INFORMÁTICA

Bandas de calificación de la asignatura

Nivel Superior

Nota final:	1	2	3	4	5	6	7
Puntuaciones:	0-13	14-26	27-38	39-50	51-63	64-75	76-100

Nivel medio

Nota final:	1	2	3	4	5	6	7
Puntuaciones:	0-15	16-30	31-41	42-52	53-63	64-74	75-100

Nivel Superior - Evaluación interna

Bandas de calificación del componente

Nota final:	1	2	3	4	5	6	7
Puntuaciones:	0-5	6-10	11-17	18-25	26-33	34-41	42-50

Rango e idoneidad del trabajo enviado

Debido a que ésta era la primera convocatoria de un nuevo conjunto de criterios para el dossier y un nuevo conjunto de aspectos de dominio, hubo inevitablemente algunos malentendidos e interpretaciones incorrectas. Este informe intentará clarificar qué se requiere en el dossier de Informática.

Es importante que los colegios sean conscientes de que los moderadores recibieron instrucciones de ser generosos en la interpretación de muchos de estos criterios y aspectos de dominio, aunque esto no ocurrirá en sesiones futuras.

El rango de temas elegidos fue bastante amplio, ya que se pretenden conceder los nuevos aspectos de dominio. Los alumnos, salvo los más preparados, siguen afrontando mal los juegos. Las mejores elecciones fueron, por lo general, aquellas que tenían un usuario claro y definido. Los alumnos menos preparados suelen elegir programas de bases de datos y poco definidos, suficiente para demostrar sus habilidades; asimismo, proporcionan la elección más directa del problema.

Como siempre, algunos alumnos se centraron en la solución y apenas discutieron el problema que intentaban resolver. A menudo, la interacción con los usuarios parece ser mínima o se reduce a un simple detalle, lo que dificulta la parte de análisis y diseño del dossier.

Aunque hay que reconocer que es un ejercicio de clase y que es poco probable que los programas finales de los alumnos se usen en el mundo real, la idea es que intenten simular un proceso de diseño real lo máximo posible.

Parece que los alumnos, con frecuencia, realizan las secciones de análisis y diseño después de haber escrito el programa. Esto es un inconveniente para los alumnos, ya que prestar una especial atención al análisis y el diseño puede reducir el tiempo y el trabajo necesarios para encontrar la solución, mientras que escribir la documentación a posteriori es una tarea inútil, que requiere mucho tiempo y que no tiene ningún valor como experiencia de aprendizaje.

Los nuevos aspectos de dominio relacionados con la POO causaron algunos problemas. Los alumnos la abordaron de manera trivial en muchas ocasiones, a menudo basándose en bibliotecas o rutinas de utilidad que forman parte del SDK de Java. Las secciones relevantes de la Guía de la asignatura (ver Tema 5) describen los conocimientos que deben tener los alumnos sobre herencia, polimorfismo y encapsulación. Esta cuestión se discute con profundidad en secciones posteriores de este informe.

Bastantes alumnos parecen no comprender otros factores de dominio nuevos, como el análisis sintáctico de un flujo de texto. Se ofrecerán ejemplos de ello.

La implementación de los TDA resultó suficiente o incluso correcta en muchos casos. Hubo varios ejemplos evidentes de plagio y otros no tan obvios. Los alumnos deben hacer referencia a cualquier código que se tome directa o indirectamente de otra fuente. Éste será el caso de los TDA, ya que es muy poco probable que un alumno escriba un TDA desde cero. Las fuentes de información deberían incluirse en la sección de diseño B1 Estructuras de datos; asimismo, el profesor debería asegurarse de que el alumno adapta el TDA al problema concreto y de que entiende todo el código que se incluye y que se requiere como aspecto de dominio.

Los alumnos deberían indicar claramente qué código están usando para un requisito de dominio y qué código no está escrito por ellos y se incluye para mejorar algunos aspectos de la solución. Los profesores deberían usar una revisión profesional y ser íntegros en este aspecto, ya que la diferencia entre el código adaptado y el que se toma prestado (aunque no se entienda totalmente) puede ser insignificante.

Las ejecuciones de prueba que incluían salida impresa fueron, de nuevo, un problema en muchos casos. La guía de la asignatura argumenta claramente que se debería incluir, como mínimo, una ejecución de prueba del programa. Esta prueba debería incluir datos normales y, siempre que sea posible, demostrar que el programa funciona como se pide. Por ejemplo, si se especifica como factor de dominio el añadir elementos a un archivo o una lista enlazada, debería haber una parte de la lista que muestre el registro que se esté añadiendo y que posteriormente se incluye en el conjunto de datos. Muchas ejecuciones de prueba sólo muestran qué ocurre cuando se introduce un dato erróneo.

Rendimiento alcanzado por los alumnos en cada uno de los criterios

Criterio A1

En muchos casos no se analiza el problema. Los alumnos estaban deseando demostrar qué hacía su programa, sin tener en cuenta por qué lo hacía. El punto de partida debería ser una afirmación clara del problema de información que se va a resolver. Este problema de información SÓLO se puede comprender adecuadamente hablando con el usuario o los usuarios que van a usar el sistema actual. Los alumnos que empiezan la solución sólo con una idea confusa del problema que se debe resolver acaban por realizar un trabajo bastante pobre en el dossier.

Se requiere un usuario real (no imaginario) para que los alumnos puedan tener los puntos de vista de una persona no técnica y obtener información de ella. No es imprescindible, aunque sí recomendable, que este usuario sea adulto, maduro y tenga un problema real, en vez de un compañero de clase o un hermano al que el alumno le resuelva el problema. En el primer caso, es probable que la calidad de las evidencias recopiladas sean más útiles para el alumno. Dicho esto, no todos los adultos maduros son buenos comunicadores, ni todos los jóvenes son incoherentes; los alumnos y los profesores deberían usar el mejor criterio.

En los dossieres vistos en esta convocatoria se omiten a menudo las pruebas de obtención de datos o resultan superficiales.
El usuario no necesita un problema que requiera un programa informático real, profesional o maravilloso. Lo ideal es que sea relativamente simple. Se debería disuadir a los alumnos de intentar crear soluciones prolijas o demasiado ambiciosas.

En este contexto, el usuario real es el encargado de sacar a los alumnos de su perspectiva limitada, desde la cual sólo desarrollan para ellos mismos. No hay recompensa (en términos del BI) para un programa de primera magnitud o que se vaya a usar posteriormente. Parte del proceso de análisis y diseño requiere limitar el ámbito de la solución a un tamaño alcanzable. En esta fase, las orientaciones del profesor son fundamentales.

