , una búsqueda binaria es $O(\log n)$ y una ordenación por selección es $O(n^2)$ , dados un conjunto de datos distribuidos de forma aleatoria.	La notación O Mayúscula se utiliza para clasificar el rendimiento de los algoritmos (velocidad). Una búsqueda secuencial es $O(n)$ , lo que significa que el tiempo para buscar una matriz es proporcional al tamaño de dicha matriz. Sin embargo, una ordenación por el método de la burbuja requiere bucles anidados y, por tanto, es $O(n^2)$ , de manera que los requisitos de tiempo son proporcionales al cuadrado del tamaño de la lista. Los alumnos deben entender que la eficiencia de un algoritmo determinado puede depender de la distribución de los datos; por ejemplo, una ordenación rápida puede deteriorarse hasta $O(n^2)$ en el peor de los casos.	1
5.6.2	Analice la eficiencia de los algoritmos (los que aparecen en 5.6.1 y los de complejidad similar), en términos de notación O Mayúscula y de requisitos de almacenamiento.	Cuando los alumnos se encuentren con un algoritmo que no hayan visto antes deberán ser capaces de escribir la notación O Mayúscula para la eficiencia de dicho algoritmo.	3
5.6.3	Esboce cómo se pueden organizar las estructuras de datos de este programa de estudios para adecuarse a los requisitos de las aplicaciones.	Los alumnos deben tener en cuenta la necesidad de los tipos y estructuras de datos para las diferentes aplicaciones. Por ejemplo, las <b>pilas</b> se pueden utilizar para realizar un seguimiento de los cambios en un documento de un procesador de textos, mientras que una <b>cola</b> podría almacenar los elementos que se introducen por teclado para su procesamiento posterior, en el orden en que se hayan introducido. Los <b>árboles binarios</b> ordenados y las <b>tablas hash</b> se suelen utilizar para almacenar campos clave que, a su vez, se utilizan para recuperar rápidamente elementos de un <b>archivo de datos</b> sin ordenar.	2
5.6.4	Evalúe los algoritmos que utilizan alguna de las estructuras de datos expuestas en este programa de estudios.	Los algoritmos pueden ser los estándares, mencionados en el programa de estudios, o algoritmos de complejidad equivalente que los alumnos no hayan visto anteriormente.	3

## Unidad 6: Otras cuestiones básicas de sistemas

El rendimiento de los sistemas informáticos reales se ve afectado por todos los componentes del sistema. Los alumnos necesitan conocer las funciones de los componentes individuales y los métodos utilizados en sus interacciones. En esta unidad se amplía la unidad 3.

### Tema 6.1: Configuración del procesador

2 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
6.1.1	Describa las funciones de los siguientes componentes del procesador: acumulador, registro de instrucción y contador de programa.	No se requieren detalles (o registros) adicionales.	2
6.1.2	Explique la función de los componentes expuestos anteriormente en la ejecución de instrucciones simples en el ciclo de una instrucción de máquina.		3
6.1.3	Describa la función de un registro de interrupción.		2
6.1.4	Describa cómo los buses enlazan el procesador, la memoria de acceso aleatorio, la memoria de sólo lectura y la memoria caché.		2

### Tema 6.2: Almacenamiento en discos magnéticos

1 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
6.2.1	Esboce los detalles del almacenamiento con referencia a bloques, sectores, cilindros y cabezales.		2
6.2.2	Describa el tiempo de acceso en términos de latencia (retardo rotacional), tiempo de búsqueda y tiempo de transferencia.		2

## Unidad 6: Otras cuestiones básicas de sistemas (continuación)

### Tema 6.3: Sistemas operativos y utilidades

2 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
6.3.1	Defina “sistema operativo”.	No se requieren conocimientos sobre sistemas operativos específicos.	1
6.3.2	Explique las funciones de los sistemas operativos.	Los alumnos deben entender que un sistema operativo es una colección de programas que cubren las siguientes tareas: control de entrada/salida (E/S), mantenimiento de archivos, interfaz software/hardware, gestión de memoria, interfaz de usuario, control de ejecución del software, y seguridad. Debe incluirse la memoria virtual, aunque no se requieren conocimientos sobre limpieza (thrashing) ni paginación. Esto amplía el punto 3.3.2.	3
6.3.3	Esboce las funciones del enlazador, el cargador y el gestor de bibliotecas.		2

### Tema 6.4: Otras cuestiones básicas de redes

4 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
6.4.1	Esboce la función de los computadores utilizados en los distintos tipos de redes: WAN, LAN e Internet.	Es necesario comprender la función de los proveedores, servidores y clientes en cada uno de los tipos de redes anteriores. Los alumnos deben ser capaces de seleccionar el tipo adecuado de red para una situación dada. Deben comprender la función de las pasarelas.	2

## Unidad 6: Otras cuestiones básicas de sistemas (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
6.4.2	Describa las características de las comunicaciones necesarias para las redes.	El alumno debe estar familiarizado con los términos Ethernet, líneas telefónicas públicas y privadas, RDSI, ADSL, fibra óptica y medios inalámbricos; asimismo, deben ser capaces de seleccionar el medio más adecuado para la comunicación en una situación determinada, además de indicar las ventajas de cada método. No se requerirán detalles técnicos.	2
6.4.3	Describa la conmutación de paquetes.	Los alumnos deben entender que cuando se descompone un mensaje en paquetes, éstos pueden tomar diferentes rutas y pasar por diferentes nodos para llegar al mismo destino. Además, deben saber que estos paquetes pueden desecharse. No es necesario conocer los circuitos virtuales.	2
6.4.4	Esboce la necesidad de los protocolos en la conmutación de paquetes.	Los alumnos no necesitan conocer detalles técnicos de TCP, IP u OSI; sin embargo, deben entender que en los protocolos se incluye información esencial que permite que los paquetes se vuelvan a ensamblar en su destino en función de los requisitos del computador receptor.	2
6.4.5	Explique la necesidad de la seguridad en redes y describa cómo se puede conseguir.	Haga énfasis en la importancia de la protección dentro de una LAN mediante el otorgamiento de acceso por niveles (por ejemplo, mediante permisos sobre áreas determinadas) a diferentes usuarios, y marcando archivos como de sólo lectura. Debe quedar clara la necesidad de contar con un cortafuegos para evitar intrusiones desde el exterior.	3

## Unidad 6: Otras cuestiones básicas de sistemas (continuación)

### Tema 6.5: Comunicación computador/periféricos

6 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
6.5.1	Defina “puerto” y “protocolo de intercambio”.		1
6.5.2	Defina “acceso directo a memoria” (DMA) y “búfer”.		1
6.5.3	Defina “interrupción” y “sondeo”.		1
6.5.4	Explique cómo se controlan los dispositivos periféricos con referencia a la impresora, el módem y la unidad de disco.	Debe incluirse el uso de búferes (incluyendo el búfer doble), interrupciones, prioridades de interrupción, sondeos, acceso directo a memoria (DMA), y protocolo de intercambio en estos dispositivos.	3
6.5.5	Compare las características del DMA, los sistemas de interrupción y los sistemas de sondeo.	Los alumnos deben cubrir una interrupción por evento o por dispositivo externo, así como un sistema de sondeo. No es necesario memorizar códigos de interrupción específicos.	3
6.5.6	Compare la transmisión en serie con la transmisión paralela.		3

## Unidad 7: Organización de archivos

Normalmente, en los sistemas informáticos se utilizan varias estructuras de archivos. Los alumnos deben estar familiarizados con varias de las estructuras más comunes. En esta unidad se amplía la unidad 1.

Los libros de informática que tratan sobre este tema resultan a menudo confusos, ya que la terminología relacionada con la estructura de archivos y los métodos de acceso a los archivos se utiliza de forma incoherente. En la tabla siguiente se aclara la terminología específica que se utiliza en este programa de estudios y que se utilizará en los exámenes.

Nombre de la estructura de archivo	Detalles de la estructura	Método de acceso (búsqueda)
Archivo secuencial	Registros ordenados o sin ordenar	Acceso secuencial
Archivo parcialmente indexado	Registros ordenados	Acceso secuencial a un índice, seguido de un acceso directo al primer registro del grupo; posteriormente, acceso secuencial para buscar el registro deseado.
Archivo completamente indexado	Registros sin ordenar	Acceso secuencial al índice, seguido de acceso directo al archivo de datos.
Archivo de acceso directo	Registros sin ordenar u ordenados	Un cálculo proporciona la dirección (ubicación) de un registro, seguido de un acceso directo a dicho registro.

### Tema 7.1: Organización de archivos

10 h

	Enunciados de evaluación	Notas para la enseñanza	Obj.
7.1.1	Defina el término “campo clave”.		1
7.1.2	Esboce la organización de archivos secuenciales en registros sin ordenar y cómo dichos registros se pueden recuperar mediante acceso secuencial a través del campo clave.		2

## Unidad 7: Organización de archivos (continuación)

	Enunciados de evaluación	Notas para la enseñanza	Obj.
7.1.3	Esboce la organización de archivos secuenciales en registros ordenados y cómo dichos registros se pueden recuperar mediante acceso secuencial a través del campo clave.		2
7.1.4	Esboce la organización de archivos secuenciales parcialmente indexados.	Un archivo secuencial parcialmente indexado posee registros ordenados, con un índice independiente pero parcial. Los alumnos deben ser capaces de describir cómo se pueden recuperar registros a través del acceso al índice, seguido de un acceso directo al primer registro de un grupo y de un acceso secuencial para ubicar el registro deseado.	2
7.1.5	Esboce la organización de archivos completamente indexados.	Un archivo completamente indexado posee registros sin ordenar, con un índice independiente y completo. Los alumnos deben ser capaces de describir cómo se pueden recuperar registros a través de un acceso al índice, seguido de un acceso directo en el archivo de datos. No es necesario memorizar el concepto de índices de varios niveles.	2
7.1.6	Esboce la organización de archivos de acceso directo.	Un archivo de acceso directo puede contener registros sin ordenar. Los alumnos deben ser capaces de esbozar cómo se pueden recuperar registros a través de un cálculo seguido de un acceso directo.	2
7.1.7	Esboce la necesidad de contar con campos y registros de longitud fija y variable y cómo éstos se relacionan con los métodos de acceso directo y secuencial.		2
7.1.8	Describa el uso de algoritmos hash para guardar y recuperar registros en un archivo de acceso directo.	Los alumnos deben entender el uso de los operadores de módulo (mod, div) en la construcción de una función hash. Véase también 5.2.3.	2
7.1.9	Compare la velocidad de acceso y los requisitos de almacenamiento para los tipos de archivos mencionados en 7.1.2-7.1.8.	También se deberían incluir los medios de almacenamiento (disco, cinta). Las velocidades de acceso deben expresarse en descripciones, cálculos de iteraciones y notación O Mayúscula.	3

## Unidad 7: Organización de archivos (continuación)

	<b>Enunciados de evaluación</b>	<b>Notas para la enseñanza</b>	<b>Obj.</b>
<b>7.1.10</b>	Explique cómo difieren la organización lógica de datos y la organización física.	Por ejemplo, en un archivo secuencial completamente indexado, los registros se pueden recuperar en orden alfabético utilizando el índice, incluso aunque no estén almacenados físicamente en ese orden.	<b>3</b>
<b>7.1.11</b>	Esboce la necesidad de ordenaciones externas.	La ordenación de archivos que son demasiado grandes para la memoria principal de un computador requiere técnicas basadas en una combinación de ordenación y fusión. No es necesario memorizar algoritmos para la ordenación por fusión.	<b>2</b>
<b>7.1.12</b>	Demuestre comprensión de los diferentes tipos de flujos de datos identificados en el apéndice 2.		<b>3</b>

# ESTUDIO DE UN CASO

---

En informática, la resolución de problemas requiere una descripción clara de una situación (o un contexto) que refleje un problema real, junto con definiciones de variables concretas. Naturalmente, los exámenes imponen restricciones de tiempo considerables, especialmente cuando los alumnos deben leer grandes cantidades de texto; no obstante, en muchas preguntas será inevitable incluir algunas descripciones extensas. Además, las situaciones presentadas en los exámenes pueden ser ajenas a la experiencia de muchos alumnos. Esto se puede deber tanto a la edad de los alumnos como a sus circunstancias culturales y tecnológicas. El estudio de un caso debe ayudar a superar estas desigualdades y proporcionar otras oportunidades de evaluación. Puesto que el estudio de un caso se facilitará mucho antes del examen, permitirá que los alumnos y los profesores se familiaricen con la situación particular y el vocabulario contenido en el mismo. Los profesores podrán recopilar material de consulta pertinente para la situación. También pueden preparar a sus alumnos de otra manera, por ejemplo mediante la organización de visitas o charlas en el colegio.

## Objetivos generales del estudio de un caso

Los objetivos generales del estudio de un caso son:

- facilitar el estudio de una situación real que implique un problema que se pueda resolver mediante sistemas informáticos y se pueda describir completamente
- ilustrar la importancia social y las implicaciones de los sistemas informáticos
- utilizar situaciones relativamente actuales y, de este modo, aprovechar nuevas iniciativas o desarrollos que hayan surgido posteriormente a la elaboración de esta guía
- ofrecer una situación real sobre la que se basen las preguntas de examen de todas las secciones del programa de estudios
- intentar reducir las diferencias de rendimiento que se podrían dar debido a la comprensión limitada del material, ya que el idioma original del texto es diferente al del alumno.

## Formato

El estudio de un caso consistirá en un cuadernillo de varias páginas que contendrá información diversa. El contenido será principalmente textual, pero también podrá haber información en forma de diagramas, diagramas de flujo, algoritmos, imágenes, tablas o gráficos.

## Procedimientos

La preparación del estudio de un caso se hace cada dos años y está a cargo de examinadores experimentados de la asignatura. Para los exámenes de NM y NS se utilizará el mismo estudio, el cual se enviará a los colegios en cantidades adecuadas con la mayor anterioridad posible al examen. Durante dos años se utilizará el mismo estudio para las dos convocatorias de exámenes (mayo y noviembre). Con los cuestionarios de examen se enviarán copias sin anotaciones del estudio de un caso.

## Contenido

El estudio de un caso contendrá material relacionado con todas las secciones del programa de estudios, tanto para NM como para NS. A los alumnos del NM no se le formularán preguntas basadas en unidades del NS.

## Examen

Una pregunta de la prueba 2 de los exámenes de NS y NM requerirá la comprensión con la profundidad adecuada de la información provista en el estudio de un caso. Los alumnos podrán consultar el estudio de un caso durante el examen. Esta pregunta estructurada también podrá poner a prueba la comprensión de otras unidades del programa de estudios; asimismo, otras preguntas de la prueba 2 podrán hacer referencia a la información del estudio de un caso, pero no pondrán a prueba el contenido de éste.

# RESUMEN DE LA EVALUACIÓN

## Informática - Nivel Medio

### Primeros exámenes: 2010

COMPONENTE	PONDERACIÓN	OBJETIVOS (Ponderación aproximada)		DURACIÓN		DETALLES Y TOTAL DE PUNTOS
		1+2	3+4	Secciones	Total	
<b>Evaluación externa</b>	<b>65%</b>				<b>3 h</b>	
<b>Prueba 1</b>	32,5%	19%	13,5%		1 h 30 m	Varias preguntas obligatorias de respuesta corta (30 puntos)  Cuatro preguntas estructuradas obligatorias (40 puntos)
Sección A	14%	11,5%	2,5%	40 m aprox.		
Sección B	18,5%	7,5%	11%	50 m aprox.		
<b>Prueba 2</b>	32,5%	12%	20,5%		1 h 30 m	Tres preguntas obligatorias:
	18,5%	5%	13,5%	50 m aprox.		Dos preguntas obligatorias de respuesta larga que incluyan la construcción de un algoritmo (40 puntos)
	14%	7%	7%	40 m aprox.		Una pregunta estructurada obligatoria, basada en el estudio de un caso (30 puntos)
<b>Evaluación interna</b>	<b>35%</b>					
<b>Dossier de trabajo personal</b>	35%	20%	15%	25 h en contacto con el profesor, más tiempo de acceso a un computador.		Un proyecto en el que se trate un solo problema, que permita al alumno demostrar el dominio de los aspectos requeridos (35 puntos)

# Informática - Nivel Superior

## Primeros exámenes: 2010

COMPONENTE	PONDERACIÓN	OBJETIVOS (Ponderación aproximada)		DURACIÓN		DETALLES Y TOTAL DE PUNTOS
		1+2	3+4	Secciones	Total	
<b>Evaluación externa</b>	<b>65%</b>				<b>4h 30 m</b>	
<b>Prueba 1</b>	32,5%	19,5%	13%		2 h 15 m	Varias preguntas obligatorias de respuesta corta (40 puntos)  Seis preguntas estructuradas obligatorias (60 puntos)
Sección A	13%	10,5%	2,5%	1 h aprox.		
Sección B	19,5%	9%	10,5%	1 h 15 m aprox.		
<b>Prueba 2</b>	32,5%	13%	19,5%		2 h 15 m	Cuatro preguntas obligatorias:
	19,5%	4%	15,5%	1 h 15 m aprox.		Tres preguntas obligatorias de respuesta larga que incluyan la construcción de un algoritmo (60 puntos).
	13%	9%	4%	1 h aprox.		Una pregunta estructurada obligatoria, basada en el estudio de un caso (40 puntos)
<b>Evaluación interna</b>	<b>35%</b>					
<b>Dossier de trabajo personal</b>	35%	20%	15%	35 h en contacto con el profesor, más tiempo de acceso a un computador.		Un proyecto en el que se trate un solo problema, que permita al alumno demostrar el dominio de los aspectos requeridos (35 puntos)

# DESCRIPCIÓN DETALLADA DE LA EVALUACIÓN

---

## Evaluación externa

El modelo de evaluación de Informática está diseñado para medir el desempeño de los alumnos en función de los cuatro **objetivos** de evaluación. La evaluación se realiza mediante una combinación de exámenes externos, que se celebran al final del programa de estudios, y una evaluación interna que realizan los profesores. Estas dos estructuras clave de la evaluación se ponderan en un 65% y en un 35% respectivamente.

## Nivel Medio

Evaluación externa 65%

**Prueba 1** **(70 puntos)** **32,5%**

- La prueba 1 dura **1 hora 30 minutos**, y consiste en **dos secciones obligatorias**. La prueba está diseñada para evaluar los conocimientos globales de los alumnos sobre el contenido del plan de estudios.
- La sección A (**40 minutos** aproximadamente) consiste en **varias** preguntas obligatorias de respuesta corta que tratan principalmente sobre los objetivos 1 y 2. La puntuación máxima es de 30 puntos.
- La sección B (**50 minutos** aproximadamente) consiste en **cuatro** preguntas estructuradas obligatorias que tratan principalmente sobre los objetivos 3 y 4. La puntuación máxima para cada pregunta es de 10 puntos.

**Prueba 2** **(70 puntos)** **32,5%**

- La prueba 2 dura **1 hora 30 minutos**, y consiste en **tres preguntas obligatorias**.
- Las dos primeras preguntas (**50 minutos** aproximadamente) son de respuesta larga, compuestas por varias partes. En ellas se requiere que los alumnos construyan algoritmos en función de las situaciones adecuadas. La puntuación máxima para cada pregunta es de 20 puntos.
- La tercera pregunta (**40 minutos** aproximadamente) está estructurada en varias partes y se basa en el estudio de un caso. En esta guía se pueden encontrar más detalles sobre el estudio de un caso. La puntuación máxima es de 30 puntos.

## Calculadoras

El uso de calculadoras **no** está permitido en ningún examen de Informática.

## Nivel Superior

Evaluación externa 65%

**Prueba 1** **(100 puntos)** **32,5%**

- La prueba 1 dura **2 horas 15 minutos**, y consiste en **dos secciones obligatorias**. La prueba está diseñada para evaluar los conocimientos globales de los alumnos sobre el contenido del plan de estudios.
- La sección A (**1 hora** aproximadamente) consiste en **varias** preguntas obligatorias de respuesta corta que tratan principalmente sobre los objetivos 1 y 2. Varias preguntas son comunes a la sección A de la prueba 1 del NM (20 puntos aproximadamente). El resto de las preguntas tratan sobre unidades de NS. La puntuación máxima es de 40 puntos.
- La sección B (**1 hora 15 minutos** aproximadamente) consiste en **seis** preguntas estructuradas obligatorias que tratan principalmente sobre los objetivos 3 y 4. La puntuación máxima para cada pregunta es de 10 puntos.

**Prueba 2** **(100 puntos)** **32,5%**

- La prueba 2 es una prueba de **2 horas 15 minutos**, compuesta por **cuatro preguntas obligatorias**.
- Las tres primeras preguntas (**1 hora 15 minutos** aproximadamente) son de respuesta larga, compuestas por varias partes. En ellas se requiere que los alumnos construyan un algoritmo en función de la situación adecuada. La puntuación máxima para cada pregunta es de 20 puntos.
- La cuarta pregunta (**1 hora** aproximadamente) está estructurada en varias partes y se basa en el estudio de un caso (común al NM). En esta guía se pueden encontrar más detalles sobre el estudio de un caso. La puntuación máxima es de 40 puntos.

## Calculadoras

El uso de calculadoras **no** está permitido en ningún examen de Informática.

## Evaluación interna:

### Dossier de trabajo personal (35 puntos) 35%

El dossier de trabajo personal es un trabajo individual que se completa durante el tiempo que dura la asignatura. El dossier debe tratar **un solo problema** que se pueda resolver mediante sistemas informáticos y que tenga un **usuario final identificado**. El análisis, el diseño y la producción del sistema final deben estar **bien documentados**.

Se enfatiza en el uso de un enfoque lógico y un pensamiento analítico, desde la definición y la descomposición del problema hasta la solución de éste, mediante la construcción de las clases adecuadas que implementen algoritmos y estructuras de datos en **lenguaje Java**.

El dossier de trabajo personal lo evalúa internamente el profesor y lo modera externamente IBO siguiendo los procedimientos indicados en el *Vademécum*.

## Distribución del tiempo

### Nivel Medio

Se espera que el profesor dedique aproximadamente 25 horas de la carga horaria total al dossier de trabajo personal, incluyendo orientación para los alumnos sobre el formato, la presentación y el contenido. Parte de las horas de enseñanza del programa de estudios también suponen trabajo en relación con el dossier de trabajo personal, aunque no se incluye el tiempo de trabajo individual que los alumnos necesitarán para desarrollar y completar sus dossiers.

### Nivel Superior

Se espera que el profesor dedique aproximadamente 35 horas de la carga horaria total al dossier de trabajo personal, incluyendo orientación para los alumnos sobre el formato, la presentación y el contenido. Parte de las horas de enseñanza del programa de estudios también suponen trabajo en relación con el dossier de trabajo personal, aunque no se incluye el tiempo de trabajo individual que los alumnos necesitarán para desarrollar y completar sus dossiers.

## Elección del problema

El papel de los profesores es esencial para aconsejar al alumno en la elección del problema. Debe evitarse la elección de problemas demasiado ambiciosos, así como de problemas demasiado simplistas.

Los alumnos pueden elegir problemas que generen ellos mismos o sus profesores. El problema elegido debe tener un usuario final identificado. Los alumnos podrán compartir el mismo problema que se debe resolver o la misma situación inicial, sin embargo, **se prohíben los trabajos en colaboración**.

Se espera que los profesores proporcionen orientación educativa en cada fase del proceso de diseño. Concretamente, el profesor y el alumno deben explorar completamente el prototipo para asegurarse de que los requisitos del usuario se pueden cubrir con la capacidad para programar del estudiante y dentro del tiempo disponible. Si éste no es el caso, se debe elegir otro problema u ofrecer una solución restringida al usuario final.

Es necesario tener en cuenta el ámbito de los aspectos de dominio disponibles en el problema. El nivel de dificultad del problema debe adecuarse a la capacidad del estudiante.

## Enfoque

Los criterios para la evaluación interna suponen que el contenido del dossier de trabajo personal se divida en cuatro fases principales:

A: Análisis

B: Diseño detallado

C: Programa

D: Documentación

Los alumnos completarán las fases, preferentemente, en el orden indicado. Sin embargo, en las fases B y C puede ser necesario que los alumnos vuelvan desde C hasta B una o varias veces para refinar sus diseños detallados en una “espiral” de diseño y desarrollo. Esto dependerá también de la naturaleza del problema (abierto o cerrado) y de la capacidad del estudiante. Los profesores no deben permitir que los alumnos realicen las fases A y B posteriormente al desarrollo de la solución.

Se aconseja a los profesores que establezcan fechas límite para la finalización de las fases A, B, C y D para ayudar a que los alumnos las completen con éxito.

Los alumnos pueden elegir una metodología de diseño (estructurada, descendente, u orientada a objetos) flexible y extensible. Por tanto, quizá sea necesario que retengan documentación de diseño de fases anteriores para presentarlas en el dossier final, para que se les evalúe con los criterios B1-B3. Los profesores pueden diseñar sus propios métodos para la recopilación de dicha información de diseño (registros de diseño, carpetas que contengan tarjetas CRC, diagramas de estilo UML, etc.). Para esta asignatura se proporcionarán ejemplos en el material de ayuda al profesor.

Cuando el estudiante haya finalizado el programa, el profesor deberá ejecutarlo en presencia del estudiante para confirmar que funciona y produce la misma salida impresa que aparece en el dossier de trabajo personal.

## Sistemas automatizados de desarrollo

Algunos sistemas de programación, como los IDEs visuales, proporcionan entornos de desarrollo interactivos, con una gran variedad de prestaciones adicionales como, por ejemplo, diseño visual, manipulación de objetos y generación automática de código. Sin embargo, el uso de estos entornos se sale del ámbito de este programa de estudios.

Los alumnos pueden utilizar dichas prestaciones en el dossier de trabajo personal, pero no para las tareas de dominio. Por ejemplo, se espera que los alumnos del NM escriban sus propios algoritmos para ordenar una matriz en lugar de limitarse a ejecutar una función de biblioteca que la ordene. De igual forma, se espera que los alumnos del NS escriban sus propios algoritmos para mantener una estructura de datos enlazada en lugar de utilizar una biblioteca del sistema que ya contenga los algoritmos necesarios.

En los listados de programas que incluyan código generado automáticamente por el sistema de desarrollo se deberá identificar claramente dicho código, para diferenciarlo del código que escriba el estudiante.

## Evaluación del profesor

Los profesores evalúan a los alumnos mediante los descriptores de nivel de los criterios correspondientes, que están relacionados con los objetivos. Los niveles y los criterios deben aplicarse al trabajo realizado en el dossier de trabajo personal, **independientemente** del número de aspectos en que se demuestre el dominio. Posteriormente, se aplica un “factor de dominio”. Este factor depende del número de aspectos diferentes en que se demuestre el dominio. (Véase la sección *Dominio* de esta guía). La evaluación del dossier de trabajo personal se modera externamente.

Al aplicar los criterios de evaluación y las notas se debe tener en cuenta sólo el código que **diseñe** y **escriba** el estudiante.

El hecho de que los profesores añadan comentarios a los dossiers junto a su calificación facilita el proceso de moderación. Además, si los profesores escriben un informe por cada estudiante que justifique el nivel logro concedido en cada criterio, también se facilitará la moderación y los comentarios del moderador serán más personalizados.

## Formato del dossier

El estudiante debe entregar todo el trabajo como un único documento. Dicho trabajo se puede grapar, o presentar en un archivador o una carpeta. Toda la información necesaria para el dossier de trabajo personal debe presentarse en formato impreso. **No** está permitido adjuntar disquetes, CD-ROMs, etc. al dossier de trabajo personal ni enviarlos al moderador.

Debe haber una tabla de contenidos; asimismo, toda la documentación escrita debe estar tratada con un procesador de textos, excepto cuando sea necesario incluir anotaciones. Las ejecuciones del programa y las pantallas de muestra se pueden anotar a mano.

Todas las páginas deben estar numeradas. En el dossier de trabajo personal, la numeración puede ser secuencial (1, 2, 3, etc.) o seguir los elementos numerados en la tabla siguiente. Por ejemplo, si el proceso de diseño es el tercer elemento, las páginas se pueden numerar 3-1, 3-2, 3-3, etc. Puede que este método sea más sencillo, ya que cada elemento se puede numerar de forma secuencial a medida que se completa. La numeración de las páginas puede realizarse manualmente si los sistemas informáticos disponibles no permiten la numeración automática.

El número de páginas dedicadas a cada elemento puede variar en función de la naturaleza y complejidad del problema que se esté resolviendo, así como de la solución programada. Sin embargo, y a modo de orientación, en la tabla siguiente se ofrece un número aproximado de páginas. Esta tabla se incluye, fundamentalmente, para asegurarse de que los alumnos proporcionan la cantidad pertinente de material.

## Elementos que se deben incluir en el dossier de trabajo personal

Todos los elementos que se enumeran en la tabla siguiente deben incluirse en el dossier de trabajo personal.

Elementos	Número de páginas recomendado
Índice	
Análisis del problema	2-3
Objetivos de logro	1-2
Solución mediante prototipos	Variable
Estructuras de datos	2-5
Algoritmos	2-5
Organización modular	3-5
Tratamiento de errores	1-2
Listado de código	Variable (500-3.000 líneas)
Copia impresa con anotaciones	Variable
Evaluación de soluciones	2
Documentación de los aspectos de dominio	2
Total	Aprox. 60-100

# Criterios de evaluación interna

## Uso de criterios de evaluación y descriptores

El método de evaluación que utiliza IBO se basa en criterios. Es decir, se evalúa a cada estudiante con relación a unos criterios de evaluación identificados, no con relación al resto de los alumnos.

- Hay **twelve** criterios de evaluación para el Dossier de trabajo personal. Para cada criterio de evaluación se definen descriptores de niveles de logro que se centran en logros positivos, aunque para los niveles inferiores (1 es el nivel más bajo) se puede incluir la ausencia de consecución del logro en la descripción.
- El objetivo general es encontrar, en cada criterio, el descriptor que exprese de la manera más adecuada el nivel logrado por el alumno.
- Después de haber examinado el trabajo que se va a evaluar, lea los descriptores para cada criterio, comenzando por el nivel 1, hasta llegar al que describa un nivel de logro que el trabajo que se esté evaluando **no** haya alcanzado. El trabajo, por tanto, se describe mejor mediante el descriptor del nivel precedente y deberá registrar este nivel.
- Utilice sólo números enteros en lugar de notas parciales, como fracciones o decimales. Si un estudiante no logra un estándar descrito por ninguno de los descriptores, se debe otorgar un cero.
- Los descriptores más elevados no implican un rendimiento perfecto; los profesores no deben dudar en utilizar los extremos, incluido el cero, si representan una descripción adecuada del trabajo que se está evaluando.
- Los descriptores no se deben considerar como notas o porcentajes, aunque los niveles de descriptores se suman al final para obtener una puntuación total sobre 35. No se debe suponer que hay otras relaciones aritméticas; por ejemplo, un rendimiento de nivel 4 no es necesariamente dos veces mejor que uno de nivel 2.
- Un alumno que obtenga un nivel de logro concreto con relación a un criterio no obtendrá necesariamente los mismos niveles con relación a los otros. No se debe suponer que la evaluación global de los alumnos producirá alguna distribución concreta de puntuaciones.

## Fase A: Análisis

### Criterio A1: Análisis del problema

La documentación se debe completar al comienzo y debe contener una discusión exhaustiva del problema que se va a resolver. Este análisis debe centrarse en el **problema** y en los objetivos establecidos, no en el método para obtener la solución. En un buen análisis se debe incluir información tal como: datos de ejemplo, información y requisitos del **usuario final identificado** y, en la medida de lo posible, información general sobre cómo se ha resuelto el problema en el pasado.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación. Por ejemplo, el alumno sólo describe la solución programada.
<b>1</b>	El alumno sólo <b>indica</b> el problema que se ha de resolver o <b>muestra</b> alguna prueba de que se ha obtenido información pertinente.
<b>2</b>	El alumno <b>describe</b> el problema que se debe resolver.
<b>3</b>	El alumno describe el problema y <b>muestra pruebas</b> de que se ha obtenido información relacionada con el problema.

Normalmente esta sección del dossier de trabajo personal debe tener dos o tres páginas de extensión. Aquí se debe incluir una exposición breve del problema desde el punto de vista del usuario final. Debe existir una discusión del problema desde el punto de vista del usuario final, en la que se incluyan las necesidades del usuario y la entrada y la salida necesarias. Las pruebas, por ejemplo, pueden ser datos de ejemplo, entrevistas, etc. que se pueden reunir en un apéndice.

### Criterio A2: Objetivos de logro

En esta sección del dossier de trabajo personal se indicarán claramente los objetivos de la solución al problema.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno <b>indica algunos</b> objetivos de la solución.
<b>2</b>	El alumno <b>describe la mayoría</b> de los objetivos de la solución.
<b>3</b>	El alumno <b>relaciona todos</b> los objetivos de la solución con el análisis del problema.

Normalmente esta sección del dossier de trabajo personal debe tener una o dos páginas de extensión. Los objetivos deben incluir una facilidad de uso y un rendimiento mínimos. En los criterios posteriores se hará referencia a estos logros, por ejemplo, en los criterios C3 (Éxito del programa) y D2 (Evaluación de soluciones).

### Criterio A3: Solución mediante prototipos

A la solución mediante prototipos la **debe** preceder un diseño inicial para lograr alguno de los objetivos principales determinados como logros de objetivos. Se debe crear un prototipo de la solución.

Un prototipo es: “La construcción de una versión simple de la solución que se utiliza como parte del proceso de diseño para demostrar cómo funcionará el sistema”.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno incluye un diseño inicial <b>y un prototipo</b> que no se corresponden.
<b>2</b>	El alumno incluye un diseño inicial y un prototipo que <b>se corresponden</b> .
<b>3</b>	El alumno incluye un diseño inicial y un prototipo completo que se corresponden y <b>documenta los comentarios del usuario</b> en la evaluación del prototipo.

No es necesario que el prototipo sea funcional; se puede construir con herramientas como Visual Basic, PowerPoint, Mac Paint o Corel Draw para un programa simple en Java. El objetivo es mostrar al usuario cómo se espera que funcione el sistema, qué entradas son necesarias y qué salidas se producirán. Es necesario incluir varias capturas de pantalla para que el usuario pueda evaluar la solución adecuadamente. El prototipo, en su forma más simple, puede ser una serie de dibujos nítidos generados por computador, un esbozo jerárquico de las funciones en modo texto o una serie de capturas de pantallas.

La documentación de los comentarios del usuario puede ser, por ejemplo, un informe sobre la opinión del usuario sobre el prototipo.

## Fase B: Diseño detallado

La ordenación de los criterios de B1 a B3 no implica que los alumnos deban desarrollar o documentar sus diseños en este orden. Esto dependerá de la metodología que se adopte.

### Criterio B1: Estructuras de datos

**En la fase de diseño**, los alumnos deben elegir estructuras de datos que cumplan completamente los requisitos para el almacenamiento de datos del problema y que permitan escribir algoritmos claros y eficientes. Las estructuras de datos deben ser completamente compatibles con los objetivos de la solución (criterio A2). Las clases elegidas deben ser lógicas: los datos deben ser adecuados para los objetos en cuestión y los métodos deben ser adecuados para los datos proporcionados. En esta sección del dossier de trabajo personal se pueden incluir definiciones de clases, estructuras de archivos, tipos de datos abstractos (sobre todo en el Nivel Superior) y algunas consideraciones sobre alternativas.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno <b>esboza algunos</b> de los tipos o las estructuras de datos que se van a utilizar en la solución.
<b>2</b>	El alumno <b>describe algunos</b> de los tipos o las estructuras de datos que se van a utilizar y ha proporcionado datos de ejemplo.
<b>3</b>	El alumno ha <b>discutido</b> e <b>ilustrado con claridad todos</b> los tipos o las estructuras de datos que se van a utilizar para resolver el problema y ha proporcionado datos de ejemplo en <b>todos los casos</b> .

Normalmente esta sección debe tener de dos a cinco páginas de extensión.

En esta sección se deben discutir las estructuras de datos y los miembros de datos que se van a utilizar en la solución programada. Para conseguir el nivel 3 en el criterio B1 es necesario mostrar los datos de ejemplo, los bocetos, ilustraciones y la discusión sobre el modo en que los objetos de datos se modificarán durante la ejecución del programa.

### Criterio B2: Algoritmos

**En la fase de diseño**, los alumnos deben elegir un algoritmo que admita completamente los procesos necesarios para conseguir los objetivos de la solución (criterio A2) y que proporcione compatibilidad suficiente con las estructuras de datos necesarias. Las clases elegidas deben ser lógicas: los métodos deben ser adecuados para los datos proporcionados. Los alumnos deben incluir parámetros y valores de retorno.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno <b>esboza algunos</b> de los algoritmos que se van a utilizar en la solución.
<b>2</b>	El alumno <b>describe la mayoría</b> de los algoritmos que se van a utilizar y ha proporcionado detalles sobre los parámetros y los valores de retorno.

Normalmente esta sección debe tener de dos a cinco páginas de extensión.

Puede ser una lista o un esbozo de todos los algoritmos, presentados como texto, posiblemente a modo de esbozo. Para los algoritmos estándares (como los de búsqueda u ordenación), basta con nombrarlos; los algoritmos que no sean estándares se deben describir con más detalle.

### Criterio B3: Organización modular

**En la fase de diseño**, los alumnos deben elegir módulos que incorporen las estructuras de datos y los métodos necesarios para la solución (criterios B1 y B2) de una forma lógica. Las estructuras de datos deben ser completamente compatibles con los objetivos de la solución (criterio A2). La organización debe estar estructurada de manera que muestre claramente las conexiones entre los módulos (descomposición jerárquica o dependencias entre clases). También deben presentarse las conexiones entre módulos, algoritmos y estructuras de datos.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno <b>esboza algunos</b> de los módulos que se van a utilizar en la solución.
<b>2</b>	El alumno <b>describe la mayoría</b> de los módulos que se van a utilizar, y <b>muestra las conexiones</b> entre dichos módulos.

Normalmente, esta sección debe tener de tres a cinco páginas de extensión.

Se admiten varias presentaciones. Algunas de las posibilidades son:

- un diagrama de descomposición jerárquica descendente que contenga los nombres de los módulos, en el que se muestren las conexiones entre dichos módulos y los detalles de qué estructuras de datos y métodos están conectados con qué módulos (o parte de éstos)
- un esquema de texto que muestre la descomposición jerárquica (equivalente a la anterior)
- una copia impresa de las tarjetas CRC en las que se muestren las dependencias entre las clases colaboradoras, junto con los detalles de qué estructuras de datos y métodos están conectados con qué clases (o parte de éstas).

El diseño se evalúa independientemente de la fase de programación (fase C). El diseño debe ser completo, lógico y utilizable, aunque el alumno puede alejarse del mismo o ampliarlo durante la fase C, sin que ello conlleve ninguna penalización.

## Fase C: Programa

Los listados de los programas deben contener **todo** el código que escriban los alumnos. Si en un listado se muestra código generado automáticamente por el sistema de desarrollo o código copiado de otra fuente, es necesario identificar claramente dicho código y distinguirlo del que hayan escrito los alumnos. Al aplicar los criterios de evaluación sólo se debe tener en cuenta el código que **diseñen y escriban** los alumnos.

### Criterio CI: Uso de un buen estilo de programación

Un buen estilo de programación se puede demostrar con un código de programa que se pueda leer fácilmente, incluso si lo lee un programador que nunca haya utilizado el programa. Deberán incluirse métodos escritos en Java breves y claramente estructurados, comentarios suficientes y adecuados, nombres de identificadores significativos y un esquema de sangría coherente.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El listado del programa demuestra <b>alguna</b> atención por un buen estilo de programación.
<b>2</b>	El listado del programa demuestra <b>en su mayor parte</b> atención por un buen estilo de programación.
<b>3</b>	Todas las partes del listado del programa demuestran una atención <b>considerable</b> por un buen estilo de programación.

Un programa normal debe tener aproximadamente 1.000-3.000 (NS) o 500-2.000 (NM) líneas de código.

Se deben incluir comentarios para describir el objetivo y los parámetros de cada método, así como cuando el código sea difícil de comprender.

El programa debe demostrar el uso de buenas técnicas de programación. Estas técnicas deben incluir:

- una cabecera de identificación en la que se indique el nombre del programa
- autor, fecha y colegio
- computador utilizado, IDE y objetivo.