La tentación de añadir funciones "atractivas" durante la codificación, que conlleva al incumplimiento de las fechas límite y a que surjan muchas dificultades en la programación, se podría evitar si los alumnos comprendieran que se trata de un ejercicio de clase, no de un producto comercial. Mientas que los alumnos insistan en intentar producir este tipo de programas, una buena estrategia de enseñanza es exigirles que cumplan los requisitos principales de un dossier del BI en primer lugar y que posteriormente añadan características complejas (si hay tiempo).

Criterio A2

Muchos alumnos no ofrecen una conexión clara entre el análisis del problema y los objetos, tal como requiere el criterio. Algunos alumnos escriben párrafos descriptivos, mientras que otros escriben enumeraciones, siendo estos últimos probablemente más eficaces. No es recomendable incluir un gran número de objetivos, por las razones que se discuten en A1.

Los objetivos deberían ser: específicos, perceptibles, factibles, realistas y restringidos en el tiempo. En este momento, el profesor y el alumno deberían convencerse de que es posible conseguir al menos 10 (y preferentemente 12) aspectos de dominio y tener una buena idea de cuáles serán. En esta fase, un plan de desarrollo ayudaría a los alumnos a crear un marco temporal realista para completar el trabajo que se debe hacer; esto no tiene que incluirse en el dossier, pero debería consultarse en un futuro, durante el desarrollo. Los profesores deberían hacer cumplir los plazos rigurosamente para asegurarse de que los alumnos obtienen la máxima recompensa por el trabajo que han hecho.

Los objetivos deberían estar relacionados explícitamente con el análisis. Argumentar como objetivo "Voy a organizar los CD" no tiene sentido a menos que esté claro qué se debe demostrar para probar que se ha alcanzado el objetivo.

Se indicaron objetivos, pero casi nunca se hizo referencia a ellos en las secciones posteriores. En las secciones posteriores se debería haber argumentado el objetivo, de forma que estuviera presente la evidencia que apoya el cumplimiento de dicho objetivo.

Los alumnos deberían evitar manejar un gran número de objetivos, ya que de esta forma se aumenta notablemente la necesidad de realizar pruebas y ejecuciones de prueba sin que se ilustren realmente aptitudes adiciones del alumno; sólo añade más volumen. Basta con unos 5 o 6 objetivos no triviales, bien descritos y alcanzables.

Otra sugerencia es escribir objetivos **centrados en el usuario** en lugar de objetivos **centrados en el programa**. Debido a que los alumnos están centrados en el punto de vista del programa, sus objetivos siempre están relacionados con el programa.

Los objetivos centrados en el usuario describen lo que conseguirá el usuario, no las "funciones" que proporcionará el programa. Debería ser más fácil comprobar los objetivos centrados en el usuario. Y es obvio que probar un objetivo centrado en el usuario requiere ejemplos impresos en lugar de leer

algunas líneas del listado de un programa. Por ejemplo, habría que considerar rescribir los objetivos de esta forma:

Orientados al programa	rescrito como	Orientados al usuario
El programa guarda títulos de CD en un archivo de datos.	→	El usuario puede guardar títulos de CD, recuperarlos y visualizarlos posteriormente.
El programa puede ordenar títulos de CD alfabéticamente.	→	El usuario puede visualizar los títulos ordenados alfabéticamente o cronológicamente (ordenados por fecha).

En algunos dossieres de la convocatoria actual se aceptaba la entrada de datos del usuario y su "almacenamiento" en una matriz, pero nunca en un archivo. Seguramente el usuario desearía conservar los datos de un día para otro, pero los programadores estudiantes pensaron, según parece, que bastaba con obtener los puntos del dominio de las matrices e ignorar la entrada/salida de ficheros.

Esto podría tener sentido en un prototipo, pero es difícil imaginarse escribir cualquier tipo de software de procesamiento de datos que pierda todos sus datos cuando se desconecte la máquina. De nuevo, un usuario real con necesidades reales y al que se consulte activamente podría identificar estos objetivos para el alumno. El alumno, como se argumenta anteriormente, no está obligado a hacer todo lo que desea el usuario, pero debería seleccionar cuidadosamente las 5 o 6 restricciones mencionadas anteriormente.

Criterio A3

Una vez que se identifican clara e inequívocamente los principales objetivos y límites de la solución propuesta para el problema, debería ser posible identificar los elementos principales de la solución, así como una planificación de la misma.

Esto lleva a un diseño modular inicial para un prototipo que se puede mostrar posteriormente al usuario para que lo comente. Los comentarios del usuario deberían documentarse, pero no hay que seguirlos siempre al pie de la letra, sobre todo si las sugerencias conducen a una solución demasiado extensa.

Un prototipo ayudará a que el alumno se forme una imagen clara del "producto" en su propia mente.

En esta convocatoria muchos alumnos no incluyeron un diseño inicial o información del usuario en el prototipo. A menudo se presentaron prototipos no funcionales, útiles, creados mediante varios paquetes gráficos.

Puede resultar beneficioso si un prototipo no funcional se escribe en Java, ya que los alumnos lo comprenderán mejor, lo cual resulta útil para el diseño de los algoritmos y, a la vez, satisface sus ansias de continuar y escribir código desde un primer momento. Esto, obviamente, no se debería llevar hasta el extremo. Un prototipo en Java con métodos para las principales funciones es un punto de partida muy útil para una posterior codificación.

El prototipo se debería usar para descomponer el programa en secciones lógicas desde el punto de vista del usuario. Esto conlleva, probablemente, elegir elementos de menú y campos de datos en un formulario. El diseño inicial puede estar más centrado en el programa, e incluir estructuras de datos y algoritmos. Si los alumnos se aproximan al "diseño inicial" y el "prototipo" desde estos dos puntos de vista diferentes, tiene sentido preguntarse si ambos están relacionados correctamente (criterios A3).

Criterio B1

En el Nivel Superior se espera alguna discusión sobre estructuras de datos alternativas, aunque éstas pueden ser bastante superficiales. Es más importante que los alumnos expliquen cómo funcionarán realmente las estructuras de datos que han seleccionado con los datos del dominio del problema.

Como en anteriores versiones de los criterios del dossier, los ejemplos son extremadamente valiosos y necesarios para obtener las máximas puntuaciones. Deberían usar datos del problema que el alumno está intentando resolver. El uso de diagramas generales sobre el funcionamiento abstracto de listas enlazadas o árboles no satisface este criterio.

Los alumnos pueden usar estructuras de datos de cualquier fuente, siempre que se reconozca dicha fuente. La estructura de datos sólo se puede usar para solicitar el dominio si se discute en esta sección y, por lo general, existe la necesidad de adaptar dicha estructura a la solución propuesta.

Los alumnos deben programar sus propios TDA (por ejemplo, listas enlazadas). El uso de Java.util LinkedList no obtendrá los puntos correspondientes al dominio de los TDA. Además, si sólo se copia una biblioteca LinkedList de otro autor tampoco se obtendrá los puntos del dominio, y cuando no se indica claramente constituye un plagio.

En el Nivel Superior, el principal problema en este sentido es el código para implementar listas enlazadas, aunque en esta convocatoria también se descubrió que algunos copiaron el código de los TDA.

Criterio B2

Los algoritmos deberían estar lo suficientemente detallados como para que se pueda crear la solución a partir de ellos. No tienen que ser perfectos o completos, pero deberían estar presentes todos los módulos esenciales identificados en B3 o A3, ya que sigue siendo un requisito del programa de Informática del BI.

En esta convocatoria, muchos alumnos no proporcionaron suficientes algoritmos con el nivel de detalle necesario (precondiciones, postcondiciones y parámetros) y fueron penalizados por ello. En algunos colegios se vieron algoritmos en hojas de codificación, lo cual da una sensación de claridad. En otros colegios se presentaron algoritmos muy parecidos a la sintaxis de Java, y esto es correcto. También se acepta que los profesores usen un pseudocódigo de su elección o que hayan creado ellos mismos.

Como siempre, algunos alumnos presentaron código que derivaba claramente de la solución y fueron penalizados en consecuencia. Los moderadores detectan esta práctica muy fácilmente.

Criterio B3

Tal como se indica en la Guía de la asignatura, en esta convocatoria se aceptan varios tipos de presentaciones, que suelen depender del enfoque utilizado. Sin embargo, todas estas variaciones deberían mostrar las conexiones entre módulos o clases y algoritmos y estructuras de datos. A menudo, esta sección muestra un "gran cuadro" de la solución propuesta, tanto si está realizada como diseño modular descendente, diseño orientado a objetos o enfoque ascendente.

Se usaron varias presentaciones, pero el fallo más común fue no identificar las conexiones con las estructuras de datos y los algoritmos.

Criterio C1

En su mayoría respondieron bien y usaron una sangría correcta y unos nombres identificadores claros. Sin embargo, en muchos listados resultó difícil averiguar dónde empezaban y terminaban las clases y

determinar la relación entre las mismas. Esto es más difícil cuando la solución se desvía del diseño, lo cual está permitido.

A ser posible, una clase debe comenzar en una nueva página o estar claramente separada de las clases precedentes, así como contar con una cabecera que describa su función. Es realmente útil contar con una lista de clases y los números de página al principio del código. Algunos IDE proporcionan diagramas de relación, que también resultan útiles para navegar a través del código.

Los alumnos deben asegurarse de que los comentarios son legibles si se escribe en blanco y negro; algunos IDE presentan los comentarios en gris y son difíciles de leer.

Criterio C2

Los alumnos realizaron correctamente esta sección en líneas generales. Muchos eligieron usar un enfoque de resumen, haciendo referencia a la salida impresa que se proporcionaba como parte de la sección D1. Esto funcionó mejor que el texto descriptivo sin ilustraciones o ejemplos específicos.

Esta sección debería hacer referencia a los criterios A2 explícitamente, y esto no se hizo en muchos casos. De nuevo, la idea que subyace a los criterios de facilidad de uso de A2 es escribirlos de forma que en esta sección se pueda demostrar que se han alcanzado. De ahí que sea difícil demostrar una "correcta facilidad de uso", mientras que resulta más fácil demostrar que "todos los menús tendrán una opción de ayuda".

Criterio C3

Respondieron correctamente en esta sección, que no ha cambiado desde el último programa, salvo que los criterios son un poco más específicos. Los mejores alumnos proporcionan ejemplos de código de gestión de errores, junto con referencias a salidas impresas en las que se muestran evidencias de la captura de dichos errores.

A menudo los alumnos realizan tablas de datos de prueba, que se incluyen en esta sección o en D1 y a las que posteriormente hacen referencia en esta sección. Ambos enfoques suelen recibir muy buenas puntuaciones y demuestran que los alumnos se organizan muy bien.

Criterio C4

Hubo unos pocos casos en los que los profesores otorgaron puntuaciones elevadas a pesar de que los alumnos no proporcionaron salida para el criterio D1. Si no hay pruebas de que un programa funciona, la nota para ese criterio es 0. Ciertamente, no basta con que el profesor argumente que el programa funciona adecuadamente, deben ofrecerse evidencias en todos los casos.

Para no llegar a esta situación, los profesores deberían dejar algún tiempo antes de la entrega final para que los alumnos realicen ejecuciones de pruebas y muestren aquellas partes de la solución que funcionan. Puede que los alumnos encuentren útil realizar algunas ejecuciones de prueba parciales durante la codificación y tenerlas como algo a lo que recurrir. Éstas se podrían incluir en un blog de desarrollo o un registro escrito. Esto resulta más fácil, obviamente, si los alumnos son capaces de seleccionar un problema que se pueda resolver mediante un desarrollo incremental (la mayoría de problemas lo permiten).

La salida a la que se haga referencia en esta función también debería estar relacionada con los objetivos argumentados en A2, que se deberían argumentar de nuevo aquí, junto con cualquier evidencia que ayude a demostrar que se han conseguido dichos objetivos.

Criterio D1

En la situación ideal, las pruebas se ejecutan sistemáticamente, están bien planificadas y se ejecutan de forma estructurada. Las anotaciones deberían indicar qué parte de la solución (tal como se esboza

en los objetivos) se está probando. No hay que repetirlo en la sección C4, pero debería aparecer en C4 o D1. Es factible considerar una sección que incorpore C4 y D1 bajo una sección titulada "Salida impresa con anotaciones", tal como se indica en la Guía de la asignatura, en el formulario 5/PDCS.

Esta sección es difícil de realizar eficazmente a menos que se enumeren objetivos claros y específicos en la sección A2.

Esta sección tiene como objetivo demostrar el funcionamiento del programa en situaciones "normales"; muchos alumnos pusieron demasiado énfasis en la detección de errores y dejaron de lado todo lo demás. Cuando sea posible, se deben incluir ejecuciones que demuestren que las secciones del código para las que se reclama el dominio funcionan realmente. Un ejemplo común en el NS sería el correcto funcionamiento de las inserciones y eliminaciones en otros TDA.