El programa debe contar con una buena documentación interna, incluyendo:

- declaraciones de constantes, tipos y variables, con comentarios explicativos
- identificadores con nombres descriptivos
- objetos claramente separados y con comentarios para sus parámetros
- sangría adecuada que ilustre las diferentes estructuras de programación.

Normalmente el nivel de logro 2 se otorgará cuando se demuestren dos de estos aspectos o más. El nivel de logro 3 se otorgará cuando se demuestren tres o más.

## Criterio C2: Tratamiento de errores

Este criterio hace referencia a la detección y el rechazo de entradas de datos erróneas, así como a evitar los errores más comunes en tiempo de ejecución ocasionados por errores de cálculo y errores en ficheros de datos. No se espera que los alumnos detecten ni corrijan errores intermitentes o graves de hardware (como p.ej. señales de falta de papel en la impresora) o de unidades de disco dañadas, ni que eviten la pérdida de datos durante un corte de electricidad.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno <b>incluye</b> documentación en la que se muestran <b>algunas</b> funciones para el tratamiento de errores en el programa o documenta sólo <b>un</b> tipo de entrada o salida.
<b>2</b>	El alumno incluye documentación en la que se muestran <b>bastantes</b> funciones para el tratamiento de errores en el programa y documenta <b>más de un</b> tipo de entrada o salida.
<b>3</b>	El alumno documenta <b>completamente</b> el tratamiento de errores de <b>cada</b> método de entrada y salida del programa.

Normalmente esta sección debe tener de una a dos páginas de extensión.

Para este criterio, los alumnos deben intentar detectar tantos errores como sea posible. La documentación del dossier puede tomar varias formas.

Por ejemplo, los alumnos pueden resaltar las partes correspondientes del código del programa o crear una tabla con dos columnas, una en la que se identifique cualquier posibilidad de error y otra en la que se muestren las acciones llevadas a cabo para detectar dichos errores. No se espera que se produzcan salidas adicionales para esta sección.

## Criterio C3: Éxito del programa

En este criterio, “prueba” hace referencia a la salida impresa mencionada en el criterio D1.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno <b>incluye</b> pruebas de que el programa <b>funciona parcialmente</b> . El alumno consigue <b>alguno</b> de los objetivos del criterio A2.
<b>2</b>	El alumno incluye pruebas de que el programa funciona <b>correctamente</b> . El alumno consigue <b>la mayoría</b> de los objetivos del criterio A2.
<b>3</b>	El alumno incluye pruebas de que el programa funciona correctamente. El alumno consigue <b>todos</b> los objetivos del criterio A2.

El profesor debe ejecutar el programa con el alumno para confirmar que funciona y que produce la misma salida impresa que se ha enviado con el dossier de trabajo personal.

## Fase D: Documentación

### Criterio D1: Inclusión de una copia impresa anotada de la salida de las pruebas

La copia impresa de la salida de las pruebas debe demostrar que el programa alcanza los objetivos indicados en el criterio A2. La salida **debe contener anotaciones** (que pueden hacerse a mano). El profesor debe confirmar que cada alumno ha completado realmente las pruebas como indica la documentación. (Véase el *Vademécum*).

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno incluye un conjunto <b>incompleto</b> de muestras de salida.
<b>2</b>	El alumno incluye un conjunto incompleto de muestras de salida <b>con anotaciones</b> .
<b>3</b>	El alumno incluye un conjunto <b>en su mayoría completo</b> de muestras de salida con anotaciones.
<b>4</b>	El alumno incluye un conjunto <b>completo</b> de muestras de salida con anotaciones, en las que prueba todos los objetivos del criterio A2.

Para demostrar que se han probado las diferentes bifurcaciones del programa, se debe incluir la salida impresa de una o más ejecuciones de prueba; no es suficiente probar sólo un conjunto de datos válidos. La copia impresa enviada debe demostrar las respuestas del programa ante datos inadecuados o erróneos, así como ante datos válidos. Por tanto, debe resultar evidente la utilidad de las rutinas para el tratamiento de errores mencionadas anteriormente. Aunque se debe incluir al menos una ejecución completa de prueba en el dossier, no es necesario que la salida impresa refleje todas las pulsaciones de teclado de cada ejecución de prueba. Para ilustrar pruebas de diferentes aspectos del programa, se deben realizar operaciones de “cortar y pegar” en ejecuciones de prueba adicionales.

Todas las ejecuciones de prueba deben contener anotaciones de forma que el alumno indique qué aspecto del programa se está probando. Las muestras de salida **nunca** se deben modificar a mano, borrar u ocultar.

La muestra de salida se puede “capturar” y combinar electrónicamente con anotaciones explicativas en un único documento. Sin embargo, no está permitido alterar o volver a formatear dicha muestra de ninguna forma (excepto para añadir números de página o anotaciones con el fin de destacar la facilidad de uso o las funciones para el tratamiento de errores, tal como se describe anteriormente), especialmente si dichas alteraciones pueden proporcionar una impresión poco realista del rendimiento del programa. Algunos ejemplos de este tipo de “abusos” son: alineación de texto originalmente no alineado, adición de color u otros efectos especiales, modificación de salidas numéricas incorrectas o eliminación de pruebas de errores.

## Criterio D2: Evaluación de soluciones

La sección evaluación/conclusión debe ser un análisis crítico de la solución resultante. La eficacia se podrá discutir en relación con la descripción original del problema y el logro de los objetivos expuestos en el criterio A2. La eficiencia se puede discutir en términos generales; por ejemplo, no se requiere la notación O Mayúscula. Las mejoras sugeridas y las posibles extensiones deberán ser realistas, por ejemplo, no deberían incluir declaraciones como “el programa sería mejor si incorporase algunas técnicas de inteligencia artificial, tales como el reconocimiento de voz y el análisis sintáctico del lenguaje natural”.

<b>0</b>	El alumno no alcanza el nivel de ninguno de los descriptores que se exponen a continuación.
<b>1</b>	El alumno sólo <b>esboza</b> la solución.
<b>2</b>	El alumno <b>esboza</b> la solución y considera <b>parcialmente</b> la eficacia, la eficiencia y las posibles mejoras.
<b>3</b>	El alumno <b>discute</b> la eficacia y la eficiencia de la solución <b>y sugiere</b> mejoras y procesos alternativos.
<b>4</b>	El alumno <b>sugiere</b> aproximaciones alternativas a la solución y al proceso de diseño.

Normalmente esta sección del dossier debe tener dos páginas de extensión.

La evaluación/conclusión debe incluir reflexiones sobre la eficacia de la solución programada para el problema original. Debe discutir las respuestas a las preguntas siguientes.

- ¿Funciona?
- ¿Alcanza los objetivos planteados?
- ¿Funciona con un conjunto de datos, pero no con otros?
- ¿Posee el programa alguna limitación en su forma actual?
- ¿Qué características adicionales podría tener el programa?
- ¿Era adecuado el diseño inicial?

Una evaluación exhaustiva debería discutir también las posibles mejoras futuras que se puedan realizar en el programa.

## Fase E: Aproximación holística

### Criterio E: Aproximación holística al dossier

El dossier de trabajo personal debe considerarse como un proceso continuo que requiere la interacción entre el alumno y el profesor. El alumno debe tomar conciencia de las expectativas del profesor desde el principio del proceso. Por su parte, el profesor debe justificar con un comentario escrito cada nivel de logro que otorgue. Los ejemplos que se muestran a continuación para cada nivel de criterio están orientados al profesor; cada profesor debe utilizarlos con criterio al juzgar los niveles.

<b>0</b>	El alumno no muestra <b>ningún</b> compromiso. Por ejemplo, no ha participado en las discusiones de clase sobre el dossier, no ha enviado el trabajo necesario y/o ha incumplido varias fechas límite.
<b>1</b>	El alumno muestra un compromiso <b>mínimo</b> . Por ejemplo, ha participado mínimamente en las discusiones de clase sobre el dossier, ha cumplido con la mayoría de las fechas límite, o ha participado en alguna discusión que ha iniciado el profesor pero no ha aprovechado las oportunidades disponibles para desarrollar o mejorar el dossier.
<b>2</b>	El alumno muestra un <b>buen</b> compromiso. Por ejemplo, ha participado en las discusiones de clases sobre el dossier, ha iniciado discusiones con el profesor y/o el resto de la clase y/o se ha implicado completamente en el desarrollo del dossier.

Para el obtener el mayor nivel en este criterio, el alumno debe destacar en áreas como las que se enumeran a continuación. Ésta no es una lista completa: se recomienda a los profesores que añadan sus propias expectativas.

El alumno:

- ha participado activamente en todas las fases del desarrollo del dossier
- ha demostrado una comprensión total de los conceptos asociados a su dossier
- ha demostrado iniciativa
- ha demostrado perseverancia
- ha mostrado perspicacia
- se ha preparado correctamente para cumplir las fechas límite que estableció el profesor.

# DOMINIO

---

Los alumnos deben demostrar el dominio de varios aspectos de Java mediante pruebas en sus dossiers de trabajo personal.

## Aspectos de dominio

### Nivel Medio

Para conseguir un factor de dominio de 1,0, los alumnos deben llegar a dominar al menos **10** de los 15 aspectos siguientes.

1. Matrices
2. Objetos definidos por el usuario
3. Objetos como registros de datos
4. Selección simple (**if-else**)
5. Selección compleja (**if** anidados, **if** con varias condiciones o **switch**)
6. Bucles
7. Bucles anidados
8. Métodos definidos por el usuario
9. Métodos con parámetros (los parámetros deben ser útiles y utilizarse dentro del cuerpo del método) definidos por el usuario
10. Métodos definidos por el usuario con valores de retorno adecuados (primitivos u objetos)
11. Ordenación
12. Búsqueda
13. E/S por archivo
14. Uso de bibliotecas adicionales (como utilidades y bibliotecas gráficas **no incluidas** en el apéndice 2: Subconjunto de herramientas de Java para el examen)
15. Uso de centinelas o indicadores

Se prevé que esta lista proporcione a los alumnos la opción de elegir algoritmos y estructuras de datos adecuadas al problema, en lugar de idear una solución que se corresponda con los aspectos de dominio.

Cuando un aspecto incluya a los otros, todos se considerarán conseguidos; por ejemplo el aspecto 10 también satisfará el 8 y el 9 (siempre que se demuestre que el uso **no es trivial**, está **bien documentado** y es **adecuado**).

## Nivel Superior

Para conseguir un factor de dominio de 1,0, los alumnos deben llegar a dominar al menos **10** de los 19 aspectos siguientes.

1. Adición de datos a una instancia de la clase **RandomAccessFile** mediante la manipulación directa del puntero de archivo, utilizando el método **seek**.
2. Eliminación de datos de una instancia de la clase **RandomAccessFile** mediante la manipulación directa del puntero de archivo, utilizando el método **seek**. (Los objetos o las primitivas de datos se pueden marcar como eliminados mediante un campo de indicador. Por tanto, los archivos pueden estar ordenados o sin ordenar).
3. Búsqueda de los datos especificados en a file.
4. Recursividad
5. Fusión de dos o más estructuras de datos ordenados.
6. Polimorfismo
7. Herencia
8. Encapsulación
9. Análisis sintáctico de un archivo de texto u otro flujo de datos.
10. Implementación de una estructura de datos compuesta jerárquica. En esta definición, una estructura de datos compuesta es una clase que implementa una estructura de datos de tipo registro. Una estructura de datos compuesta jerárquica es aquella que contiene más de un elemento y en la que al menos uno de los elementos es una estructura de datos compuesta. Algunos ejemplos son una matriz o una lista enlazada de registros, un registro que tenga un campo que sea a su vez un registro, o una matriz.
11. Uso de cualquiera de los cinco factores de dominio del Nivel Medio: sólo se puede aplicar una vez.
- 12-15. Se pueden otorgar hasta cuatro aspectos por la implementación de tipos de datos abstractos (TDA) según la tabla denominada “Implementación de TDA”.

Un TDA se puede implementar como una clase o interfaz que contenga miembros dato y métodos adecuados para dicho TDA. El número de aspectos de dominio concedido dependerá de la minuciosidad y corrección de la implementación del alumno; en la tabla siguiente se muestran algunos ejemplos.

16. Use of additional libraries (such as utilities and graphical libraries **not included** in appendix 2 Java Examination Tool Subsets)
17. Inserting data into an ordered sequential file without reading the entire file into RAM.
18. Deleting data from a sequential file without reading the entire file into RAM.
19. Arrays of two or more dimensions.

“No trivial” significa que el programador debe demostrar que el programa se beneficia del uso del aspecto.

Cuando un aspecto incluya a los otros, todos se considerarán conseguidos (siempre que se demuestre que el uso **no es trivial**, está **bien documentado** y es **adecuado**).

## Implementación de TDA

Nombre del TDA	Un aspecto	Dos aspectos	Tres aspectos	Cuatro aspectos
Criterios generales	Se implementa un TDA incompleto.	Se implementa un TDA con todos los métodos clave.	Se implementa un TDA que incluye alguna verificación de errores.	Se implementa un TDA completamente y de forma robusta.
Listas, implementadas mediante referencias (es decir, una lista enlazada dinámicamente).	Una clase de tipo nodo con los constructores y métodos adecuados para definir y recuperar elementos de datos.	Se implementan métodos para añadir a o eliminar desde la cola o la cabeza de la lista.	Se realizan verificaciones adecuadas para detectar errores como el intento de recuperar un elemento de una lista vacía o la inserción de un mismo elemento dos veces.	Se comprueban todas las condiciones de error y se implementan todos los métodos adecuados. En una lista doblemente enlazada, éstos pueden ser: size                   insertHead isEmpty               insertTail first                   insertAfter last                   insertBefore before after
Árbol (es suficiente un árbol binario simple y ordenado, utilizando matrices o instancias de objetos enlazados dinámicamente).	Una clase o interfaz con los constructores y métodos adecuados para definir y recuperar elementos de datos.	Se implementan métodos para añadir a o eliminar desde el punto correcto del árbol.	Se realizan verificaciones adecuadas para detectar errores como el intento de recuperar un elemento de un árbol vacío o la inserción de un mismo elemento dos veces.	Se comprueban todas las condiciones de error y se implementan todos los métodos adecuados. En un árbol binario simple y ordenado, éstos pueden ser: size isEmpty root parent leftChild rightChild

<b>Nombre del TDA</b>	<b>Un aspecto</b>	<b>Dos aspectos</b>	<b>Tres aspectos</b>	<b>Cuatro aspectos</b>
Pila implementada dinámica o estáticamente.	Una clase o interfaz con los constructores y métodos adecuados para añadir y eliminar elementos.	Se añaden métodos para comprobar pilas llenas y vacías.	Se realizan verificaciones adecuadas para detectar errores como el intento de recuperar un elemento de una pila vacía.	Métodos posibles: push pop top isEmpty isFull size
Cola implementada dinámica o estáticamente.	Una clase o interfaz con los constructores y métodos adecuados para añadir y quitar elementos de una cola.	Se añaden métodos para comprobar colas llenas y vacías.	Se realizan verificaciones adecuadas para detectar errores como el intento de recuperar un elemento de una cola vacía.	Métodos posibles: enqueue dequeue front rear isEmpty isFull size
Tabla hash implementada en una matriz.	Una clase o interfaz con los constructores y métodos adecuados para insertar y eliminar elementos.	Se añaden métodos para comprobar tablas llenas y claves duplicadas.	Se realizan verificaciones adecuadas para detectar errores como el intento de recuperar una clave que no existe; los conflictos se tratan en forma apropiada.	Métodos posibles: hashFunction insertKey removeKey isDuplicate isEmpty isFull size

No es posible ofrecer una lista completa; los profesores deberán emplear su propio juicio para la implementación de este aspecto de dominio. No se recomienda que los profesores animen a los alumnos a desarrollar grafos, montículos, diccionarios, colas de prioridad y TDA de complejidad similar.

“Completo y robusto” significa que todas las necesidades de la solución se cumplen sin fallo.

## Factor de dominio

Los criterios de dominio debe evaluarse de la misma manera en el NM y el NS. Por tanto, los criterios se deberán aplicar de la misma forma a los dossiers de trabajo personal tanto del NM como del NS.

Los alumnos del NM y los del NS deben demostrar el dominio en **al menos 10** aspectos. Los descriptores de niveles y criterios se deberán aplicar en primer lugar al trabajo en el dossier, **independientemente** de los aspectos de dominio demostrados.

A continuación, se determinará el factor de dominio adecuado a partir de la tabla que se muestra a continuación. Después de aplicar el factor de dominio, la puntuación final del alumno se deberá redondear al entero más cercano (0,5 o más redondea al siguiente entero superior).

Los alumnos también deben documentar los dossiers minuciosamente. Para mostrar el dominio de un aspecto, **no** es suficiente que el alumno lo **utilice** en un programa: en la documentación escrita, los alumnos deben incluir información sobre **por qué** es adecuada una estructura de datos, **cómo** se utiliza (por ejemplo, cómo se añaden, eliminan o buscan nodos) y **dónde** se utiliza en el programa. En otras palabras, los alumnos deben ofrecer referencias cruzadas entre la documentación y los procedimientos específicos dentro del programa.

Número de aspectos en los que el alumno demuestra dominio	Factor de dominio
10 o más	1,0
9	0,9
8	0,8
7	0,7
6	0,6
5	0,5
4	0,4
3	0,3
0, 1 o 2	0,2

## Ejemplos

- Un alumno logra 29 puntos al aplicarse los criterios de evaluación, y demuestra dominio en ocho aspectos diferentes.  
Por tanto, la puntuación final es  $29 \times 0,8 = 23,2 = 23$ .
- Un alumno logra 32 puntos al aplicarse los criterios de evaluación, y demuestra dominio en doce aspectos diferentes.  
Por tanto, la puntuación final es  $32 \times 1,0 = 32$ .

## Documentación de los aspectos de dominio

Los aspectos de dominio se deben enumerar con:

- los números de las páginas en las que aparecen en el listado del código
- una breve descripción de cómo su uso beneficia a la solución.

# APÉNDICE I

---

## Glosario de términos informáticos

Ninguna lista de términos de informática puede ser exhaustiva. En este glosario se incluyen términos pertinentes a la asignatura de Informática del Programa del Diploma del BI y no son necesariamente aplicables de forma universal. Los libros de texto no siempre coinciden con las definiciones de algunos términos; sin embargo, la ambigüedad debe reducirse en los casos en que se utilice más de una palabra para el mismo concepto. Para ello, los términos deben emplearse en el sentido que indica la definición del glosario. Asimismo y para mayor claridad, a continuación de las definiciones se incluye un glosario inglés–español.

Los términos que se aplican sólo en el Nivel Superior se indican mediante <sup>NS</sup> en la columna central.

estructura de datos abstracta <i>abstract data structure</i>	<sup>NS</sup> Forma de organizar los datos y los procedimientos y funciones relacionados.
métodos accesoros <i>accessor methods</i>	<sup>NS</sup> Métodos que no alteran el estado o los atributos de un objeto; su objetivo es devolver información.
acumulador <i>accumulator</i>	<sup>NS</sup> Registro de almacenamiento ubicado en la ALU que contiene temporalmente datos mientras éstos se están procesando y antes de que se transfieran a la memoria.
convertidor A/D <i>A–D converter</i>	Convertidor analógico/digital. Dispositivo para convertir señales analógicas en señales digitales para un procesamiento posterior en un computador; en ocasiones se denomina “digitalizador”. Un convertidor digital/analógico (D/A) opera en la dirección opuesta.
ADSL (Línea asimétrica digital de abonado) <i>ADSL (Asymmetrical Digital Subscriber Line)</i>	<sup>NS</sup> Tecnología que aumenta la tasa de datos en las líneas telefónicas existentes mediante la integración de voz y transmisión de datos digitales. Para tener acceso a esta tecnología es necesario contar con un módem especial.
bus de direcciones <i>address bus</i>	Vía de comunicación desde la memoria a la unidad de procesamiento que porta las direcciones de memoria desde y hacia las que se transfieren los datos. Véase las definiciones de “bus” y “bus de datos”.

algoritmo <i>algorithm</i>	Conjunto ordenado de instrucciones bien definidas para la resolución de un problema en un número finito de pasos.
ALU <i>ALU</i>	Véase la definición de “unidad aritmético-lógica”.
datos analógicos <i>analog data</i>	Representación y medida del rendimiento o comportamiento de un sistema por medio de entidades físicas que varían continuamente como, por ejemplo, corriente, voltajes, etc. Véase también la definición de “datos digitales”.
<b>and</b> <i>and</i>	La salida de “and” es verdadera si todas las sentencias son verdaderas, y falsa si cualquier sentencia es falsa.
applet (Java) <i>applet (java)</i>	Programa que se ejecuta en el contexto de un navegador.
aplicación (Java) <i>application (java)</i>	Programa que se ejecuta cuando lo traduce un compilador de Java.
archivo <i>archive</i>	Datos que representan un registro de datos almacenados y procesados en un momento concreto, que se mantienen almacenados para una consulta posterior o por motivos legales.
argumento <i>argument</i>	<sup>NS</sup> Valor u objeto que pasa a un método cuando se lo llama.
unidad aritmético lógica (ALU) <i>arithmetic and logic unit (ALU)</i>	Parte del computador que realiza operaciones aritméticas, lógicas y otras relacionadas.
matriz <i>array</i>	<ol style="list-style-type: none"> <li>1. Colección de datos de una o más dimensiones.</li> <li>2. En los lenguajes de programación, conjunto de objetos de datos con atributos idénticos; a cada uno de estos elementos se puede hacer referencia única mediante indexación.</li> </ol>
ASCII: Código estándar estadounidense para el intercambio de información <i>ASCII: American Standard Code for Information Interchange</i>	Principal conjunto de caracteres de codificación que se utiliza en computadores para transferir datos textuales entre aplicaciones. En este conjunto se utilizan 8 bits para cada código de carácter, uno de los cuales es un bit de verificación, que comprueba los 7 bits necesarios para representar un carácter. El código ASCII admite la mayoría de los alfabetos europeos. El código Unicode admite la mayoría de alfabetos y cada vez se utiliza más en la transferencia de datos. Véase también la definición de “Unicode”.
atributo <i>attribute</i>	<sup>NS</sup> Elemento de datos contenido en un objeto tal como se especifica en la clase de dicho objeto.

B <i>B</i>	Byte.
copia de seguridad (archivo) <i>back-up (file)</i>	Segunda copia de un archivo, para utilizar en caso de que se dañe el archivo original.
árbol equilibrado <i>balanced tree</i>	<sup>NS</sup> Árbol en el que los subárboles derecho e izquierdo de cualquier nodo difieren en altura, como máximo, en un elemento. Véase también la definición de “árbol no equilibrado”.
código de barras <i>bar code</i>	Patrón de líneas verticales que se distinguen entre sí por el grosor. Para transferir datos a un computador, se puede leer con un lector de códigos de barras.
lector de códigos de barras <i>bar code reader</i>	Lector óptico que reconoce e interpreta códigos de barras.
base <i>base</i>	Fundamento de una notación o un sistema de numeración, que define un sistema numérico figurativo mediante representación posicional. En un sistema decimal la base es 10, en uno hexadecimal es 16 y en un sistema binario es 2.
procesamiento por lotes <i>batch processing</i>	Método para el procesamiento de datos, en los que las transacciones se obtienen y se preparan como entradas en el computador para que se procesen como una única unidad (por ejemplo, una nómina de empleados).
comportamiento <i>behaviour</i>	<sup>NS</sup> Forma en que reacciona un objeto ante los métodos aplicados.
notación O mayúscula <i>BigO notation</i>	<sup>NS</sup> Notación utilizada para describir el rendimiento relativo (velocidad) de un algoritmo.
operador binario <i>binary operator</i>	<sup>NS</sup> Operador que combina dos operandos para dar un resultado simple; por ejemplo, adición, multiplicación, división, mod y div. Véase también la definición de “operador unario”.
búsqueda binaria <i>binary search</i>	Búsqueda en la que, en cada paso de la misma, el conjunto de elementos de datos se divide en dos hasta encontrar el elemento que se está buscando. Véase también la definición de “búsqueda secuencial”.
árbol binario <i>binary tree</i>	<sup>NS</sup> Árbol en que cada nodo tiene como máximo dos hijos.

bit (b) <i>bit (b)</i>	Dígito binario. Unidad mínima de información para el almacenamiento y la transmisión de datos. Cada bit puede tomar un valor de “0” o “1”.
bloque <i>block</i>	Unidad mínima de datos que se puede transferir entre la memoria y el almacenamiento virtual en una operación.
BMP <i>BMP</i>	Extensión de los archivos de mapas de bits.
expresión booleana <i>boolean expression</i>	Expresión cuyo valor puede ser verdadero (V) o falso (F).
bps <i>bps</i>	Bits por segundo.
navegador <i>browser</i>	Programa que normalmente se utiliza para proporcionar acceso interactivo a la información de Internet, de donde recupera y muestra páginas web.
ordenación por el método de la burbuja <i>bubble sort</i>	Búsqueda en la cual los dos primeros elementos que se van a ordenar se examinan y se intercambian, si es necesario, para situarlos en el orden especificado; el segundo elemento, a continuación, se compara con el tercero (intercambiándose si es necesario), el tercero se compara con el cuarto y el proceso se repite hasta que se hayan examinado todas las parejas y todos los elementos estén en el orden adecuado. Véase también las definiciones de “ordenación por inserción”, “ordenación por selección” y “ordenación rápida”.
búfer <i>buffer</i>	Parte del almacenamiento utilizada para retener datos de entrada o salida de forma temporal.
bus <i>bus</i>	Vía de comunicación utilizada para enviar señales entre los componentes internos de un computador. Los componentes pueden compartir el mismo bus pero no pueden transmitir simultáneamente. Véase las definiciones de “bus de datos” y “bus de direcciones”.
topología de bus <i>bus topology</i>	Red en la que todos los dispositivos están conectados a un cable central denominado “bus”. Véase también las definiciones de “topología de estrella” y “topología de árbol”.
byte (B) <i>byte (B)</i>	Conjunto de bits considerados como una unidad; normalmente está formado por 8 bits y se corresponde con un carácter simple de información.
cable <i>cable</i>	Fibra de vidrio o alambre que se utiliza para conectar computadores en una red. Los más comunes son los de cobre (coaxial y par trenzado) y fibra de vidrio (cable de fibra óptica).

caché <i>cache</i>	Parte del almacenamiento principal ubicado entre la memoria principal y el procesador. Contiene una copia de los datos y las instrucciones que suele utilizar el procesador a continuación; es, por tanto, más rápido que la memoria principal. Véase también la definición de “caché de disco”.
CASE <i>CASE</i>	Véase la definición de “ingeniería del software asistida por computador”.
conjunto de caracteres <i>character set</i>	Conjunto completo de caracteres diferentes y que tiene un objetivo determinado; por ejemplo, los 128 caracteres ASCII.
dígito de verificación <i>check digit</i>	Dígito que se añade a un dato numérico que se puede volver a calcular y, por tanto, utilizar para verificar la integridad de los datos después de que se haya realizado alguna entrada, transmisión, etc.
suma de verificación <i>check sum</i>	Suma generada mediante dígitos individuales de un número y que se utiliza como dispositivo para la detección de errores.
cola circular <i>circular queue</i>	<sup>NS</sup> Cola en la que área de almacenamiento está fijada y el primer elemento se guarda en una ubicación lógicamente próxima a la ubicación de almacenamiento del último elemento de la cola. Se puede considerar que los elementos de datos se ordenan de manera circular.
conflicto (colisión) <i>clash (collision)</i>	<sup>NS</sup> Situación en la cual se da a dos o más entradas de un archivo u otra estructura de datos la misma ubicación de memoria mediante el uso de una tabla hash.
clase <i>class</i>	Combinación de datos y operaciones que se pueden realizar sobre éstos; especificación de los miembros datos y de los métodos del objeto.
cliente <i>client</i>	Computador o terminal que se utiliza para acceder a un sistema basado en computadores.
cliente-servidor <i>client-server</i>	Arquitectura de red en la que un sistema se divide entre las tareas de servidor que se realizan bajo las instrucciones recibidas de los clientes, que solicitan información.
colección <i>collection</i>	Clase diseñada para contener objetos (en el programa de estudios se denomina “estructura de datos”).
lenguaje de órdenes <i>command language</i>	<sup>NS</sup> Conjunto de operadores procedimentales que poseen una sintaxis relacionada, utilizada para indicar las funciones que ha de realizar un sistema operativo.

compilador <i>compiler</i>	Programa que traduce un programa fuente en código máquina que se puede convertir, a su vez, en un programa ejecutable (programa objeto). Véase también la definición de “intérprete”.
ingeniería del software asistida por computador <i>computer-assisted software engineering</i>	Automatización de metodologías bien definidas que se utilizan para el desarrollo y mantenimiento de productos. Estas metodologías se aplican a prácticamente todos los procesos o las actividades del ciclo de desarrollo de un producto como, por ejemplo, la planificación de un proyecto, el diseño de productos, la codificación y las pruebas.
arquitectura de computadores <i>computer architecture</i>	Estructura lógica y características funcionales de un computador, incluidas las relaciones entre sus componentes de hardware y software.
programa de computador <i>computer program</i>	Secuencia de instrucciones adecuadas para que las procese un computador.
método constructor <i>constructor method</i>	Método que tiene el mismo nombre que la clase e inicializa las variables instancia de un objeto de la clase cuando se instancia dicho objeto.
tarjetas CRC <i>CRC cards</i>	Tarjetas de clase, responsabilidad y colaboración. Herramienta de diseño de clases que enumera el nombre de una clase, sus responsabilidades y las clases con las que colabora en una tarjeta índice.
cilindro <i>cylinder</i>	<sup>NS</sup> Pistas concéntricas de un disco duro (superpuestas) que forman un cilindro.
sistema de gestión de bases de datos (SGBD) <i>database management system (DBMS)</i>	Sistema informático para definir, crear, manipular, controlar, gestionar y utilizar bases de datos.
bus de datos <i>data bus</i>	Vía de comunicación entre la memoria o los periféricos y la unidad de procesamiento, que transporta datos que se van a procesar o que se han procesado. Véase también las definiciones de “bus” y “bus de direcciones”.
compresión de datos <i>data compression</i>	Método para reducir el tamaño de los datos. Se eliminan todas las redundancias de los datos para disminuir el almacenamiento necesario o para aumentar la velocidad de transferencia. Los datos se pueden volver a descomprimir a su estado original.
integridad de los datos <i>data integrity</i>	Exactitud de los datos después de su procesamiento, almacenamiento o transmisión.
miembro dato <i>data member</i>	Tipo de dato que es miembro de una clase.

<p>paquete de datos <i>data packet</i></p>	<p>Parte de un mensaje transmitido que se envía por separado. Además de contener una parte del mensaje, tiene otros datos como, por ejemplo, dígitos de verificación, direcciones de destino, etc.</p>
<p>protección de datos <i>data protection</i></p>	<p>Método para asegurar que los datos personales son correctos, que no se utilizan de forma inadecuada y que solamente tienen acceso a ellos quienes tienen autorización.</p>
<p>seguridad en los datos <i>data security</i></p>	<p>Método para asegurar que los datos son correctos, seguros y que no pueden ser leídos o modificados por aquellos que no tienen acceso a los mismos.</p>
<p>SGBD <i>DBMS</i></p>	<p>Véase la definición de “sistema de gestión de bases de datos”.</p>
<p>herramienta de depuración <i>debugging tool</i></p>	<p>Programa utilizado para detectar, rastrear y eliminar errores en programas informáticos u otro software.</p>
<p>software de desfragmentación <i>defragmentation software</i></p>	<p>Aplicación que lee segmentos de archivos de secciones no contiguas de un dispositivo de almacenamiento y posteriormente escribe los archivos en el mismo dispositivo de tal forma que cada segmento de archivo sea contiguo.</p>
<p>ley de De Morgan <i>De Morgan's law</i></p>	<p><sup>NS</sup> Sean A y B expresiones booleanas, entonces</p> $\overline{A + B} = \overline{A} \cdot \overline{B}$ $\overline{A \cdot B} = \overline{A} + \overline{B}$
<p>quitar de la cola <i>dequeue</i></p>	<p><sup>NS</sup> Eliminar el primer elemento de una cola. Véase también la definición de “añadir a la cola”.</p>
<p>datos digitales <i>digital data</i></p>	<p>Datos discretos.</p>
<p>firma digital <i>digital signature</i></p>	<p>Código digital anexo a un mensaje o documento electrónico; es único y se puede utilizar para autenticar al emisor o al propietario. Se suele utilizar en el comercio electrónico.</p>
<p>archivo de acceso directo <i>direct access file</i></p>	<p>Archivo organizado de tal forma que un cálculo proporciona la dirección (ubicación) de un registro, de manera que se pueda acceder directamente a dicho registro. Los registros del archivo pueden estar ordenados o sin ordenar.</p>
<p>DMA (acceso directo a memoria) <i>DMA</i></p>	<p><sup>NS</sup> Acceso a memoria y dispositivos sin que exista el control directo del procesador. A menudo se utiliza para accesos al disco duro y la pantalla.</p>

caché de disco <i>disk cache</i>		RAM reservada para acelerar el acceso a un disco duro. Puede ser parte del propio disco o estar incorporada en la memoria caché.
procesamiento distribuido <i>distributed processing</i>		Red en la que todas o algunas de las funciones de procesamiento, almacenamiento y control, además de las funciones de entrada/salida, se dispersan entre sus nodos.
búfering doble <i>double buffering</i>	NS	Dos áreas de memoria reservadas para la transferencia de datos entre el procesador y los periféricos. A medida que una se vacía, la otra se llena para acelerar la transferencia.
lista doblemente enlazada <i>doubly linked list</i>	NS	Lista enlazada en la que cada nodo tiene un puntero de cabeza y otro de cola.
estructura de datos dinámica <i>dynamic data structure</i>	NS	Estructura de datos que puede cambiar de tamaño durante la ejecución de un programa. Véase también la definición de “estructuras de datos estáticas”.
encapsulación <i>encapsulation</i>	NS	Combinación de datos y las operaciones que actúan sobre dichos datos en una unidad de programa simple denominada “objeto”.
encriptación <i>encryption</i>		En seguridad informática, proceso de transformación de datos en una forma ininteligible, de forma que no sea posible conocer dichos datos a menos que se utilice un proceso de desencriptación.
añadir a la cola <i>enqueue</i>	NS	Insertar un elemento al final de una cola. Véase también la definición de “quitar de la cola”.
excepción <i>exception</i>		Objeto que se crea cuando se produce una situación anómala en un programa. Véase también la definición de “manipulador de excepciones”.
manipulador de excepciones <i>exception handler</i>		Código de programa que trata las excepciones que se producen durante la ejecución de un programa. En lugar de ocasionar un error grave, se lanza una excepción al manipulador. Véase también la definición de “excepción”.
expresión <i>expression</i>		Secuencia de símbolos que se pueden evaluar.
fibra óptica <i>fibre optic</i>		Cableado que utiliza hebras finas de vidrio y se usa en redes. El medio puede transportar una gran cantidad de datos y proporciona una elevada tasa de transferencia.

<p>campo (atributo de objeto) <i>field (object attribute)</i></p>	<p>Subdivisión de un registro que contiene una unidad de información. Por ejemplo, un registro en una nómina de empleados puede contener los siguientes campos: número de empleado, salario bruto, deducciones y salario neto.</p>
<p>FIFO <i>FIFO</i></p>	<p><sup>NS</sup> El primero en entrar es el primero en salir. Véase también las definiciones de “cola”, “pila” y “LIFO”.</p>
<p>archivo <i>file</i></p>	<p>Colección organizada de datos.</p>
<p>gestor de archivos <i>file manager</i></p>	<p>Software de aplicación que puede acceder a archivos, crearlos, modificarlos, almacenarlos y recuperarlos.</p>
<p>registro de longitud fija <i>fixed-length records</i></p>	<p><sup>NS</sup> Registro cuyo tamaño está determinado previamente. En un archivo, todos los registros de este tipo tienen la misma longitud. Véase también la definición de “registro de longitud variable”.</p>
<p>punto fijo <i>fixed point</i></p>	<p><sup>NS</sup> Realización de cálculos aritméticos independientemente de la posición del punto decimal. Es necesario controlar la posición relativa del punto durante los cálculos.</p>
<p>indicador <i>flag</i></p>	<p>Señalizador que puede tener dos estados, “activado” o “desactivado”, los cuales se pueden representar por un bit. Los indicadores se pueden utilizar para especificar que se puede eliminar un registro, para señalar el final de la entrada/salida y para detectar si se ha producido una interrupción.</p>
<p>punto flotante <i>floating point</i></p>	<p><sup>NS</sup> En aritmética de punto flotante, la posición del punto decimal no depende de la posición relativa de los dígitos de los números (como ocurre en la aritmética de punto fijo), ya que las dos partes del número en punto flotante determinan el valor absoluto del número.</p>
<p>parámetro formal <i>formal parameter</i></p>	<p>Véase también la definición de “parámetro”.</p>
<p>salida formateada <i>formatted output</i></p>	<p>Datos preparados para su salida de forma que se muestren en el formato deseado (por ejemplo, los ceros a la derecha en el decimal 7,50\$ en lugar de 7,5\$).</p>
<p>archivo completamente indexado <i>fully-indexed file</i></p>	<p><sup>NS</sup> Archivo en el que, aunque todos los registros estén sin ordenar, se puede encontrar un registro concreto mediante un acceso secuencial al índice del archivo, seguido de un acceso directo al archivo de datos. Véase también la definición de “archivo parcialmente indexado”.</p>

pasarela <i>gateway</i>	<sup>NS</sup> Enlace entre dos sistemas informáticos que toma los datos que se le transfieren y los convierte a los formatos necesarios para cada sistema.
tableta (almohadilla) digitalizadora <i>graphics tablet</i> ( <i>graphics pad</i> )	Dispositivo de entrada en el que el usuario puede escribir o dibujar. La imagen se reproduce en el monitor.
GUI <i>GUI</i>	Interfaz gráfica de usuario.
hacking <i>hacking</i>	Obtención de acceso no autorizado a recursos protegidos.
protocolo de intercambio <i>handshaking</i>	<sup>NS</sup> Intercambio de señales predeterminadas durante el establecimiento de una conexión entre dos dispositivos o componentes.
código hash <i>hash code</i>	<sup>NS</sup> Método de codificación para obtener una clave de búsqueda con el objetivo de almacenar y recuperar elementos de datos.
tabla hash <i>hash table</i>	<sup>NS</sup> Tabla de información a la que se accede por medio de una clave de búsqueda acertada (el valor hash).
hexadecimal <i>hexadecimal</i>	Sistema numérico cuya base es 16; los dígitos hexadecimales tienen un rango comprendido entre 0 y 9, y entre A y F, donde A representa 10, y F representa 15.
lenguaje de alto nivel <i>high-level language</i>	Lenguaje de programación cuyos conceptos y estructuras son adecuados para el razonamiento humano. Estos lenguajes son independientes de las estructuras de los computadores y de los sistemas operativos.
HTML (lenguaje de marcas de hipertexto) <i>HTML (Hyper Text Markup Language)</i>	Lenguaje informático utilizado para crear páginas web. Para denotar la forma en que se van a mostrar el texto y los gráficos se utilizan etiquetas. El lenguaje lo interpreta un navegador, que es el encargado de mostrar las páginas.
hub <i>hub</i>	En redes de computadores, conmutador que envía datos a las estaciones a las que está conectado.
IDE (entorno de desarrollo integrado) <i>IDE (integrated development environment)</i>	Herramienta de programación que ofrece a los programadores un entorno simple (es decir, entorno hardware y software en el que se ejecutan los programas) para la construcción de programas, en contraposición al uso de editores y depuradores individuales.