Criterio D2

Se espera que el alumno reflexione sobre los logros del proyecto, aunque no debería olvidarse de todo lo demás. Tal como se indica en la introducción, el proceso es más importante que la complejidad de la aplicación que se esté creando. Por tanto, la eficacia de la solución debería discutirse sinceramente; cómo funciona para el usuario (en relación con los criterios argumentados en A2) en lugar de su calidad en términos de codificación (esto se puede discutir desde el punto de vista de la eficacia).

Además de cualquier mejora posible, el alumno debería examinar el proceso global que se ha llevado a cabo. Para ello, se puede ayudar de notas tomadas durante el proyecto, ya sea electrónicamente (por ejemplo, un blog) o en un cuaderno. Los resultados más valiosos de un dossier de Informática del BI podrían ser los defectos del proceso de diseño y cualquier otra lección que haya aprendido el alumno. En muy pocas ocasiones los alumnos tuvieron en cuenta alternativas al propio proceso de diseño.

Criterio D3

Parece que muchos profesores no tienen en cuenta aquello de "ejecutar el programa en otra máquina sin usar el IDE" a la hora de conceder 3 puntos en esta sección. Por lo general, los mejores alumnos usan instrucciones claras e ilustradas en los programas. Los alumnos menos preparados muestran evidencias de haberse precipitado en esta sección. Con una planificación detallada de las ejecuciones de prueba es posible realizar capturas de pantalla que proporcionen documentación para D1 y D3.

Criterio E

Da la impresión de que la nota holística se concedió justamente y sólo unos pocos colegios optaron por conceder los 3 puntos máximos, como se podría haber temido. Con frecuencia, los profesores tuvieron la amabilidad de explicar el porqué de la concesión de esta nota, lo cual ofrece confianza a los moderadores sobre la integridad de los colegios y los profesores.

Aspectos de dominio

En la convocatoria de 2006 se puso de manifiesto alguna variación en la interpretación de los aspectos de dominio según los profesores, colegios y alumnos. En concreto, no se conocen bien algunos aspectos o no se entiende que aspectos como la encapsulación y la herencia deben usarse tal como se define en la Guía de la asignatura, Tema 5.

Este informe está diseñado para ilustrar la forma en que los alumnos deberían enfocar los aspectos de dominio y cómo deberían puntuar los profesores.

Debemos insistir, no obstante, en que hay muchas formas de programar y de conseguir el dominio. Ninguno de los ejemplos que se presentan en esta sección deben tomarse como plantilla para que los profesores o los alumnos la apliquen a sus propios problemas.

Tal vez el mayor problema con los aspectos de dominio sea la frecuente falta de documentación o las inexactitudes en la misma. En estas carencias se incluyen varios aspectos como errores administrativos, falta de salida impresa y ausencia de discusiones sobre la necesidad de la solución para algunos aspectos de dominio.

Los números de página escritos por los estudiantes son a veces incorrectos, y el profesor debería revisarlos cuidadosamente antes de enviar el dossier y el formulario 5/PDCS. No se espera que los moderadores revisen las páginas con el listado de código para encontrar factores de dominio relevantes.

Los alumnos y los profesores, si lo prefieren, pueden hacer referencia a listas más completas de aspectos de dominio en el dossier. La ubicación de dichas páginas debería estar claramente indicada en el formulario 5/PDCS.

Si el profesor no completa los formularios adecuadamente, se podría devolver los dossieres a los colegios para su corrección, lo que produciría retrasos considerables en los resultados del diploma de los alumnos.

Documentación de aspectos de la fase de diseño

Entre los problemas típicos se encuentran los aspectos de dominio que no se discuten en la sección de estructuras de datos o que no se identifican algoritmos o módulos como manipuladores de estructuras de datos. Todavía hay otro aspecto que aparece misteriosamente en el código. A menudo se incluye una ordenación cuando aparentemente no hay necesidad de ello, y algunos alumnos reclaman el dominio de la recursividad (ordenación rápida, ordenación por fusión) en el Nivel Superior.

En ocasiones aparecen TDA en el listado que reciben aspectos de dominio, cuando realmente no se han descrito completamente. La guía de la asignatura indica claramente que este uso se considera trivial y no puede recibir aspectos de dominio.

Uso de algoritmos estándares

Se permite el uso de algoritmos estándares, incluso se anima a que se incluyan en el dossier. Cualquier algoritmo estándar de alguna fuente pública se podría reconocer en condiciones normales. No obstante, los moderadores deben asegurarse de que el alumno comprende dichos algoritmos. Esto se puede demostrar de varias formas:

- El alumno discute las modificaciones necesarias para que el algoritmo estándar cumpla sus propios requisitos o explica claramente la función de dichos algoritmos en el contexto de su problema/solución concretos.
- El alumno demuestra al profesor una comprensión suficiente, que lo confirma realizando observaciones en 5/PDCS o 5/IACS y firmando los formularios.

Al igual que con otra materia del BI, copiar código de un libro, sitio Web u otra fuente (como por ejemplo otro compañero) se considera plagio, y con toda probabilidad obtendrán una nota de 0 puntos o un reprobado/suspenso en el componente y, por tanto, en la asignatura.

Bibliotecas y utilidades de Java

Java posee amplias colecciones y rutinas de biblioteca incorporadas, y hay disponibles otras muchas de terceros. Aunque se podrían usar en las soluciones, ello no implica que reciban aspectos de dominio.

Esto se aplica en concreto a estructuras como listas enlazadas, tablas hash y otros TDA.

No se permite que los alumnos soliciten el dominio cuando usen métodos de biblioteca dentro de clases triviales y métodos creados por ellos mismos.

Ejemplos ilustrativos y comentarios adicionales

Los aspectos siguientes deben estar documentados en la sección de diseño y, cuando corresponda, debe demostrarse que funcionan mediante salida impresa.

Adición de datos a una instancia de la clase RandomAccessFile mediante la manipulación directa del puntero de archivo usando el método seek

Algunos ejemplos triviales de esto son:

- Desplazarse al inicio del archivo –seek(0)- y escribir un flujo de datos al mismo.
- Desplazarse al final del archivo –seek(file.length) y agregar datos al mismo.

Si se va a usar un tipo de archivo de acceso aleatorio en el dossier es necesario hacerlo de esta forma. Los datos deben ser de tipo primitivo (por ejemplo, int), aunque en ese caso será necesario realizar algún cálculo para escribir el entero en una posición específica del archivo. Escribir una matriz de enteros, un valor tras otro, no satisface este aspecto.