<p>identificador <i>identifier</i></p>	<p>Nombre o etiqueta que elige el programador para representar una variable, un método, una clase, un tipo de dato o cualquier otro elemento definido dentro del programa.</p>
<p>notación infija <i>infix notation</i></p>	<p><sup>NS</sup> Notación para representar operadores lógicos en la que el operador se escribe entre los operandos, por ejemplo, A+B o A*B. Véase también las definiciones de “notación postfija” y “notación prefija”.</p>
<p>herencia <i>inheritance</i></p>	<p><sup>NS</sup> Nombre aplicado a la propiedad por la cual un objeto que deriva de otro objeto hereda los miembros dato y las funciones miembro del objeto original.</p>
<p>recorrido en orden <i>in-order traversal</i></p>	<p><sup>NS</sup> Exploración de un árbol pasando por todos los nodos en el orden hijo-izquierdo, padre, hijo-derecho. Véase también las definiciones de “recorrido en orden previo” y “recorrido en orden posterior”.</p>
<p>ordenación por inserción <i>insertion sort</i></p>	<p><sup>NS</sup> Ordenación en la que cada elemento de un conjunto se inserta en la posición que le corresponde en función de un criterio específico. Véase también las definiciones de “ordenación por el método de la burbuja”, “ordenación por selección” y “ordenación rápida”.</p>
<p>interfaz <i>interface</i></p>	<p>Hardware y software asociados necesarios en la comunicación entre procesadores y dispositivos periféricos para compensar las diferencias de sus características de funcionamiento.</p>
<p>intérprete <i>interpreter</i></p>	<p>Programa que traduce y ejecuta cada instrucción de un lenguaje de programación antes de traducir y ejecutar la instrucción siguiente. Véase también la definición de “compilador”.</p>
<p>interrupción <i>interrupt</i></p>	<p><sup>NS</sup> Suspensión de un proceso como, por ejemplo, la ejecución de un programa informático causada por un evento externo; dicha suspensión se realiza de tal forma que el proceso pueda reanudarse.</p>
<p>RDSI (red digital de servicios integrados) <i>ISDN (integrated services digital network)</i></p>	<p><sup>NS</sup> Estándar internacional de comunicaciones para el envío de voz, video y otros datos a través de líneas telefónicas digitales.</p>
<p>ISO <i>ISO</i></p>	<p>Organización Internacional de Normalización.</p>
<p>iteración <i>iteration</i></p>	<p>Proceso que consiste en ejecutar repetidamente un conjunto de instrucciones en el computador hasta que se satisfaga alguna condición.</p>

JPEG (joint photographic expert group) <i>JPEG (joint photographic expert group)</i>		Estándar reconocido de compresión de archivos gráficos cuyas pérdidas son mínimas.
clave <i>keys</i>	NS	<ol style="list-style-type: none"> <li>1. En seguridad informática, secuencia de símbolos utilizados con un algoritmo criptográfico para encriptar o desencriptar datos.</li> <li>2. En bases de datos, la clave de un registro es un campo con un valor único que se puede utilizar para localizar dicho registro.</li> </ol>
latencia <i>latency</i>		Véase la definición de “retardo rotacional”.
hijo-izquierdo <i>left-child</i>	NS	En un árbol, nodo situado inmediatamente a la izquierda de un nodo padre. Véase también las definiciones de “padre” e “hijo-derecho”.
gestor de bibliotecas <i>library manager</i>	NS	Muchos lenguajes de programación permiten almacenar centralmente funciones definidas por el usuario y reutilizarlas en varios programas. Este almacén central se denomina “biblioteca”. Un gestor de bibliotecas es un programa de utilidad que cataloga, precompila y enlaza módulos de biblioteca.
LIFO <i>LIFO</i>	NS	El último en entrar es el primero en salir. Véase también las definiciones de “pila”, “cola” y “FIFO”.
lista enlazada <i>linked list</i>	NS	Técnica de estructura de datos para el almacenamiento de datos en diferentes áreas de memoria (no en un bloque contiguo), y para el seguimiento de los datos mediante punteros.
enlazador <i>linker</i>	NS	Programa de utilidad que reúne los módulos objeto, las rutinas del sistema operativo y otro software de utilidad para producir un programa completo y ejecutable.
cargador <i>loader</i>	NS	Programa que copia un programa objeto almacenado en memoria en el área de memoria para ejecución que designe el sistema operativo.
red de área local (LAN) <i>local area network</i> (LAN)		Red informática en la que todos los computadores están unidos directamente por cables y/o transmisión de microondas. Normalmente se ubica en el local de un usuario dentro de un área geográfica limitada. Véase también la definición de “red de área ancha (WAN)”.
variable local <i>local variable</i>		Variable que se define y que se puede utilizar sólo en un bloque específico de programa.

<p>circuito lógico <i>logic circuit</i></p>	<p><sup>NS</sup> Circuito cuya salida se puede determinar a partir de las entradas y mediante un seguimiento de la ruta a través de las puertas lógicas.</p>
<p>error lógico <i>logic error</i></p>	<p>Fallo que surge tras una apreciación incorrecta del problema, lo que conduce a una acción incorrecta y, por tanto, a un resultado falso.</p>
<p>puerta lógica <i>logic gate</i></p>	<p><sup>NS</sup> Circuito combinacional que realiza una operación lógica elemental y, normalmente, tiene una única salida.</p>
<p>reconocimiento de caracteres de tinta magnética (MICR) <i>magnetic ink character recognition (MICR)</i></p>	<p>Identificación de caracteres por medio de tinta magnética. Véase también la definición de “OCR”.</p>
<p>computador central <i>mainframe</i></p>	<p>Computador, normalmente utilizado en un centro de computación, que posee amplias capacidades y recursos y al que se pueden conectar otros computadores para compartir funciones.</p>
<p>archivo maestro <i>master file</i></p>	<p>Archivo permanente que contiene información a la que se puede acceder y que se actualiza periódicamente mediante procesamiento con un archivo de transacción. Véase también la definición de “archivo de transacción”.</p>
<p>registro de dirección de memoria (MAR) <i>memory address register (MAR)</i></p>	<p>Registro que almacena en memoria la dirección de la instrucción que se esté ejecutando en ese momento.</p>
<p>gestor de memoria <i>memory manager</i></p>	<p><sup>NS</sup> Programa que normalmente forma parte del sistema operativo y controla la asignación de memoria destinada a varias aplicaciones. Resulta especialmente importante en sistemas multitarea, en los que las aplicaciones pueden ocasionar conflictos, así como para la implementación de máquinas y memorias virtuales.</p>
<p>E/S mapeada por memoria <i>memory mapped I/O</i></p>	<p>Véase también la definición de “DMA”.</p>
<p>menú <i>menu</i></p>	<p>Visualización de una lista de funciones opcionales que el usuario puede elegir para llevar a cabo diferentes tareas en un sistema.</p>
<p>método <i>method</i></p>	<ol style="list-style-type: none"> <li>1. Comportamiento o funcionamiento de un objeto.</li> <li>2. Procedimiento que utiliza un objeto, tal como se especifica dentro de la clase de dicho objeto. Véase también la definición de “firma de un método”.</li> </ol>

<p>firma de un método <i>method signature</i></p>	<p>Número y tipo de argumentos de un método.</p>
<p>MICR <i>MICR</i></p>	<p>Véase la definición de “reconocimiento de caracteres de tinta magnética”.</p>
<p>microprocesador <i>microprocessor</i></p>	<p>Circuito integrado que incorpora los componentes principales de un procesador central. Estos circuitos se utilizan en microcomputadores y pequeños dispositivos controlados por computador.</p>
<p>transmisión de microondas <i>microwave transmission</i></p>	<p>Método de comunicación electrónica que no requiere el uso de cables.</p>
<p>módem <i>modem</i></p>	<p>Forma abreviada de “modulador/demodulador”: Parte de un equipo electrónico que convierte señales digitales de un computador en señales de audio que se transmiten a través de las líneas telefónicas, y viceversa.</p>
<p>lenguaje modular <i>modular language</i></p>	<p>Lenguaje en el que un programa completo se puede dividir en componentes separados (módulos), siendo cada uno, en cierto modo, independiente. Por ejemplo, el alcance de las variables puede estar limitado a un módulo y no extenderse al programa completo. Véase también la definición de “diseño descendente”.</p>
<p>modularidad <i>modularity</i></p>	<p>Aspecto de la programación estructurada en el que las tareas individuales se programan como módulos o secciones diferentes. Una ventaja es la facilidad con la que se pueden modificar las secciones individuales sin hacer referencia a otras secciones.</p>
<p>módulo <i>module</i></p>	<p>Subconjunto independiente de un programa.</p>
<p>aritmética de módulo <i>modulo arithmetic</i></p>	<p>Aritmética que utiliza el resultado entero y el resto entero de una división como dos entidades separadas.</p>
<p>multitarea <i>multi-tasking</i></p>	<p>Modo de funcionamiento que proporciona rendimiento concurrente o la ejecución intercalada de dos o más tareas.</p>
<p>sistema multiusuario <i>multi-user system</i></p>	<p>Sistema que permite que dos o más personas utilicen los servicios de un procesador dentro de un período de tiempo determinado.</p>
<p>multiprocesamiento <i>multi-processing</i></p>	<p>Ejecución simultánea de dos o más programas informáticos o secuencias de instrucciones por parte de un computador (procesamiento paralelo).</p>

<b>nand</b> <i>nand</i>	<sup>NS</sup> La salida de “nand” es falsa sólo si todas las entradas son verdaderas, de lo contrario, la salida es verdadera.
red <i>network</i>	Cualquier conjunto de sistemas informáticos interconectados que comparten recursos y datos. Véase también las definiciones de “interconexión”, “red de área local (LAN)” y “red de área ancha (WAN)”.
interconexión <i>networking</i>	Utilización de los servicios de una red. Véase también las definiciones de “red”, “red de área local (LAN)” y “red de área ancha (WAN)”.
nodo <i>node</i>	<ol style="list-style-type: none"> <li>1. En terminología de estructuras de árboles, cada una de las posiciones en un árbol.</li> <li>2. Cualquier dispositivo de una red de computadores al que se puede asignar una dirección, de forma que cualquier otro computador pueda acceder a él.</li> <li>3. Computador “host” de una red.</li> </ol>
<b>nor</b> <i>nor</i>	<sup>NS</sup> La salida de “nor” es verdadera si todas las sentencias son falsas, y falsa si al menos una sentencia es verdadera.
<b>not</b> <i>not</i>	La salida de “not” es verdadera si la sentencia es falsa, y falsa si la sentencia es verdadera.
objeto <i>object</i>	Un objeto es una combinación de datos y las operaciones que se pueden realizar en asociación con dichos datos. A cada parte de datos de un objeto se la conoce con el nombre de “miembro dato”, mientras que es posible denominar “métodos” a las operaciones. El estado actual de un objeto se almacena en sus miembros dato; sólo los métodos pueden modificar o acceder a dicho estado. Entre las categorías de operaciones más comunes se incluyen: construcción de objetos, operaciones que establecen (métodos mutadores) o devuelven (métodos accesorios) los miembros dato; operaciones únicas para los tipos de datos y operaciones que utiliza internamente el objeto.
programación orientada a objetos ( <i>OOP</i> ) <i>object-oriented</i> <i>programming (OOP)</i>	Aproximación a la programación en la que las unidades de datos se ven como “objetos” activos, no como unidades pasivas, tal como se conciben en el paradigma procedimental.
OCR <i>OCR</i>	Reconocimiento (lector) óptico de caracteres. Se refiere al uso de dispositivos y el software utilizado para “leer” caracteres y traducirlos a código ASCII para su posterior procesamiento. Entre las aplicaciones de OCR se incluyen el uso de escáner en documentos impresos para convertir texto a formato digital ASCII para que se puedan modificar en procesadores de texto.

formularios para OMR <i>OMR forms</i>	Formularios para lectores ópticos de marcas.
en línea <i>on-line</i>	Situación en la que un usuario tiene acceso a un computador a través de un terminal.
procesamiento en línea (interactivo) <i>on-line processing (interactive)</i>	Procesamiento de datos en el que todas las operaciones se realizan directamente bajo el control de un procesador central; por ejemplo, las reservas de pasajes de avión.
interconexión de sistemas abiertos (OSI) <i>open systems interconnection (OSI)</i>	<sup>NS</sup> Conjunto de protocolos que permiten que se conecten entre sí varios tipos de computadores diferentes.
operando <i>operand</i>	<sup>NS</sup> En una expresión aritmética, el operando es el dato sobre el que se va a operar.
sistema operativo (OS) <i>operating system (OS)</i>	Software que controla la ejecución de programas y que puede proporcionar servicios como la asignación de recursos, la planificación, el control de entrada/salida y la gestión de datos.
operador <i>operator</i>	<sup>NS</sup> Carácter o cadena de caracteres que designan una operación. Véase también las definiciones de “operador binario” y “operador unario”.
precedencia de operadores <i>operator precedence</i>	En los lenguajes de programación, relación que define el orden en que se aplican los operadores dentro de una expresión.
<b>or</b> <i>or</i>	La salida de “or” es falsa si al menos una entrada es verdadera, de lo contrario la salida es falsa.
desbordamiento <i>overflow</i>	<sup>NS</sup> Generación de una cantidad, como resultado de una operación aritmética, que es demasiado grande para estar contenida en la ubicación del resultado. Véase también la definición de “subdesbordamiento”.
paquete <i>packet</i>	<sup>NS</sup> Grupo de bits formado por señales de control, bits de control de errores, información codificada y el destino de los datos.
conmutación de paquetes <i>packet switching</i>	<sup>NS</sup> Método de transmisión de datos en el cual los paquetes de datos se transmiten como una entidad, sin tener en cuenta el mensaje completo.

interfaz paralela <i>parallel interface</i>	NS Interfaz a través de la cual un computador transmite o recibe datos que se componen de varios bits enviados simultáneamente por cables separados. Véase también la definición de “interfaz serie”.
parámetro <i>parameter</i>	NS Los parámetros se pasan a una rutina o a un método mediante nombre y tipo de variable. Cuando se ejecuta el código, el parámetro se sustituye por el valor de la variable y se convierte en el argumento de la rutina, a la que hace referencia el nombre de la variable en la definición.
paso de parámetros <i>parameter passing</i>	Asignación de valores a los parámetros que se van a utilizar en un procedimiento.
padre (nodo) <i>parent (node)</i>	NS Nodo inmediatamente encima de otro. Los nodos sólo pueden tener un nodo padre, y diferentes nodos pueden compartir un mismo nodo padre.
bit de paridad <i>parity bit</i>	Dígito binario que se añade a un grupo de dígitos binarios para realizar la suma de todos los dígitos, incluido el dígito binario anexo, impar o par según lo preestablecido.
análisis sintáctico <i>parsing</i>	NS Descomposición de las sentencias de un lenguaje de programación de alto nivel en sus partes componentes durante el proceso de traducción. Un ejemplo sería la identificación de variables y palabras reservadas.
archivo parcialmente indexado <i>partially-indexed file</i>	NS Archivo en el que los registros están ordenados en grupos. Acceso secuencial a un índice seguido de un acceso directo al primer registro del grupo; posteriormente, el acceso secuencial al registro deseado recupera un registro completo. Véase también la definición de “archivo completamente indexado”.
paso por referencia <i>pass-by-reference</i>	Mecanismo para el paso de parámetros por el cual se pasa la dirección de una variable al subprograma invocado. Si el subprograma modifica el parámetro formal, también se modifica el parámetro real correspondiente. En Java todos los objetos, incluidas las matrices, se pasan por referencia. Véase también la definición de “paso por valor”.
paso por valor <i>pass-by-value</i>	Mecanismo para el paso de parámetros por el que se pasa una copia del valor del parámetro real al procedimiento invocado. Si el procedimiento invocado modifica, el parámetro formal el parámetro real correspondiente no se ve afectado. En Java, todos los primitivos se pasan por valor. Véase también la definición de “paso por referencia”.
dispositivo periférico <i>peripheral device</i>	Cualquier dispositivo que pueda comunicarse con un computador concreto como, por ejemplo, unidades de entrada/salida, almacenamiento auxiliar e impresoras.

puntero <i>pointer</i>	<sup>NS</sup> Referencia a una dirección que permite la recuperación de un registro o elemento de datos. Se utiliza en estructuras de datos dinámicas para desplazarse por los elementos.
dispositivo apuntador <i>pointing device</i>	Instrumento como, por ejemplo, un ratón, trackball o joystick, utilizado para mover un icono (a veces con forma de flecha) en pantalla.
sondeo <i>polling</i>	<sup>NS</sup> Interrogación a dispositivos, con el objetivo de evitar la contención y determinar el estatus de funcionamiento o la disponibilidad para enviar o recibir datos.
polimorfismo <i>polymorphism</i>	<sup>NS</sup> Capacidad que tienen diferentes objetos para responder de la forma adecuada a la misma operación.
sacar <i>pop</i>	<sup>NS</sup> Eliminar un elemento de la parte superior de la pila.
puerto <i>port</i>	<sup>NS</sup> Punto de acceso para la entrada o salida de datos.
notación postfija <i>postfix notation</i>	<sup>NS</sup> Método para formar expresiones matemáticas en el que cada operador está precedido por sus operandos e indica la operación que se va a realizar sobre dichos operandos o los resultados inmediatos que lo preceden; por ejemplo, la multiplicación del resultado de A más B por C se representa mediante la expresión $AB+C*$ . Véase también las definiciones de “notación infija” y “notación prefija”.
recorrido en orden posterior <i>post-order traversal</i>	<sup>NS</sup> Exploración de un árbol, pasando por todos los nodos de forma recursiva en el orden hijo-izquierdo, hijo-derecho, padre. Véase también las definiciones de “recorrido en orden previo” y “recorrido en orden”.
notación prefija <i>prefix notation</i>	<sup>NS</sup> Método para formar expresiones matemáticas en el que cada operador precede a sus operandos e indica la operación que se va a realizar sobre los operandos o el resultado intermedio que lo sigue. Véase también las definiciones de “notación infija” y “notación postfija”.
recorrido en orden previo <i>pre-order traversal</i>	<sup>NS</sup> Exploración de un árbol, pasando por todos los nodos de forma recursiva en el orden padre, hijo-izquierdo, hijo-derecho. Véase también las definiciones de “recorrido en orden” y “recorrido en orden posterior”.
memoria principal <i>primary memory</i>	Parte de la memoria en la que se almacenan los datos y programas que se están utilizando en ese momento.

<p>tipo de dato primitivo <i>primitive data type</i></p>	<p>Tipos de datos entero, real, carácter o booleano.</p>
<p>miembros de clase privados <i>private class members</i></p>	<p>Miembros de una clase a los que sólo se puede acceder desde los métodos que pertenecen a la clase.</p>
<p>contador de programa <i>program counter</i></p>	<p><sup>NS</sup> Registro que almacena la dirección de la siguiente instrucción que se va a seleccionar en el ciclo de ejecución de búsqueda.</p>
<p>protocolo <i>protocol</i></p>	<p>Conjunto de reglas consensuado internacionalmente para garantizar la transferencia de datos entre dispositivos. Un protocolo estándar es aquél que está reconocido como habitual para un tipo concreto de transferencia. Por ejemplo, TCP/IP.</p>
<p>creación de prototipos <i>prototyping</i></p>	<p>Construcción de una versión simple de un sistema durante la fase de diseño, en la que se muestra la interfaz de usuario sin todo el procesamiento subyacente. Esto permite al usuario proponer cambios en la fase de diseño.</p>
<p>pseudocódigo <i>pseudocode</i></p>	<p>Lenguaje artificial utilizado para describir algoritmos de programas informáticos que no utiliza la sintaxis de ningún lenguaje concreto. Durante el desarrollo de un algoritmo, el pseudocódigo suele contener secciones en lenguaje natural que se sustituirán posteriormente.</p>
<p>miembros de clase públicos <i>public class members</i></p>	<p>Miembros de una clase a los que se puede acceder desde cualquier ubicación y cualquier clase.</p>
<p>meter <i>push</i></p>	<p><sup>NS</sup> Insertar un elemento en la parte superior de una pila.</p>
<p>cola <i>queue</i></p>	<p><sup>NS</sup> Estructura de datos abstracta en la que los elementos se insertan en un extremo y se recuperan desde el otro extremo (FIFO). (Las operaciones estándares se exponen en 5.2.7).</p>
<p>ordenación rápida <i>quicksort</i></p>	<p><sup>NS</sup> Ordenación en la que una lista se particiona, en primer lugar, en una sublista inferior y otra superior, para las cuales todas las claves son, respectivamente, menores que alguna clave pivote o mayores que la clave pivote. Véase también las definiciones de “ordenación por el método de la burbuja”, “ordenación por selección” y “ordenación por inserción”.</p>
<p>procesamiento en tiempo real <i>real-time processing</i></p>	<p>Manipulación de datos requeridos o generados mediante algún procedimiento mientras que el proceso se está ejecutando; generalmente, los resultados se utilizan para influir en el proceso y, quizá, en procesos relacionados mientras que se produce esta manipulación.</p>

registro <i>record</i>		Conjunto formado por objetos de datos, generalmente con atributos diferentes, que suelen tener identificadores asociados. Véase también la definición de “campo”.
recursividad <i>recursion</i>	NS	Proceso por el cual un método hace referencia a sí mismo. En muchos lenguajes de programación, los procedimientos y funciones se pueden invocar a sí mismos.
referencia <i>reference</i>	NS	Contiene la ubicación en memoria de un objeto. El objeto puede contener varios miembros dato individuales.
registro <i>register</i>	NS	Parte del almacenamiento interno que posee una capacidad específica de almacenamiento y se utiliza generalmente con un objetivo concreto.
especificación de requisitos <i>requirements specification</i>		Documento que expone los requisitos del usuario de un sistema informático. Se escribe como parte del análisis de sistemas y se puede utilizar posteriormente para evaluar el sistema después de su implementación.
hijo-derecho <i>right-child</i>	NS	En un árbol, nodo situado inmediatamente a la derecha de un nodo padre. Véase también las definiciones de “padre” e “hijo-izquierdo”.
robótica <i>robotics</i>		Técnicas utilizadas en el diseño, la construcción y la utilización de robots.
robustez <i>robustness</i>		Término utilizado para describir la capacidad que tiene un programa de resistir colapsos debidos a entradas o resultados intermedios incorrectos.
retardo rotacional <i>rotational delay</i>	NS	En una unidad de disco, tiempo necesario para que el disco gire hasta que el sector correcto se encuentre por encima o por debajo de los cabezales de lectura/escritura. Véase también la definición de “tiempo de búsqueda”.
encaminador <i>router</i>		Dispositivo que identifica el destino de los mensajes y los envía a través del camino adecuado.
motor de búsqueda <i>search engine</i>		Programa que busca en una gran base de datos para encontrar elementos coincidentes. El uso más común de un motor de búsqueda es buscar direcciones de Internet en función de las palabras clave proporcionadas.
memoria secundaria <i>secondary memory</i>		Tipo de memoria que permite a un usuario almacenar datos y programas durante el tiempo que desee; por ejemplo, un disco duro.

<p>sector <i>sector</i></p>	<p><sup>NS</sup> Menor unidad de almacenamiento a la que se puede acceder en un disco. El punto en el que el sector se interseca con una pista se utiliza para hacer referencia a la ubicación.</p>
<p>seguridad <i>security</i></p>	<p>La seguridad, en un contexto informático, es un tema amplio; sin embargo, a grandes rasgos puede referirse a:</p> <ol style="list-style-type: none"> <li>1. riesgo para el hardware</li> <li>2. riesgo para el software</li> <li>3. riesgo para la información.</li> </ol>
<p>tiempo de búsqueda <i>seek time</i></p>	<p><sup>NS</sup> En una unidad de disco, tiempo necesario para que los cabezales de lectura/escritura se posicionen sobre la pista adecuada. Véase también la definición de “retardo rotacional”.</p>
<p>ordenación por selección <i>selection sort</i></p>	<p>Búsqueda en la cual los elementos de un conjunto se examinan con el fin de encontrar uno que cumpla unos criterios especificados. Dicho elemento se añade al conjunto ordenado y ya no se tiene en cuenta; el proceso se repite hasta que todos los elementos estén en el conjunto ordenado. Véase también las definiciones de “ordenación por el método de la burbuja”, “ordenación por inserción” y “ordenación rápida”.</p>
<p>semántica <i>semantics</i></p>	<p>Relaciones de caracteres o grupos de caracteres con sus significados, independientemente de la forma en que se interpreten y utilicen.</p>
<p>sensor <i>sensor</i></p>	<p>Dispositivo que detecta elementos medibles de un proceso físico para su transferencia a un computador.</p>
<p>centinela <i>sentinel</i></p>	<p><sup>NS</sup> Valor especial que indica el final de un conjunto de datos.</p>
<p>acceso secuencial <i>sequential access</i></p>	<p>Método de acceso en el que los registros se leen, escriben o se eliminan desde un archivo en función del orden lógico de los registros de dicho archivo.</p>
<p>archivo secuencial <i>sequential file</i></p>	<p>Archivo en el que los registros están ordenados y se recuperan mediante acceso secuencial.</p>
<p>búsqueda secuencial <i>sequential search</i></p>	<p>Búsqueda en la que los registros de un archivo o de otra estructura de datos se examinan uno por uno en el orden en que se introdujeron, hasta que se cumpla un criterio especificado o hasta que no haya más registros que examinar. Véase también la definición de “búsqueda binaria”.</p>
<p>interfaz serie <i>serial interface</i></p>	<p><sup>NS</sup> Interfaz a través de la cual un computador transmite o recibe datos, un bit cada vez. Véase también la definición de “interfaz paralela”.</p>

servidor <i>server</i>	<ol style="list-style-type: none"> <li>1. Programa que proporciona servicios que solicitan los programas clientes.</li> <li>2. Computador que proporciona servicios a otro computador que se encuentra conectado a una red.</li> </ol>
firma <i>signature</i>	Combinación de especificadores, nombre del método y lista de parámetros que identifica al método de manera única.
simulación <i>simulation</i>	Utilización de un sistema de procesamiento de datos para representar características seleccionadas del comportamiento de un sistema físico o abstracto.
monotarea <i>single-tasking</i>	Modo de funcionamiento que permite que sólo un programa esté en uso en un momento determinado.
sistema monousuario <i>single-user system</i>	Sistema que no permite que lo utilice más de un usuario al mismo tiempo.
diseño de software <i>software design</i>	Aplicación sistemática del conocimiento, los métodos y la experiencia científicos y técnicos al diseño, la implementación y las pruebas del software para optimizar su producción y soporte.
reutilización del software <i>software reuse</i>	<sup>NS</sup> Creación de clases que operan en una gran variedad de objetos diferentes y que se pueden “incorporar” a un proyecto en curso, lo que conlleva la reducción en el costo del software y el aumento en la fiabilidad.
reconocimiento del discurso (reconocimiento de voz) <i>speech recognition</i> ( <i>voice recognition</i> )	Proceso que compara palabras habladas con las almacenadas en el sistema.
pila <i>stack</i>	<sup>NS</sup> Estructura de datos abstracta en la que sólo la parte superior es accesible para la inserción y recuperación de elementos (LIFO).
topología de estrella <i>star topology</i>	Red en la que cada dispositivo está conectado a un hub central. Véase también las definiciones de “topología de árbol” y “topología de bus”.
estructura de datos estática <i>static data structure</i>	Estructura de datos cuyo tamaño y naturaleza están determinados antes de la ejecución de un programa.
requisitos de almacenamiento <i>storage requirements</i>	Descripción de la cantidad de memoria necesaria durante la ejecución del programa.

guión gráfico ( <i>storyboard</i> ) <i>storyboard</i>	Forma diagramática de un prototipo en la que se muestra una secuencia planificada de pantallas en la que se puede ver las diferentes rutas disponibles para el usuario.
diagrama de estructura <i>structure diagram</i>	Diagrama en el que se representan las relaciones operativas entre las partes de un sistema o programa.
subclase <i>subclass</i>	<sup>NS</sup> Clase que tiene los atributos y métodos de una superclase.
subprograma <i>subprogram</i>	Programa al que se invoca desde otro programa.
subárbol <i>subtree</i>	<sup>NS</sup> Árbol que forma parte de otro árbol.
superclase <i>superclass</i>	<sup>NS</sup> Clase que cede sus atributos y métodos a una subclase.
sintaxis <i>syntax</i>	Reglas que determinan la estructura de las instrucciones de los lenguajes, concretamente, reglas para formar instrucciones correctamente en un lenguaje fuente.
error de sintaxis <i>syntax error</i>	Error en las reglas que determinan la estructura de las instrucciones de un lenguaje.
documentación del sistema <i>system documentation</i>	Documentación del resultado de la fase de análisis de sistemas, en la que se expone el objetivo de dicho sistema, las entradas y salidas necesarias, un plan de pruebas y los resultados esperados.
ciclo de vida del sistema <i>system life cycle</i>	Transcurso de los cambios por los que pasa el sistema durante su desarrollo, desde su concepción hasta que no se utiliza más; por ejemplo, las fases y actividades asociadas con el análisis, la adquisición, el diseño, el desarrollo, las pruebas, la integración, el funcionamiento, el mantenimiento y la modificación de un sistema.
analista de sistemas <i>systems analyst</i>	Persona que lleva a cabo una investigación sistemática de un sistema real o planificado para determinar los requisitos de información y los procesos del sistema, y cómo estos se relacionan con los demás y con otro sistema.
diseño de sistemas <i>systems design</i>	Investigación y registro de los sistemas existentes, así como el diseño de otros nuevos.

diagrama de flujo de un sistema <i>systems flowchart</i>	Diagrama de flujo utilizado para describir un sistema de procesamiento de datos completo, desde el flujo de datos, pasando por las operaciones administrativas necesarias, hasta llegar al nivel de programas individuales, pero sin incluir detalles sobre dichos programas.
TCP/IP (protocolo de control de transmisión/protocolo de Internet) <i>TCP/IP (transmission control protocol/Internet protocol)</i>	Conjunto de protocolos de comunicación utilizados para conectar hosts en Internet.
diseño descendente <i>top-down design</i>	Método de resolución de problemas que consiste en la división de los mismos en subproblemas. Éstos se descomponen a su vez hasta obtener una representación en pseudocódigo que se pueda utilizar como base para la construcción del programa. Véase también la definición de “lenguaje modular”.
rastreo <i>trace</i>	Registro de la ejecución de un algoritmo informático que muestra las secuencias en las que se han ejecutado las instrucciones.
pista <i>track</i>	<sup>NS</sup> Serie de anillos concéntricos que el sistema operativo escribe en la superficie de un disco.
archivo de transacción <i>transaction file</i>	Archivo temporal que contiene datos que posteriormente se utilizarán para su procesamiento, normalmente para actualizar un archivo maestro. Véase también la definición de “archivo maestro”.
traductor <i>translator</i>	Programa informático que transforma todo o parte de un programa expresado en un lenguaje de programación en otro lenguaje o en un lenguaje máquina adecuado para la ejecución. Véase también las definiciones de “compilador” e “intérprete”.
árbol <i>tree</i>	<sup>NS</sup> Estructura de datos no lineal (que representa un sistema de datos estrictamente jerárquico) en el que cada elemento de datos se concibe como un nodo.
topología de árbol <i>tree topology</i>	Red en que se combina las características de las topologías de bus y estrella. Grupos de topologías de estrella que se conectan a un cable central. Véase también las definiciones de “topología de estrella” y “topología de bus”.

truncamiento <i>truncation</i>	NS	1. Proceso de aproximación a un número ignorando toda la información posterior a un número determinado de cifras significativas. Error de truncamiento es el error introducido por este proceso.  2. Eliminación u omisión de una parte inicial o final de una cadena de acuerdo con unos criterios especificados.
tabla de verdad <i>truth table</i>	NS	Tabla en la que se describe una función lógica mediante la enumeración de todas las combinaciones posibles de los valores de entrada y la indicación del valor de salida para cada combinación.
complemento a dos <i>two's complement</i>	NS	Método de representación de números negativos en sistema binario.
operador unario <i>unary operator</i>	NS	Operador que sólo necesita un operando para obtener un resultado simple; por ejemplo, la negación (barra horizontal sobre la expresión booleana). Véase también la definición de “operador binario”.
árbol no equilibrado <i>unbalanced tree</i>	NS	Árbol cuyos subárboles derecho e izquierdo difieren en altura, como mínimo, en un elemento. Véase también la definición de “árbol equilibrado”.
subdesbordamiento <i>underflow</i>	NS	Generación de un resultado cuyo valor es demasiado pequeño para el rango de la representación numérica usada. Véase también la definición de “desbordamiento”.
Unicode <i>Unicode</i>		Conjunto de caracteres de 16 bits estandarizado que permite representar los conjuntos de caracteres de la mayoría de idiomas. Véase también la definición de “ASCII”.
método definido por el usuario <i>user-defined methods</i>		Método escrito por el usuario que no es inherente al lenguaje de programación.
objeto definido por el usuario <i>user-defined objects</i>		Objeto cuyos miembros y métodos son definidos por el usuario y no son inherentes al lenguaje de programación.
interfaz de usuario <i>user interface</i>		Hardware y/o software que permiten que un usuario interactúe y realice operaciones en un sistema, programa o dispositivo.
utilidad <i>utility</i>		Programa diseñado para realizar una tarea cotidiana, como copiar datos desde un dispositivo de almacenamiento a otro.

validación (entrada de datos) <i>validation (data input)</i>	Proceso para comprobar, mediante software, que el tipo de datos introducido es correcto y se encuentra dentro de los límites razonables. Véase también la definición de “verificación (entrada de datos)”.
registro de longitud variable <i>variable-length records</i>	<sup>NS</sup> Registro cuya longitud no está determinada previamente. A cada registro se le asigna el espacio necesario para almacenar la información que contiene. Véase también la definición de “registro de longitud fija”.
verificación (entrada de datos) <i>verification (data input)</i>	Método para asegurar que los datos que hay en el sistema informático son los mismos que los datos fuente originales. Esto puede hacerse mediante entradas dobles. Véase también la definición de “validación (entrada de datos)”.
memoria virtual <i>virtual memory</i>	Uso de la memoria secundaria como si fuera primaria.
virus <i>virus</i>	Programa que infecta otros programas o archivos añadiendo una copia de sí mismo a los archivos destino.
antivirus <i>virus checker</i>	Programa de utilidad que busca y elimina los virus conocidos.
red de área ancha (WAN) <i>wide area network (WAN)</i>	Red que proporciona servicios de comunicación a un área geográfica mayor que la que cubre una red de área local o una red de área metropolitana y que puede proporcionar o utilizar instalaciones públicas de comunicación. Véase también la definición de “red de área local (LAN)”.
palabra <i>word</i>	Grupo de bits que la unidad central de procesamiento puede direccionar, transferir y manipular como una sola unidad.
<b>xor</b> <i>xor</i>	<sup>NS</sup> (Puerta “or” exclusiva). La salida es verdadera si las dos entradas son diferentes; la salida es falsa si las dos entradas son similares.

Traducido y adaptado por IBO con la autorización de Pearson Education Limited, a partir del original en inglés.

## Glosario inglés–español

Inglés	Español
abstract data structure	estructura de datos abstracta
accessor methods	métodos accesoros
accumulator	acumulador
address bus	bus de direcciones
ADSL (Asymmetrical Digital Subscriber Line)	ADSL (Línea asimétrica digital de abonado)
algorithm	algoritmo
ALU	ALU
analog data	datos analógicos
<b>and</b>	<b>and</b>
applet (java)	applet (Java)
application (java)	aplicación (Java)
archive	archivo
argument	argumento
arithmetic and logic unit (ALU)	unidad aritmético lógica (ALU)
array	matriz
ASCII: American Standard Code for Information Interchange	ASCII: Código estándar estadounidense para el intercambio de información
attribute	atributo
A–D converter	convertidor A/D

---

B	B
back-up (file)	copia de seguridad (archivo)
balanced tree	árbol equilibrado
bar code	código de barras
bar code reader	lector de códigos de barras
base	base
batch processing	procesamiento por lotes
behaviour	comportamiento
BigO notation	notación O mayúscula
binary operator	operador binario
binary search	búsqueda binaria
binary tree	árbol binario
bit (b)	bit (b)
block	bloque
BMP	BMP
boolean expression	expresión booleana
bps	bps
browser	navegador
bubble sort	ordenación por el método de la burbuja
buffer	búfer
bus	bus

bus topology	topología de bus
byte (B)	byte (B)
cable	cable
cache	caché
CASE	CASE
character set	conjunto de caracteres
check digit	dígito de verificación
check sum	suma de verificación
circular queue	cola circular
clash (collision)	conflicto (colisión)
class	clase
client	cliente
client–server	cliente-servidor
collection	colección
command language	lenguaje de órdenes
compiler	compilador
computer architecture	arquitectura de computadores
computer program	programa de computador
computer-assisted software engineering	ingeniería del software asistida por computador
constructor method	método constructor
CRC cards	tarjetas CRC

---

cylinder	cilindro
data bus	bus de datos
data compression	compresión de datos
data integrity	integridad de los datos
data member	miembro dato
data packet	paquete de datos
data protection	protección de datos
data security	seguridad en los datos
database management system (DBMS)	sistema de gestión de bases de datos (SGBD)
DBMS	SGBD
De Morgan's law	ley de De Morgan
debugging tool	herramienta de depuración
defragmentation software	software de desfragmentación
dequeue	quitar de la cola
digital data	datos digitales
digital signature	firma digital
direct access file	archivo de acceso directo
disk cache	caché de disco
distributed processing	procesamiento distribuido
DMA	DMA (acceso directo a memoria)
double buffering	búfering doble

doubly linked list	lista doblemente enlazada
dynamic data structure	estructura de datos dinámica
encapsulation	encapsulación
encryption	encriptación
enqueue	añadir a la cola
exception	excepción
exception handler	manipulador de excepciones
expression	expresión
fibre optic	fibra óptica
field (object attribute)	campo (atributo de objeto)
FIFO	FIFO
file	archivo
file manager	gestor de archivos
fixed point	punto fijo
fixed-length records	registro de longitud fija
flag	indicador
floating point	punto flotante
formal parameter	parámetro formal
formatted output	salida formateada
fully-indexed file	archivo completamente indexado
gateway	pasarela

---

graphics tablet (graphics pad)	tableta (almohadilla) digitalizadora
GUI	GUI
hacking	hacking
handshaking	protocolo de intercambio
hash code	código hash
hash table	tabla hash
hexadecimal	hexadecimal
high-level language	lenguaje de alto nivel
HTML (Hyper Text Markup Language)	HTML (lenguaje de marcas de hipertexto)
hub	hub
IDE (integrated development environment)	IDE (entorno de desarrollo integrado)
identifier	identificador
in-order traversal	recorrido en orden
infix notation	notación infija
inheritance	herencia
insertion sort	ordenación por inserción
interface	interfaz
interpreter	intérprete
interrupt	interrupción
ISDN (integrated services digital network)	RDSI (red digital de servicios integrados)
ISO	ISO

iteration	iteración
JPEG (joint photographic expert group)	JPEG (joint photographic expert group)
keys	clave
latency	latencia
left-child	hijo-izquierdo
library manager	gestor de bibliotecas
LIFO	LIFO
linked list	lista enlazada
linker	enlazador
loader	cargador
local area network (LAN)	red de área local (LAN)
local variable	variable local
logic circuit	circuito lógico
logic error	error lógico
logic gate	puerta lógica
magnetic ink character recognition (MICR)	reconocimiento de caracteres de tinta magnética (MICR)
mainframe	computador central
master file	archivo maestro
memory address register (MAR)	registro de dirección de memoria (MAR)
memory manager	gestor de memoria

---

memory mapped I/O	E/S mapeada por memoria
menu	menú
method	método
method signature	firma de un método
MICR	MICR
microprocessor	microprocesador
microwave transmission	transmisión de microondas
modem	módem
modular language	lenguaje modular
modularity	modularidad
module	módulo
modulo arithmetic	aritmética de módulo
multi-processing	multiprocesamiento
multi-tasking	multitarea
multi-user system	sistema multiusuario
<b>nand</b>	<b>nand</b>
network	red
networking	interconexión
node	nodo
<b>nor</b>	<b>nor</b>
<b>not</b>	<b>not</b>

object	objeto
object-oriented programming (OOP)	programación orientada a objetos (OOP)
OCR	OCR
OMR forms	formularios para OMR
on-line	en línea
on-line processing (interactive)	procesamiento en línea (interactivo)
open systems interconnection (OSI)	interconexión de sistemas abiertos (OSI)
operand	operando
operating system (OS)	sistema operativo (OS)
operator	operador
operator precedence	precedencia de operadores
<b>or</b>	<b>or</b>
overflow	desbordamiento
packet	paquete
packet switching	conmutación de paquetes
parallel interface	interfaz paralela
parameter	parámetro
parameter passing	paso de parámetros
parent (node)	padre (nodo)
parity bit	bit de paridad
parsing	análisis sintáctico