Eliminación de datos de una instancia de la clase RandomAccessFile mediante la manipulación directa del puntero de archivo usando el método seek

Los tipos primitivos o los objetos se pueden marcar para su eliminación. Las partes del archivo marcadas de esta forma deberían reutilizarse, estar disponibles para operaciones de "deshacer eliminación" o compactados en algún momento para que no afecten especialmente al acceso al archivo. Si no se hace así, debido a que no se considera necesario para el problema concreto, el alumno debería discutirlo en la sección B1.

Si el archivo se quiere conservar como archivo ordenado, los componentes de dicho archivo deberían poder eliminarse mediante la mezcla de registros (manipulando el puntero de archivo) en función del elemento que se va a eliminar.

Sobrescribir un registro existente con un registro modificado podría reclamarse como eliminación para este aspecto. Sin embargo, la modificación podría no hacerse en una estructura externa (como una lista o un árbol) y sí, posteriormente, escribir secuencialmente el contenido de esta estructura en el archivo.

Búsqueda de un dato específico en una instancia de la clase RandomAccessFile

Esto se podría combinar con la adición o la eliminación, según convenga.

Recursividad

La recursividad debe documentarse e implementarse correctamente. Un uso trivial de la recursividad es una solución iterativa que funcione igual de bien (por ejemplo, contar nodos o realizar una búsqueda en una lista enlazada).

No se puede exigir el dominio de la recursividad cuando se use un algoritmo estándar de ordenación (ordenación por fusión o rápida) que no se haya documentado en la fase de diseño. Normalmente, se espera contar con esta documentación para justificar el uso de estos algoritmos en contraposición a un simple O(n^2), por ejemplo. Para estas ordenaciones se debería discutir la distribución probable de datos en la lista no ordenada. El profesor también debería confirmar, en sus notas de 5/PDCS o 5/IACS, que el alumno comprende los algoritmos.

Un uso erróneo común de la recursividad se produce en las rutinas de entrada:

```
public int obtenerPorcentaje()
{
  int entero = inputInt("Introduzca un número entre 1 y 100");
  if ( (entero < 0) || (entero > 100) )
  {
     obtenerPorcentaje();
  }
  return entero;
}
```

Este ejemplo es trivial, ya que el método se podría haber programado de forma iterativa e igual de bien. Es un uso incorrecto de la recursividad, ya que no hay definida claramente una condición de terminación. Tenga en cuenta que este código no funciona (aunque pueda parecerlo si se añade una instrucción return después de la llamada a `getPercent()`).

Este ejemplo tampoco ilustra la recursividad:

```
void escribir()
{
  for (int i = 0; i < 10; i++)
    escribir(a[i]);
}
void escribir(int num)
{
  System.out.println(num);
}
```

Fusión de dos o más estructuras de datos ordenadas

En este caso, el error más común es tener una ordenación por fusión y posteriormente solicitar el dominio de este aspecto. La fusión requiere dos conjuntos de datos previamente ordenados.

También hay que distinguir entre fusión y la adición de datos a una estructura de tipo lista.

Polimorfismo

Los tres aspectos de dominio de la POO (polimorfismo, herencia y encapsulación) se incluyen para los alumnos/profesores que prefieren adoptar un enfoque orientado a objetos adecuado para el proceso de diseño. Por tanto, se arriesgan a que se considere trivial un dossier que adopte un enfoque de diseño estructurado y simplemente incluya las palabras clave extends o implements. A veces, sin embargo, es adecuado recompensar estos aspectos en otros enfoques en los que el alumno puede explicar la necesidad del aspecto.

El polimorfismo en Java se suele implementar mediante interfaces o herencia. En una subclase o una clase que implemente una interfaz dada, los métodos de la clase pueden sobrescribir los métodos de la clase padre para producir diferentes comportamientos adecuados para dicha clase.

Por tanto, implementar un método toString() que existe en la clase Object (de la cual derivan todas las clases Java) es técnicamente polimorfismo.

De igual forma, cuando se implementa la interfaz ActionListener, se requiere que el método la clase que implementa la interfaz implemente actionPerformed(ActionEvent). Ésta no es una decisión de diseño que deba tomar el alumno.

Los alumnos no pueden usar código de biblioteca y solicitar el dominio, no han escrito ninguna de esas superclases. Por tanto, las implementaciones anteriores de polimorfismo se consideran triviales.

Para obtener el dominio del polimorfismo los alumnos deben haber diseñado, documentado e implementado la superclase/interfaz y la subclase o clase que implementa la interfaz. El diseño OO debería discutirse adecuadamente en la fase B. En este caso podría ser adecuado no seguir el orden recomendado de criterios que se indica en la página 57 de la Guía de la asignatura.

También se puede usar el polimorfismo para hacer referencia a construcciones relativamente simples, como la sobrecarga de métodos (incluyendo constructores). Un estudiante podría reclamar este ejemplo como muestra de polimorfismo si:

- La modificación en los parámetros del método conlleva un cambio en el comportamiento adecuado en un algoritmo.
- El requisito para el enfoque se documenta correctamente en la sección B.

Para ilustrar el primer punto, consideremos un método ordenar( ) que se podría usar para clasificar la lista en orden alfabético. Podríamos usar el polimorfismo para implementar métodos como:

- ordenar(boolean dirección) donde true/false determinen el orden ascendente/descendente y
- ordenar(int inicio, int fin) para ordenar sólo una parte de la lista.

Al igual que con otros aspectos, es responsabilidad del alumno demostrar que conoce por qué usa un aspecto concreto, así como argumentar los beneficios que dicho aspecto aporta a su código. Nunca se puede aceptar que se incluyan estos aspectos sin describir su necesidad en la solución al problema.

Herencia

De igual forma, la herencia no puede extender sólo a una clase incorporada o de biblioteca y esperar que se conceda el dominio de este aspecto "automáticamente". Una clase que extiende a Object es trivial e innecesaria.

Para conseguir la herencia, el alumno debe documentar y mostrar cómo la subclase usa los métodos y los miembros dato de la clase padre. Por ejemplo, se acepta que el alumno construya elementos Panel, JPanel, JFrame y TextField "especiales", siempre y cuando documenten la necesidad de la nueva clase derivada.

El uso de setSize(int, int), setVisible(boolean) o métodos de utilidad similar es trivial (y necesario para que el código realmente muestre resultados visibles).

Sin embargo, los siguientes serían ejemplos correctos (siempre y cuando los usen otras clases y estén correctamente documentados):

- Creación de clases Exception personalizadas.
- Creación de objetos de interfaz gráfica de usuario personalizados.
    - J/Button con eventos de ratón incorporados.
    - J/TextField que manejen sólo tipos de datos definidos estrictamente (números, nombres, valores booleanos).
    - J/Panel que tengan una matriz preestablecida de botones estándar (Cerrar, cancelar, ayuda) u otras funciones.

Por tanto, los alumnos de Nivel Superior que no estén usando un enfoque orientado a objetos aún pueden implementar este aspecto si lo desean.

Encapsulación

La mayoría de los alumnos de Nivel Superior deberían poder implementar una clase que tuviera métodos y miembros dato. Los miembros dato deberían ser, por lo general, privados o protegidos, de forma que otras clases no accedan a los miembros datos a través de la notación punto, sino de métodos adecuados en las clases encapsuladas. De nuevo, el alumno debería ser capaz de explicar por qué estas técnicas de "ocultación de datos" son adecuadas para las clases que están definiendo para resolver el problema que han identificado.

Análisis sintáctico de un fichero de texto u otra corriente de datos

El uso de clases incorporadas tipo wrapper, como Integer y Double con los métodos parseInt(String) y parseDouble(String) se considera trivial.

Un flujo de datos de entrada, como el de un archivo, el teclado o la red, que este formado por bytes de caracteres que representan texto, debe descomponerse en partes separadas de alguna forma. Se acepta el uso de String.split() o java.util.StringTokenizer. Por tanto, los componentes deben reconocerse como pertenecientes a un conjunto particular, cada uno con unos requisitos de procesamiento concretos. Obtener líneas de un archivo de texto csv o formatos similares y crear instancias similares a un registro se considera un uso aceptable. En esta conversión de tipos es correcto usar Integer.parseInt(). Los formatos y el proceso de interpretación deberían discutirse en las estructuras de datos y algoritmos de la fase B.

Implementación de una estructura de datos jerárquica compuesta

Esto no ha cambiado con respecto a la Guía de la asignatura anterior, y parece que los alumnos lo entienden razonablemente bien. En B1 se debe describir la necesidad de la estructura y los datos de ejemplo que almacenará.

Uso de alguno de los cinco factores de dominio del nivel medio

No se requiere documentación adicional, salvo indicar dónde aparecen estos factores en la lista.

12-15 Se pueden otorgar un máximo de 4 aspectos por la implementación de tipos de datos abstractos (TDA)

Éste es otro aspecto de dominio que se ha introducido para proporcionar flexibilidad en el diseño de soluciones y para evitar una aproximación demasiado normativa a la construcción del programa.

Normalmente, el alumno elegirá un TDA que sea adecuada para el problema particular y el conjunto de datos y explicará el porqué. Es una práctica común, aunque no aceptable, trabajar con dichos TDA en la fase B de forma teórica, en lugar de indicar cómo cumple el TDA los requisitos especificados.

Algunos TDA son más complejos que otros, una lista o una pila son bastante simples de implementar y los alumnos deberían, por tanto, estar preparados para explicar por qué se ha elegido una estructura concreta. Al igual que con la ordenación, se pueden usar algoritmos estándares extraídos de un libro o sitio Web, pero en las fuentes debería reconocerse debidamente. La salida impresa debe demostrar que estos TDA funcionan; asimismo, el profesor necesita escribir una nota en 5/PDCS o 5/IACS confirmando que el alumno entiende el código que usa.

Cuando uno o más alumnos usen el mismo TDA el profesor debe realizar una anotación en relación con cada alumno; asimismo, los alumnos deben justificar individualmente el uso de la estructura de datos (que será diferente para diferentes problemas).

Existe el riesgo de que al adoptar un enfoque "tipo plantilla" por el cual todos en la clase tengan un problema similar con una solución similar y usando el mismo TDA surja la duda del plagio. Por

tanto, se debe animar a los alumnos a que "planifiquen detenidamente" el diseño para sí mismos. En el dossier está prohibido trabajar en equipo.

Los alumnos pueden obtener algunos puntos por los TDA correctamente documentados e implementados y cuyo funcionamiento, mediante salida impresa, se demuestre. Estos TDA, que sólo funcionan parcialmente, podrían obtener los 4 puntos (aunque parece una forma difícil de conseguirlo).

No basta con que los alumnos usen una clase de biblioteca con sus métodos asociados en su estructura trivial de clase. Por ejemplo:

```
import java.util.LinkedList;
public class ListaTrivial
{
  // ejemplo de un TDA trivial
  private LinkedList miLista = new LinkedList();

  public static void main(String[] args)
  {
    new ListaTrivial();
  }
/**
 * Constructor para objetos de la clase ListaTrivial
 */
  public ListaTrivial()
  {
    super();
  }

  public void agregarAlInicio(Object o)
  {
    miLista.addFirst(o);
  }
  public void agregarAlFinal(Object o)
  {
    miLista.addLast(o);
  }
  public boolean isEmpty()
  {
    return miLista.isEmpty();
  }
}
```

Hay que tener en cuenta que una pila, cola o cola inversa trivial se pueden construir fácilmente mediante LinkedList u otras clases de java.util.

Un problema frecuente es presentar insuficientes ejecuciones de prueba para mostrar que un programa puede escribir y leer de un archivo de datos de acceso aleatorio u otro TDA y puede actualizar estas estructuras. Esto requiere con frecuencia más de una ejecución para que se registre.

Todos estos aspectos deben estar documentados en la sección de diseño y, cuando corresponda, debe demostrarse que funcionan mediante salida impresa.

Recomendaciones para la enseñanza de alumnos futuros

Muchas recomendaciones serán evidentes a partir de la sección anterior sobre criterios específicos; sin embargo, todas las secciones del dossier deberían beneficiarse de que los alumnos se aseguran que su elección del tema:

- Incluye un usuario concreto.
- Implica algo que entiende.
- Se puede resolver con el lenguaje Java.
- Es aprobado por el profesor.
- Tiene el potencial de satisfacer un número de aspectos de dominio del NS suficientes.
- Es investigado y analizado ANTES de empezar a escribir el programa.

Al igual que en sesiones anteriores, muchos profesores no completan el reverso del Formulario 5/IACS y no incluyen comentarios con los ejemplos enviados. Los comentarios ayudan al moderador a evaluar la nota otorgada por el profesor. El moderador desea confirmar la nota concedida por el profesor siempre que sea razonable hacerlo.