---

partially-indexed file	archivo parcialmente indexado
pass-by-reference	paso por referencia
pass-by-value	paso por valor
peripheral device	dispositivo periférico
pointer	puntero
pointing device	dispositivo apuntador
polling	sondeo
polymorphism	polimorfismo
pop	sacar
port	puerto
post-order traversal	recorrido en orden posterior
postfix notation	notación postfija
pre-order traversal	recorrido en orden previo
prefix notation	notación prefija
primary memory	memoria principal
primitive data type	tipo de dato primitivo
private class members	miembros de clase privados
program counter	contador de programa
protocol	protocolo
prototyping	creación de prototipos
pseudocode	pseudocódigo

public class members	miembros de clase públicos
push	meter
queue	cola
quicksort	ordenación rápida
real-time processing	procesamiento en tiempo real
record	registro
recursion	recursividad
reference	referencia
register	registro
requirements specification	especificación de requisitos
right-child	hijo-derecho
robotics	robótica
robustness	robustez
rotational delay	retardo rotacional
router	encaminador
search engine	motor de búsqueda
secondary memory	memoria secundaria
sector	sector
security	seguridad
seek time	tiempo de búsqueda
selection sort	ordenación por selección

---

semantics	semántica
sensor	sensor
sentinel	centinela
sequential access	acceso secuencial
sequential file	archivo secuencial
sequential search	búsqueda secuencial
serial interface	interfaz serie
server	servidor
signature	firma
simulation	simulación
single-tasking	monotarea
single-user system	sistema monousuario
software design	diseño de software
software reuse	reutilización del software
speech recognition (voice recognition)	reconocimiento del discurso (reconocimiento de voz)
stack	pila
star topology	topología de estrella
static data structure	estructura de datos estática
storage requirements	requisitos de almacenamiento
storyboard	guión gráfico ( <i>storyboard</i> )
structure diagram	diagrama de estructura

subclass	subclase
subprogram	subprograma
subtree	subárbol
superclass	superclase
syntax	sintaxis
syntax error	error de sintaxis
system documentation	documentación del sistema
system life cycle	ciclo de vida del sistema
systems analyst	analista de sistemas
systems design	diseño de sistemas
systems flowchart	diagrama de flujo de un sistema
TCP/IP (transmission control protocol/Internet protocol)	TCP/IP (protocolo de control de transmisión/protocolo de Internet)
top-down design	diseño descendente
trace	rastreo
track	pista
transaction file	archivo de transacción
translator	traductor
tree	árbol
tree topology	topología de árbol
truncation	truncamiento

---

truth table	tabla de verdad
two's complement	complemento a dos
unary operator	operador unario
unbalanced tree	árbol no equilibrado
underflow	subdesbordamiento
Unicode	Unicode
user interface	interfaz de usuario
user-defined methods	método definido por el usuario
user-defined objects	objeto definido por el usuario
utility	utilidad
validation (data input)	validación (entrada de datos)
variable-length records	registro de longitud variable
verification (data input)	verificación (entrada de datos)
virtual memory	memoria virtual
virus	virus
virus checker	antivirus
wide area network (WAN)	red de área ancha (WAN)
word	palabra
<b>xor</b>	<b>xor</b>

# APÉNDICE 2

---

## Subconjunto de herramientas de Java para el examen (JETS)

El programa de estudios de Informática tiene como requisito el aprendizaje del lenguaje de programación Java por parte de los alumnos. Esto **no** significa aprender la **totalidad** de Java, lo cual resultaría inviable dada la cantidad de bibliotecas (libraries) y clases (classes), y el constante cambio en el lenguaje. El objetivo no es formar alumnos “expertos” en Java, sino utilizar la plataforma que provee Java para que los alumnos desarrollen y demuestren sus conocimientos de los conceptos **algorítmicos** fundamentales. Por lo tanto, los alumnos sólo deben aprender un subconjunto del lenguaje, denominado Subconjunto de herramientas de Java para el examen (JETS).

Los profesores podrán encontrar ejemplos de estos algoritmos en el material de ayuda al profesor para esta asignatura.

En las preguntas de examen, solamente aparecerán los comandos, símbolos y estructuras especificados en JETS. No se pedirá a los alumnos que lean o escriban respuestas referidas a otras clases y otros métodos. Dado que el dossier de trabajo personal también debe estar escrito en Java, los alumnos podrían querer utilizar en sus respuestas de examen estructuras (constructs) y clases (classes) que hayan aprendido durante la creación del dossier de trabajo personal. Sin embargo, algunas clases y algunos métodos están específicamente prohibidos ya que contienen instrucciones (commands) que implementan algoritmos que los alumnos deberán construir a partir de estructuras (constructs) más simples. Por ejemplo, el paquete **java.util** no está permitido ya que contiene bibliotecas que implementan algoritmos de ordenación.

JETS también especifica una **nomenclatura** y un **estilo** para las preguntas de examen. Los profesores deben lograr que sus alumnos se familiaricen con JETS, incluyendo las convenciones de estilo y nomenclatura. El propósito de estas convenciones es hacer que las preguntas de examen resulten más claras y fáciles de comprender. No se exigirá a los alumnos apegarse a ellas en sus respuestas. Sin embargo, deben escribir respuestas claras, coherentes y legibles, y no deben utilizar bibliotecas de tipo no estándar que transformen la solución en algo trivial.

**No** se exigirá a los alumnos que escriban sus respuestas con una sintaxis **perfecta** (por ejemplo, en general no se penalizaría un error de mayúsculas o la omisión de un punto y coma), pero se penalizarán los errores que cambien sustancialmente el **significado** del algoritmo (por ejemplo, si se omite un signo de exclamación).

Se espera que tanto los alumnos como los examinadores empleen un estilo lo más claro y fácil de leer que sea posible. Los alumnos deberán tener especial cuidado y evitar escribir en una sintaxis que resulte difícil de leer, como ser el uso de doble signos de menos (--) u operadores de asignación compuestos, tales como -=. Por ejemplo:

**x = x + 1** es más claro que **x++** o **x += 1**

**x = x - 1** es más claro que **x--** o **x -= 1**.

# Presentación de JETS

## Convenciones de estilo

Las convenciones de estilo que se utilizarán en todas las pruebas de examen son las siguientes:

- Las preguntas de examen e instrucciones generales se imprimirán en el tipo de letra Times New Roman (proporcional) tamaño 12. Algunos enunciados se imprimirán en *cursiva*. El código JETS se imprimirá en letra tipo Courier (espaciado fijo) tamaño 10.5.
- Todas las palabras reservadas se escribirán en **minúsculas y negrita**.
- Los nombres de las clases siempre empezarán con Mayúsculas.
- Los nombres de variables y métodos siempre empezarán con letra minúscula.
- Los identificadores `MultiPalabra` utilizarán mayúsculas para separar las palabras (no se utilizará el guión bajo).
- Los identificadores generalmente utilizarán palabras enteras y no abreviaciones ni acrónimos.
- Se utilizará siempre una sangría adecuada.
- El orden de los módulos no es importante aunque el programa principal (**main**) y/o el método **constructor** deberán siempre figurar al principio de la clase.
- Algunas preguntas de examen pueden incluir ejemplos de sentencias para ayudar al alumno a recordar el uso de determinados comandos poco frecuentes. Por ejemplo: ***Recuerde que `String.indexOf(String)` puede utilizarse para hallar la posición de un string en otro, de la siguiente forma:***

```
String email = "exams@ibo.org";
int arroba = email.indexOf("@"); //resulta en 5
```

- Se puede explicar el uso de ciertos elementos del lenguaje no estándar (clases de biblioteca) escribiendo: “Una biblioteca provee el método | tipo de datos...”, seguido de una explicación y un ejemplo.

### En las versiones en español y francés de las pruebas de examen:

- las palabras reservadas permanecerán en inglés
- las constantes de texto se traducirán
- los identificadores definidos por el usuario (nombres de clases, variables y métodos) se traducirán según corresponda.

## La sintaxis de JETS

### Operadores

Aritméticos: +, -, \*, /, % (los alumnos deberán entender el comportamiento polimórfico del operador de división, por ejemplo: int / int ==> int)

Relacionales: ==, >, <, >=, <=, !=

Booleanos: !, &&, ||

(No se requieren los operadores booleanos bit-a-bit &, |.)

### Precedencia de operadores

Se asume que los alumnos conocen el estándar de precedencia de operadores en Java. Las preguntas de examen podrían incluir paréntesis adicionales con el propósito de mejorar la claridad de los enunciados. Se debe además incentivar a los alumnos a usarlos en sus soluciones.

### Valores constantes

**string** : "entre comillas"

**char** : 's' // entre comillas simples

**integer** : 123456 o -312

**double** : 124.75 (punto fijo) o 1.2475E+02 (punto flotante)

**boolean** : true, false

Los identificadores de **constantes** se escribirán **TODOS\_EN\_MAYÚSCULA** utilizando el guión bajo para separar las palabras. Los mismos se definirán como campos **final static** :

```
final static double NATURAL_LOG_BASE = 2.1782818;
```

### Tipos de datos primitivos (o atómicos)

**byte int long double char boolean**

(**short** y **float** no se incluirán)

### Tipos de datos estructurados

**class String**

**class StringBuffer**

**Arreglos Lineales** : **int**[ ] **numeros** = **new int**[100];

(un arreglo de 100 enteros indexados del 0 al 99)

**arreglos 2-D**: **int**[ ][ ] **checkers** = **new int**[8][8];

**Archivos de texto** (archivos secuenciales)

**Archivos de acceso aleatorio** (campos como tipo de datos atómicos)

\*\* Las clases numéricas envolventes Integer, Double, etc., sólo se utilizarán para proveer funcionalidades de métodos estáticos con el propósito de realizar conversiones de tipo, como se expone a continuación en métodos IBIO.

## Pasaje de parámetros

Se seguirá la especificación en Java. Por ejemplo, los tipos primitivos (o atómicos) serán automáticamente parámetros por valor y los tipos estructurados (arreglos y objetos) serán siempre parámetros por referencia.

## Símbolos

```
/* comentarios  
multi-línea */
```

```
// comentarios  
// de una línea
```

( ) paréntesis curvos para pasaje de parámetros

[ ] paréntesis rectos para índices de arreglos

. notación de punto para derreferenciar métodos y campos de objetos

{ } para definir los bloques de código

{ 1 , 2 , 3 } para inicializar un arreglo

Se utilizará el siguiente conjunto de comandos IBIO.

## Métodos de entrada

Todos los métodos de entrada despliegan un mensaje (*String*), aceptan una entrada de teclado hasta que el usuario oprima la tecla [Intro] y devuelven un valor del tipo especificado. Se puede asumir que ninguna de las rutinas de entrada producen errores en tiempo de ejecución. Si el usuario ingresa un *String* que no puede convertir al tipo correcto, la rutina de entrada devolverá un valor por defecto, por ejemplo un *String* vacío, un valor numérico 0, etc.

```
String inputString(String mensaje)
String input(String mensaje)
String input() // no imprime mensaje previo a la entrada
char inputChar(String mensaje)
boolean inputBoolean(String mensaje)
byte inputByte(String mensaje)
int inputInt(String mensaje)
long inputLong(String mensaje)
double inputDouble(String mensaje)
output(String)           --> despliega un String
output(char)            --> despliega un valor de tipo char
output(boolean)         --> despliega un valor de tipo boolean
output(byte)            --> despliega un valor de tipo byte
output(int)             --> despliega un valor de tipo int
output(long)           --> despliega un valor de tipo long
output(double)         --> despliega un valor de tipo double
```

JETS también utiliza los comandos *System* de salida de consola:

```
System.out.print(String)
System.out.println(String)
// System.in.read() no está incluido en JETS, pero se utiliza en IBIO.
```

## Bucles y condicionales

```

if (condicion booleana)
    { ... comandos ... }
else if (condicion booleana)
    { ... comandos ... }
else
    { ... comandos ... };

// switch..break.. no se incluye en JETS, pero los alumnos pueden
// utilizarlo en sus respuestas si así lo desean.

for ( incio; limite; incremento)
    { ...comandos... };

while (condicion booleana)
    { ...comandos... };

do
    { ...comandos... }
while (condicion booleana) ;

```

## Archivos

Nivel Medio/Nivel Superior

**BufferedReader(FileReader)** - abre un archivo secuencial en modo lectura

```

.read
.read
.readLine
.close

```

**PrintWriter(FileWriter)** - abre un archivo secuencial en modo escritura

```

.read
.print
.println
.close

```

// **No se requiere** el concepto de serialización.

Solamente Nivel Superior

**RandomAccessFile**

constructor: randomAccessFile(String nombreArchivo, String modoDeAcceso)

```

.seek
.length
.read .... readInt, readDouble, readBytes, readUTF
.write .... writeInt, writeDouble, writeBytes, writeUTF
.close

```

## Métodos estándar y miembros de datos

### class Math

-----

.abs, .pow, .sin, .cos, .round, .floor

### class String

-----

+ para concatenar

.equals(String)  
.substring(posComienzo, posFinal)  
.length()  
.indexOf(String)  
.compareTo(String)  
.toUpperCase()  
.toLowerCase()

### Arreglos

-----

.length

### (casts)

-----

(int) (double) (byte) (char)

(numeric + "") // para convertir un valor numérico a String

## Métodos estáticos

Los alumnos deben tener presente que los métodos estáticos (**static**) de algunas clases pueden utilizarse sin crear una instancia del objeto. Tal es el uso de **Integer.parseInt(stringVal)** para convertir un string a un integer (sin instanciar **new Integer**).

Se requiere la comprensión de la construcción **new**. Los alumnos deben saber que **new** crea una nueva instancia del objeto y que esto es distinto a declarar una variable de tipo primitivo o atómico. Los conceptos de **alcance** y **tiempo de vida** de las referencias a identificadores deben comprenderse claramente, así como el hecho de que algunas instancias pueden ser **destruidas automáticamente** por el **recolector de basura** cuando caen fuera de su alcance. Por ejemplo, deben entender que una variable local de un método perderá su valor cuando el método finalice su ejecución y que tal valor no podrá ser recuperado en invocaciones subsecuentes al mismo método. Se requiere comprender el concepto de estático (**static**) pero no se evaluará directamente en el código (podría figurar, pero el significado en el código no se examinará directamente).

## Asignación dinámica de memoria (solamente NS)

Los alumnos también deben comprender que cuando se declara un objeto sin instanciar, el mismo se puede luego reasignar a modo referencial (puntero) hacia una nueva instancia o a otra previamente creada.

## Otras consideraciones sintácticas

Java permite que una sentencia se desarrolle en varias líneas. Esto está permitido en preguntas de examen siempre y cuando ello mejore la legibilidad y claridad del código. Por ejemplo, en una lista extensa de parámetros:

```
public int ordenarArreglo( String[ ] nombres ,
                          int tamanoLista ,
                          char ascendenteDescendente )
```

Las llaves siempre deberán estar alineadas ya sea horizontal o verticalmente.

```
public void imprimirNumeros()
{   int x = 0;
    while ( x < 10 )
    { output( x ); } //llaves de bucle alineadas horizontalmente
} //llaves del cuerpo del método
//alineadas verticalmente
```

## Alcance de clases

**public, private**

// **implements** y **abstract** no se incluyen

// **interface** no se incluye

## Estructura general de las clases

Los alumnos deben entender el concepto de **constructor** y método principal (**main method**), así como la diferencia entre ambos. También deben entender el concepto de **extends**.

No se examinará el uso de **Applets** en códigos algorítmicos, pero podrán preguntarse algunos **conceptos** de applets (por ejemplo, seguridad).

## Manejo de errores

```
try { ...commandos... }
catch (Exception e) { ...manejar el error... };
```

*// En los exámenes, el manejo de errores se limitará simplemente a mostrar un mensaje de error, modificar una bandera (flag) o retornar del método. No se esperará que se capturen excepciones específicas: solamente se deberán capturar las excepciones genéricas, Exception e IO Exception.*

```
nombreDelMetodo() throws IOException
```

Los alumnos deben comprender la idea de lanzar (**throw**) una excepción en vez de capturarla con **try...catch....**

## Algoritmos que ejemplifican los elementos de JETS

Los ejemplos que se presentan a continuación pretenden ilustrar la mayor parte de los elementos del lenguaje JETS. En los exámenes, la mayoría de los algoritmos serán considerablemente más cortos que estos ejemplos. Los mismos fueron compilados con JDK 1.3 (Java 2) de Sun Microsystems. Funcionan como aplicaciones consola (modo texto) utilizando una biblioteca estándar de métodos entrada/salida de consola (IBIO).

---

```
//- HOLA - ejemplifica métodos simplificados de
//      entrada/salida (IBIO) -
public class Hola
{ public static void main(String[] args)
  { new Hola();}

  public Hola()
  { String nombre = inputString("¿Cual es tu nombre?");
    int edad = inputInt("¿Que edad tienes?");
    output("Hola " + nombre);
    output("En el año 2010, tendras " + (edad + 7) + " años ");
  }

//=====
//  A continuación se presentan los métodos simplificados de
//  entrada y salida.
//  Los mismos se copiarán en el código fuente en todos los algoritmos.
//  Al final de cada algoritmo, habrá una nota recordatoria para que los
//  alumnos no lo olviden. Los alumnos deberán
//  comprender el USO de estos métodos y no memorizar el código.
//=====

  static void output(String info)
  { System.out.println(info);
  }
  static void output(char info)
  { System.out.println(info);
  }
  static void output(byte info)
  { System.out.println(info);
  }
  static void output(int info)
  { System.out.println(info);
  }
  static void output(long info)
  { System.out.println(info);
  }
  static void output(double info)
  { System.out.println(info);
  }
  static void output(boolean info)
  { System.out.println(info);
  }
  static String input(String prompt)
  { String inputLine = "";
    System.out.print(prompt);
    try
    {inputLine = (new java.io.BufferedReader(
      new java.io.InputStreamReader(System.in))).readLine();}
```

```

        catch (Exception e)
        { String err = e.toString();
          System.out.println(err);
          inputLine = "";
        }
        return inputLine;
    }
    static String inputString(String prompt)
    { return input(prompt);
    }
    static String input()
    { return input("");
    }
    static int inputInt()
    { return inputInt("");
    }
    static double inputDouble()
    { return inputDouble("");
    }
    static char inputChar(String prompt)
    { char result=(char)0;
      try{result=input(prompt).charAt(0);}
      catch (Exception e){result = (char)0;}
      return result;
    }
    static byte inputByte(String prompt)
    { byte result=0;
      try{result=Byte.valueOf(input(prompt).trim()).byteValue();}
      catch (Exception e){result = 0;}
      return result;
    }
    static int inputInt(String prompt)
    { int result=0;
      try{result=Integer.valueOf(
          input(prompt).trim()).intValue();}
      catch (Exception e){result = 0;}
      return result;
    }
    static long inputLong(String prompt)
    { long result=0;
      try{result=Long.valueOf(input(prompt).trim()).longValue();}
      catch (Exception e){result = 0;}
      return result;
    }
    static double inputDouble(String prompt)
    { double result=0;
      try{result=Double.valueOf(
          input(prompt).trim()).doubleValue();}
      catch (Exception e){result = 0;}
      return result;
    }
    static boolean inputBoolean(String prompt)
    { boolean result=false;
      try{result=Boolean.valueOf(
          input(prompt).trim()).booleanValue();}
      catch (Exception e){result = false;}
      return result;
    }
    //===== end IBIO =====
}