En algunos casos parece que los profesores no concedieron factores de dominio aun habiendo evidencias en dossier de que se habían alcanzado. El moderador, por lo general, corroborará la opinión del profesor, que es el más adecuado para otorgar esos puntos y no modificará la nota.

Los comentarios del profesor y una adecuada documentación por parte del alumno podrían evitar penalizaciones en los casos en los que no se entendieron correctamente los aspectos de dominio y, por tanto, no se obtuvieron cuando podrían haberse conseguido.

Se han cometido algunos errores al calcular la nota final en el Formulario 5/IACS. Los profesores deberían leer atentamente las instrucciones en el Vademécum a la hora de calcular y registrar las notas.

Los profesores deberían incentivar a los alumnos a mostrar el dominio en todos los aspectos, ya que pueden perder muchos puntos si no lo hacen.

La discusión en la evaluación debería realizarse desde el punto de vista de la programación, no desde los logros personales como programadores.

Los profesores deben discutir los criterios y directrices de evaluación con los alumnos antes de comenzar el dossier de trabajo personal. Los alumnos deben tener claro lo que cada criterio requiere de ellos. En el Nivel Superior, los profesores deberían tener cuidado en esbozar los aspectos de dominio requeridos y discutir el uso trivial frente al no trivial.

El desarrollo del dossier debería seguir las fases siguientes:

- análisis
- diseño
- desarrollo (codificación)
- documentación.

En concreto, tal como se advierte anteriormente, el análisis y el diseño no deberían completarse después de haber finalizado la solución, ya que sería una pérdida de tiempo y esfuerzo por parte del alumno.

Una estrategia excelente para la enseñanza es insistir a los alumnos en la realización de las secciones A y B antes de llevar a cabo ningún desarrollo. Esto no sólo creará una base sólida para una mejor calificación en el dossier, sino que también mejorará el desarrollo de la solución real del alumno. En términos matemáticos, teniendo en cuenta que estas secciones valen 24 puntos, los alumnos podrían haber conseguido una puntuación de 4, sin tener que alcanzar la perfección en los criterios posteriores para obtener una calificación final elevada (6/7). No obstante, esto es extremadamente difícil de conseguir a menos que se curse el programa en dos años.

Los moderadores podrán no confirmar los aspectos de dominio y los alumnos serán penalizados si:

- No se documentan adecuadamente los aspectos de dominio en 5/PDCS (o cualquier otra sección del dossier).
- No se documentan los aspectos de dominio en la Sección B.
- No se documentan los aspectos de dominio en la sección D1 (si procede).
- El alumno utiliza algoritmos de un libro, biblioteca de utilidades o sitio Web.
- El profesor no ha confirmado la comprensión de algoritmos estándar por parte del alumno.

Nivel Medio - Evaluación interna

Bandas de calificación del componente

Nota final:	1	2	3	4	5	6	7
Puntuaciones:	0-6	7-13	14-20	21-27	28-35	36-42	43-50

Rango e idoneidad del trabajo enviado

Debido a que ésta era la primera convocatoria de un nuevo conjunto de criterios para el dossier y un nuevo conjunto de aspectos de dominio, inevitablemente hubo algunos malentendidos e interpretaciones incorrectas. Este informe intentará aclarar qué se requiere para el dossier de Informática en el Nivel Medio.

Es importante que los colegios sean conscientes de que los moderadores recibieron instrucciones de ser generosos en la interpretación de muchos de estos criterios y aspectos de dominio, aunque esto no ocurrirá en sesiones futuras.

La presentación del dossier fue variable. El formato se indica en la página 54 de la Guía de la asignatura, aunque algunos alumnos decidieron no seguirlo. Mientras que la presentación fue, en muchos casos, correcta, la organización y la estructura resultaron en ocasiones pobres. Para ser justos con los alumnos, parece que no se compartió el contenido de la guía con ellos. Algunos colegios y alumnos enviaron el trabajo usando el formato antiguo, lo que inevitablemente condujo a la pérdida de puntos.

Algunos alumnos incluyeron elementos en los apéndices que deberían haber ubicado en otra sección. Por ejemplo: comentarios del usuario en relación con el prototipo y capturas de pantalla sin etiquetar y generadas, aparentemente, de forma aleatoria, que podrían formar parte de las ejecuciones de prueba o la documentación del usuario. Realmente, sólo es necesario incluir información del usuario en un

apéndice independiente, cuando dicha información sea voluminosa. Los resultados de la aplicación individual y similares pueden incluirse perfectamente en la sección A1.

El rango de temas elegidos fue bastante amplio, ya que se pretenden conceder los nuevos aspectos de dominio. Los alumnos, salvo los más preparados, siguen afrontando mal los juegos. Las mejores elecciones fueron, por lo general, aquellas que tenían un usuario claro y definido. Algunos colegios enviaron dossieres que no demostraban dominio de al menos 10 aspectos, aunque las soluciones podrían haberlo conseguido potencialmente, lo que indica una planificación pobre.

Los alumnos menos preparados suelen elegir programas de bases de datos, poco definidos, y suficientes para demostrar sus habilidades; asimismo, proporcionan la elección más directa del problema. Se presentaron varias simulaciones para esta convocatoria y, en líneas generales, resultaron ser temas adecuados.

Al igual que en el NS, algunos alumnos se centraron en la solución y apenas discutieron el problema que intentaban resolver. A menudo, la interacción con los usuarios parece ser mínima o se reduce a un simple detalle, lo que dificulta la parte de análisis y diseño del dossier.

Aunque hay que reconocer que es un ejercicio de clase y que es poco probable que los programas finales de los alumnos se usen en el mundo real, la idea es que intenten simular un proceso de diseño real lo máximo posible.

Parece que los alumnos, con frecuencia, realizan las secciones de análisis y diseño después de haber escrito el programa. Esto es un inconveniente para los alumnos, ya que prestar una especial atención al análisis y el diseño puede reducir el tiempo y el trabajo necesarios para encontrar la solución, mientras que escribir la documentación a posteriori es una tarea inútil, que requiere mucho tiempo y que no tiene ningún valor como experiencia de aprendizaje.

En muchos casos faltaban ejecuciones de prueba con salida impresa. La Guía de la asignatura argumenta claramente que se debería incluir, como mínimo, una ejecución de prueba del programa. Esta prueba debería incluir datos normales y, siempre que sea posible, demostrar que el programa funciona como se pide. Por ejemplo, si se especifica como factor de dominio el añadir elementos a una matriz, debería haber una parte de la lista que muestre el registro que se esté añadiendo y que posteriormente se incluye en el conjunto de datos. Muchas ejecuciones de prueba sólo muestran qué ocurre cuando se introduce un dato erróneo.