```

```
//-----  
// QUADRATIC encuentra las raíces de un polinomio cuadrático.  
//-----  
  
public class Quadratic  
{ public static void main(String[] args)  
  { new Quadratic();  
    public Quadratic()  
    { int a = inputInt("A? ");  
      int b = inputInt("B? ");  
      int c = inputInt("C? ");  
      if (esResoluble(a,b,c))  
        { output("x1 = " + raizMayor(a,b,c));  
          output("x2 = " + raizMenor(a,b,c));  
        }  
      else  
        { output("No tiene raices");  
          input("--- oprima [Intro] ---");  
        }  
    }  
  
    boolean esResoluble(int a, int b, int c)  
    { if ((a != 0) && (discriminante(a,b,c) < 0))  
      { return false; }  
      else  
        { return true; }  
    }  
  
    double discriminante(int a, int b, int c)  
    { return b*b - 4*a*c;  
    }  
  
    double raizMenor(int a, int b, int c)  
    { return (-b - Math.pow(discriminante(a,b,c),0.5) ) / (2*a);  
    }  
  
    double raizMayor(int a, int b, int c)  
    { return (-b + Math.pow(discriminante(a,b,c),0.5) ) / (2*a);  
    }  
  
    //-----  
    //-- IBIO - incluir métodos simplificados de entrada y salida --  
    //-----  
  }  
}
```

```

//-----
// Ejemplo de algoritmo: GuardaNombres - ingresa una lista de nombres
// a un arreglo. "XXX" termina la entrada y luego la lista se
// almacena en un archivo secuencial.
// Esta clase no maneja las excepciones IOException
// (por ejemplo, archivo protegido o sin lugar en disco) sino que
// simplemente las lanza (throws).
//-----

import java.io.*;

public class GuardaNom
{ public static void main(String[] args) throws IOException
  { new GuardaNom();

  String nombres[] = new String[1000];
  int cuentaNombres = 0;

  public GuardaNom() throws IOException
  { entrarNombres();
    guardarNombres();
  }

  void entrarNombres()
  { String esteNombre = "";
    cuentaNombres = 0;
    do
    { output("Ingrese un nombre");
      esteNombre = input();
      if (!esteNombre.equals("XXX"))
      { nombres[cuentaNombres] = esteNombre;
        cuentaNombres = cuentaNombres + 1;
      }
    } while (!esteNombre.equals("XXX") && (cuentaNombres < 1000));
  }

  void guardarNombres() throws IOException
  { PrintWriter outFile = new PrintWriter(new
    FileWriter("listanombres.txt"));
    for (int c = 0; c < cuentaNombres; c++)
    { outFile.println(nombres[c]);
    }
    outFile.close();
  }

  //-----
  //-- IBIO - incluir métodos simplificados de entrada y salida --
  //-----
}

```

```
//-----
// ENCRYPT - Encripta un string sumándole al código ASCII de cada
// mayúscula la longitud del string. Luego, el resultado se imprime
// en orden inverso. Nótese que sólo las mayúsculas serán
// modificadas: "HOT2Day" --> sumar 7 --> "OVA2Kay" --> "yaK2AVO"
//-----

public class Encrypt
{ public static void main(String[] args)
  { new Encrypt();

  public Encrypt()
  { String mensaje, codificado;
    output("Ingrese el mensaje");
    mensaje = input();
    codificado = encrypt(mensaje);
    output(reverso(codificado));
    input("---- oprima [Intro] ----");
  }

  String encrypt(String mensaje) // Strings no modificables por
  { int p,num;                  // carácter. Utilice un StringBuffer
    char codigoLetra;           // para acceder a caracteres

    StringBuffer texto = new StringBuffer(mensaje);

    num = texto.length();
    for(p = 0; p < num; p++)
    { codigoLetra = sumarCodigo( texto.charAt(p), num );
      texto.setCharAt(p,codigoLetra);
    }
    return texto.toString();
  }

  char sumarCodigo(char letra,int cambio)
  { if ((letra >= 'A') && (letra <= 'Z')) // chars se comportan
    { char codigoAntes = (char)(letra - 'A') ; // como ints
                                              // Se pueden operar
                                              // aritméticamente
      char codigoNuevo = (char)((codigoAntes + cambio) % 26);

      return (char)('A' + codigoNuevo); // El cast (char) es
    } // necesario para evitar
      // mensajes de advertencia

  else
  { return letra; }
  }

  String reverso(String mensaje)
  { String haciaAtras = "";
    for(int c = mensaje.length() - 1; c >= 0; c = c-1)
    { haciaAtras = haciaAtras + mensaje.charAt(c);
    }
    return haciaAtras;
  }
//-----
//-- IBIO - incluir métodos simplificados de entrada y salida --
//-----
}
```

```
//-----
// OrdenarArchivo muestra cómo almacenar registros en un
// RandomAccessFile (archivo de acceso aleatorio). Java no provee
// estructurados (STRUCT o RECORD). Una "clase interna" puede ser
// utilizada en estos casos. No existe un comando que lea o escriba
// registros enteros por lo tanto esto se debe programar escribiendo
// un campo a la vez.
//-----
import java.io.*; // contiene las clases y los métodos relacionados con
archivos

public class OrdenarArchivo
{ public static void main(String[] args) throws IOException
  { new OrdenarArchivo ();}

  public OrdenarArchivo () throws IOException
  { RandomAccessFile archAleat = new RandomAccessFile("Items.dat","rw");
    crear(archAleat);
    System.out.println("--- Registros antes de ordenar ---");
    mostrar(archAleat);
    ordenar(archAleat);
    System.out.println("--- Registros luego de ordenar ---");
    mostrar(archAleat);
    archAleat.close();
  }

  class Item //----- clase interna que simula un registro -----
  { int id; // clase Item contiene 3 campos de datos
    String nombre; // que serán escritos y leídos desde
    double precio; // el archivo de acceso aleatorio

    final static int LONGNOMBRE = 20;
    final static int TAMANOREGISTRO = LONGNOMBRE*2 + 12;
    // constantes utilizadas para hallar los valores de SEEK

    void leerDeArchivo(RandomAccessFile archAleat, long regNum)
    //-----
    // Lee un registro de archAleat, el cual ya debe estar abierto.
    // Lee cada campo - id, precio, nombre. Utiliza TRIM para
    // eliminar espacios intermedios. Serán capturadas y desplegadas
    // las Excepciones tipo IOException.
    //-----
    { try
      { archAleat.seek( regNum * TAMANOREGISTRO);
        id = archAleat.readInt();
        precio = archAleat.readDouble();
        StringBuffer nombreBuffer = new
          StringBuffer(Item.LONGNOMBRE);
        nombreBuffer.setLength(LONGNOMBRE);
        for (int c = 0; c < LONGNOMBRE; c++)
        { nombreBuffer.setCharAt(c, archAleat.readChar());
        }
        nombre = nombreBuffer.toString().trim();
      }
      catch(IOException exc)
      { System.out.println("Al leer registro # " + regNum);
        System.out.println(exc.toString());
      }
    }
  }
}
```

```

void escribirEnArchivo(RandomAccessFile archAleat, long regNum)
//-----
// Escribe un registro en archAleat, el cual debe estar abierto.
// Serán capturadas y desplegadas las Excepciones tipo
// IOException.
//-----
{ try
  { archAleat.seek( regNum * TAMANOREGISTRO);
    archAleat.writeInt(id);

    archAleat.writeDouble(precio);
    archAleat.writeChars(setLength(nombre, LONGNOMBRE));
  }
  catch(IOException exc)
  { System.out.println("Al escribir " + exc.toString()); }
}

String setLength(String s, int len)
//-----
// Fuerza al string a tener una longitud determinada.
// Esto es necesario al escribir en un archivo de acceso
// aleatorio.
//-----
{ StringBuffer sb = new StringBuffer(s);
  sb.setLength(len);
  return sb.toString();
}
} //---- fin de clase Item -----

void crear(RandomAccessFile archAleat) throws IOException
//-----
// Agrega los registros a archAleat, el cual debe estar abierto.
//-----
{ Item esteReg = new Item();
  for (int c=0; c < 5; c++)
  { esteReg.id = inputInt();
    esteReg.nombre = input();
    esteReg.precio = inputDouble();
    esteReg.escribirEnArchivo(archAleat, c);
  }
}

void mostrar (RandomAccessFile archAleat)
//-----
// Lee todos los registros de archAleat y muestra los campos.
//-----
{ try
  { long cuentaRegistros = archAleat.length()/Item.TAMANOREGISTRO;
    Item esteReg = new Item();
    for (int c=0; c < cuentaRegistros; c++)
    { esteReg.leerDeArchivo(archAleat, c);
      System.out.println(esteReg.id + ":" + esteReg.nombre
        + "=" + esteReg.precio);
    }
  }
  catch (IOException exc)
  { System.out.println(exc.toString()); }
}

```

```
void ordenar(RandomAccessFile archAleat)
//-----
// Ordena archAleat utilizando el método Burbuja en orden
// ascendente de nombres.
//-----
{ try
  { long cuentaRegistros = archAleat.length()/Item.TAMANOREGISTRO;
    Item esteReg = new Item();
    Item sigReg = new Item();
    for (int pass = 0; pass < cuentaRegistros; pass++)
    { for (int pos = 0; pos < cuentaRegistros-1; pos++)
      { esteReg.leerDeArchivo(archAleat,pos);
        sigReg.leerDeArchivo(archAleat,pos+1);
        if (esteReg.nombre.compareTo(sigReg.nombre)>0)
        { sigReg.escribirEnArchivo(archAleat,pos);
          esteReg.escribirEnArchivo(archAleat,pos+1);
        }
      }
    }
  }
  catch (IOException exc)
  { System.out.println(exc.toString());}
}

//-----
//-- IBIO - incluir métodos simplificados de entrada y salida --
//-----
}
```

```

//-----
// Algoritmo ejemplo ArbolDeFactores - genera un árbol de factores
// primos. Este algoritmo es para alumnos del NS puesto que el
// tema árboles binarios no figura en el programa de estudios de NM.
//-----
public class ArbolDeFactores
{ public static void main(String[] args)
  { new ArbolDeFactores();

  class Nodo // Utilice una clase interna como
  { int dato; // estructura de dato similar
    Nodo hijoIzq; // a un RECORD o STRUC en
    Nodo hijoDer; // lenguajes tradicionales de AN
  }

  public ArbolDeFactores()
  { int numero;
    Nodo raiz = null;
    numero = inputInt("Ingrese un numero entero:");
    if (numero > 2)
    { raiz = armarArbol(numero);
      output("Los factores primos son");
      mostrarFactores(raiz);
    }
    output("-----");
    delinear(raiz,"");
    input("");
  }

  Nodo armarArbol(int numero) // arma el árbol de factores
                              // recursivamente.
  { Nodo temp = new Nodo(); // crea un Nodo (asigna memoria)
    temp.hijoIzq = null;
    temp.hijoDer = null;
    temp.dato = numero;
    int cont = 1;
    int fac = 0;
    while (cont*cont <= numero)
    { if ( (numero % cont) == 0 )
      { fac = cont; }
      cont = cont + 1;
    }
    if (fac > 1)
    { temp.hijoIzq = armarArbol(fac);
      temp.hijoDer = armarArbol(numero / fac);
    }
    return temp;
  }

  void mostrarFactores(Nodo aqui)
  { if (aqui == null) { output("null"); return;}
    if ( (aqui.hijoIzq == null) && (aqui.hijoDer == null) )
    { output(aqui.dato);
    }
    else
    { mostrarFactores(aqui.hijoIzq);
      mostrarFactores(aqui.hijoDer);
    }
  }
}

```

```
void delinear(Nodo aqui,String indent)// Recorrida en orden previo
{ output(indent + aqui.dato);          //(pre-order) imprime el
  if (aqui.hijoIzq != null)            //árbol en forma abreviada.
    {delinear(aqui.hijoIzq, indent + "  ");}
  if (aqui.hijoDer != null)
    {delinear(aqui.hijoDer, indent + "  ");}
}

//-----
//-- IBIO - incluir métodos simplificados de entrada y salida --
//-----
}
```

```

//-----
// La siguiente clase Calendario es utilizada por una compañía para
// agendar reuniones, repartos, etc. Todas las funciones aceptan
// fechas en una variedad de formatos ("December 25, 2002" o
// "25 Dec 02" o "12/25/2002") pero los resultados siempre se
// devuelven en el formato "dd MMM yyyy EEE", por ejemplo,
// "01 Jul 1998 Wed". Esto también es aceptado como parámetro.
//-----
//Nota: este código sólo funcionará correctamente en un entorno de
utilización configurado en inglés.

import java.util.*;
import java.text.*;

public class Calendario
{ private static final long UN_DIA = (long)24*60*60*1000;

  private static final SimpleDateFormat dateFormatter =
      new SimpleDateFormat("dd MMM yyyy EEE");

  private static final String feriados[] =
      {"01 Jan", "01 Apr", "01 May", "23 Aug", "25 Dec", "xxxxxx"};

  public static String normalDate(String date)
  //-----
  // Determina el día de la semana (Mon, Tues, Wed,...) y devuelve DATE
  // en el formato estándar dd MMM yyyy EEE. Por ejemplo,
  // normalDate("4/1/2003") --> "01 Apr 2003 Tues". Devuelve string
  // vacío "" si la fecha no es válida.
  //-----
  { try{Date df = new Date(date);
    return normalDate(df);}
    catch(Exception e){return "";}
  }

  private static String normalDate(Date df)
  { try{return dateFormatter.format(df);}
    catch(Exception e){return "";}
  }

  public static int esDiaLaborable(String check)
  //-----
  // Invoca a NORMALDATE, para producir dd MMM yyyy EEE. Si EEE es
  // "Sat" o "Sun", la función devuelve 0 (false).
  // En caso contrario, consulta un archivo de calendario para
  // comprobar si es día feriado, devolviendo 1 para días laborables,
  // 0 para feriados y fines de semana y código de error -1 si CHECK
  // no es una fecha válida.
  //-----
  { String d;
    try { d = normalDate(check); }
    catch (Exception e) { return -1; }

    String objetivo = d.substring(0,6);
    String diaDeLaSemana = d.substring(12,15);
    int diaLaborable = 1;
    if (diaDeLaSemana.equals("Sat") || diaDeLaSemana.equals("Sun"))
    { diaLaborable = 0; }
  }
}

```

```

else
{ int c = 0;
  while (c<5)
  { if (objetivo.equals(feriados[c]))
    { diaLaborable = 0; }
    c = c+1;
  }
}
return diaLaborable;
}

public static String proxDia(String date)
//-----
// Acepta DATE en varios formatos, devolviendo el próximo día en el
// formato estándar dd MMM yyyy EEE. Si DATE es una fecha no válida
// (ejemplo: 1998.37.58) entonces devuelve un string vacío. Se toman
// en consideración los fines de mes, fines de año, años bisiestos,
// etc. Por ejemplo: PROXDIA("28 Feb 1998 Sat") ---->
// "01 Mar 1998 Sun"
//-----
{ return normalDate(new Date(new Date(date).getTime() + UN_DIA));
}

public static int diasEntre(String primera,String segunda)
//-----
// Cuenta la cantidad de días entre dos fechas incluyendo los
// extremos. Si PRIMERA es posterior a SEGUNDA devuelve un número
// negativo. Si PRIMERA y SEGUNDA son la misma fecha devuelve 1. Si
// PRIMERA o SEGUNDA no son fechas válidas, devuelve código de
// error 0.
//-----
{ try
{ Date d1 = new Date(primera);
  Date d2 = new Date(segunda);
  return (int)((long)(d2.getTime() - d1.getTime()) / UN_DIA);
}
catch(Exception exc)
{ return 0; }
}

public static String today()
//-----
// Devuelve la fecha de hoy en el formato estándar dd MMM yyyy EEE
//-----
{ try
{ Date now = new Date();
  return normalDate(
    new Date(now.getYear(),now.getMonth(),now.getDate()));
}
catch (Exception exc)
{ return ""; }
}
}

```

```
//-----  
// Como los métodos son PUBLIC STATIC, otras clases los pueden  
// invocar sin crear un objeto. La funcionalidad provista es la  
// misma que en las bibliotecas tradicionales de procedimientos. Se  
// puede mejorar la posibilidad de reutilización y la fiabilidad mediante  
// un cuidadoso  
// manejo de excepciones. En las preguntas de examen sólo se deben  
// proveer los encabezados de estos métodos y comentarios. Los  
// alumnos no necesitan conocer CÓMO funciona el algoritmo.  
//-----
```

```

//-----
// Algoritmo ejemplo: DIASLABORABLES - pregunta dos fechas y cuenta
// el numero de días laborables entre las dos fechas incluyendo los
// extremos. Se importa la clase Calendario, de manera de poder
// utilizar sus métodos.
//-----

//utiliza clase Calendario, ver página anterior
//Nota: este código sólo funcionará correctamente en un entorno de
utilización configurado en inglés.

public class DiasLaborables
{ public static void main(String[] args)
  { new DiasLaborables();

  public DiasLaborables()
  { String primera,ultima,temp,guardarPrimero;
    int entre;
    output( "Este algoritmo cuenta los dias laborables entre dos fechas");

    primera = "";
    while (primera.equals("")) // fecha inválida devuelve ""
    { output("Ingresar fecha inicial:"); // Iterar hasta obtener
      primera = input(); // fecha válida.
      primera = Calendario.normalDate(primera);
    }

    ultima = "";
    while (ultima.equals("")) // fecha inválida devuelve ""
    { output("Ingresar fecha final:"); // Iterar hasta obtener
      ultima = input(); // fecha válida.
      ultima = Calendario.normalDate(ultima);
    }
    entre = Calendario.diasEntre(primera,ultima);
    if (entre < 0)
    { temp = primera; // Intercambiar PRIMERA y ULTIMA
      primera = ultima;
      ultima = temp;
    }
    guardarPrimero = primera;
    entre = Calendario.esDiaLaborable(primera);
    output(primera);

    while (!primera.equals(ultima)) // No comparar strings con ==
    { primera = Calendario.proxDia(primera);
      entre = entre + Calendario.esDiaLaborable(primera);
      output(primera);
    }
    output(entre + " dias laborables entre " + guardarPrimero + " y " +
ultima );
  }
//-----
//-- IBIO - incluir métodos simplificados de entrada y salida --
//-----
}

```

```
//----- Ejemplo de Salida -----
/*
Este algoritmo cuenta los días laborables entre dos fechas
Ingresar fecha inicial:
12/21/2002
Ingresar fecha final
12/31/2002
21 Dec 2002 Sat
22 Dec 2002 Sun
23 Dec 2002 Mon
24 Dec 2002 Tue
25 Dec 2002 Wed
26 Dec 2002 Thu
27 Dec 2002 Fri
28 Dec 2002 Sat
29 Dec 2002 Sun
30 Dec 2002 Mon
31 Dec 2002 Tue
7 días laborables entre 21 Dec 2002 Sat y 31 Dec 2002 Tue */

//----- fin Ejemplo de Salida -----
```

```
public class TestIBIO
{ public static void main(String[] args)
  { new TestIBIO(); }

  public TestIBIO()
  { String elString = inputString("String:");
    if (elString.equals("1"))
    { output("Si"); }
    else
    { output(elString); }

    char elChar = inputChar("char:");
    if (elChar == '2')
    { output("Si"); }
    else
    { output(elChar); }

    byte elByte = inputByte("byte:");
    if (elByte == 3)
    { output("Si"); }
    else
    { output(elByte); }

    int elInt = inputInt("int:");
    if (elInt == 4)
    { output("Si"); }
    else
    { output(elInt); }

    long elLong = inputLong("long:");
    if (elLong == 5)
    { output("Si"); }
    else
    { output(elLong); }

    double elDouble = inputDouble("double:");
    if (elDouble == 6)
    { output("Si"); }
    else
    { output(elDouble); }

    boolean elBoolean = inputBoolean("boolean:");
    if (elBoolean == true)
    { output("Si"); }
    else
    { output(elBoolean); }

    input("-- oprima [intro] para finalizar --");
  }

  //-----
  //-- IBIO - incluir métodos simplificados de entrada y salida --
  //-----
}
```

# APÉNDICE 3

---

## Símbolos de diagramas de flujo de sistemas

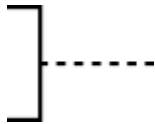
*Acción o proceso*



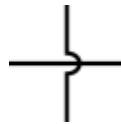
*Dispositivo de entrada o salida  
(descripción dentro)*



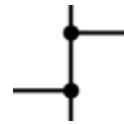
*Anotación*



*Líneas que se cruzan*



*Líneas que se juntan*



*Flujo de datos*



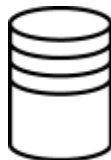
*Documento*



*Cinta*



*Disco*



*Almacenamiento en línea*



*Enlace de comunicación (dos  
sentidos si no se indica lo  
contrario)*



# APÉNDICE 4

---

## Símbolos de puertas lógicas