Cuando el programa tiene un defecto fatal, por ejemplo si no compila o entra en un bucle infinito, no se puede obtener salida de ejemplo y la consecución de los aspectos de dominio es dudosa.

Para un moderador es difícil distinguir este tipo de casos de aquellos en que hay una solución que funciona pero de la que no se proporciona salida de ejemplo por alguna otra razón. Aunque los moderadores deberían realizar un esfuerzo para evaluar cuidadosamente el listado de código, no debería dejarse en manos de los colegios. Los profesores deberían comentar explícitamente tales casos usando el formulario 5/PDCS que se proporciona.

Los alumnos deberían beneficiarse de las oportunidades que ofrece el lenguaje y los distintos IDE gratuitos para estructurar los listados de código de forma más inteligible. Los diagramas de clases o las tablas de contenido permiten tanto a los profesores como a los moderadores obtener un buen resumen de la solución y ayudan en la documentación de los aspectos de dominio.

Rendimiento alcanzado por los alumnos en cada uno de los criterios

Criterio A1

En muchos casos no se analiza el problema. Los alumnos estaban deseando demostrar qué hacía su programa, sin tener en cuenta por qué lo hacía. El punto de partida debería ser una afirmación clara del problema de información que se va a resolver. Este problema de información SÓLO se puede comprender adecuadamente hablando con el usuario o los usuarios que van a usar el sistema actual. Los alumnos que empiezan la solución sólo con una idea confusa del problema que se debe resolver acaban por realizar un trabajo bastante pobre en el dossier.

Se requiere un usuario real (no imaginario) para que los alumnos puedan tener los puntos de vista de una persona no técnica y obtener información de ella. A menudo, la evidencia de la obtención de datos se omite o es superficial. A menudo, los alumnos reducen la obtención de datos a marcar un recuadro y nunca se discute o se explica posteriormente.

El usuario no necesita un problema que requiera un programa informático real, profesional o maravilloso. Lo ideal es que sea relativamente simple. Se debería disuadir a los alumnos de intentar crear soluciones prolijas o demasiado ambiciosas.

En este contexto, el usuario real es el encargado de sacar a los alumnos de su perspectiva limitada, desde la cual sólo desarrollan para ellos mismos. No hay recompensa (en términos del BI) para un programa de primera magnitud o que se vaya a usar posteriormente. Parte del proceso de análisis y diseño requiere limitar el ámbito de la solución a un tamaño alcanzable. En esta fase, las orientaciones del profesor son fundamentales.

La tentación de añadir funciones “atractivas”, que conlleva al incumplimiento de las fechas límite y a que surjan muchas dificultades en la programación, se podría evitar si los alumnos comprendieran que se trata de un ejercicio de clase, no de un producto comercial. Mientas que los alumnos insistan en intentar producir este tipo de programas, una buena estrategia de enseñanza es exigirles que cumplan los requisitos principales de un dossier del BI en primer lugar y que posteriormente añadan características complejas (si hay tiempo).

Criterio A2

Muchos alumnos no ofrecen una conexión clara entre el análisis del problema y los objetos, tal como requiere el criterio. Algunos alumnos escriben párrafos descriptivos, mientras que otros escriben enumeraciones, siendo estos últimos probablemente más eficaces.

En este nivel el número de objetivos debería ser reducido, 6 u 8 es lo ideal. Un número mayor sólo añade volumen a la información sin demostrar capacidades adicionales. Por ejemplo: si se eligen 12 objetivos se doblará el número de pruebas que se han de realizar, la salida necesaria para demostrar la corrección, las instrucciones para el usuario y hará mucho menos probable que criterios como C2 y C3 consigan la máxima puntuación. Por tanto, la elección de objetivos requiere un razonamiento cuidadoso.

Los objetivos deberían ser: específicos, perceptibles, factibles, realistas y restringidos en el tiempo. En este momento, el profesor y el alumno deberían convencerse de que es posible conseguir al menos 10 (y preferentemente 12) aspectos de dominio y tener una buena idea de cuáles serán. En esta fase, un plan de desarrollo ayudaría a los alumnos a crear un marco temporal realista para completar el trabajo que se debe hacer; esto no tiene que incluirse en el dossier, pero debería consultarse en un futuro, durante el desarrollo. Los profesores deberían hacer cumplir los plazos rigurosamente para asegurarse de que los alumnos obtienen la máxima recompensa por el trabajo que han hecho.

Los objetivos deberían estar relacionados explícitamente con el análisis. Argumentar como objetivo "Voy a organizar los CD" no tiene sentido a menos que esté claro qué se debe demostrar para probar que se ha alcanzado el objetivo.

Se indicaron objetivos, pero casi nunca se hizo referencia a ellos en las secciones posteriores. En las secciones posteriores se debería haber argumentado el objetivo, de forma que estuviera presente la evidencia que apoya el cumplimiento de dicho objetivo.

Las limitaciones del sistema deberían abarcar más del hardware disponible, aunque por supuesto, esto se tendrá en cuenta. Tal como se menciona anteriormente, el alumno elegirá con frecuencia para resolver sólo una pequeña parte del problema del usuario, y eso se puede incluir como limitación.

Otra sugerencia es escribir objetivos **centrados en el usuario** en lugar de objetivos **centrados en el programa**. Debido a que los alumnos están centrados en el punto de vista del programa, sus objetivos siempre están relacionados con el programa.

Los objetivos centrados en el usuario describen lo que conseguirá el usuario, no las "funciones" que proporcionará el programa. Debería ser más fácil comprobar los objetivos centrados en el usuario. Y es obvio que probar un objetivo centrado en el usuario requiere ejemplos impresos en lugar de leer algunas líneas del listado de un programa. Por ejemplo, habría que considerar rescribir los objetivos de esta forma:

Orientados al programa	rescrito como	Orientados al usuario
El programa guarda títulos de CD en un archivo de datos.	→	El usuario puede guardar títulos de CD, recuperarlos y visualizarlos posteriormente.
El programa puede ordenar títulos de CD alfabéticamente.	→	El usuario puede visualizar los títulos ordenados alfabéticamente o cronológicamente (ordenados por fecha).

En algunos dossieres de la convocatoria actual se aceptaba la entrada de datos del usuario y su "almacenamiento" en una matriz, pero nunca en un archivo. Seguramente el usuario desearía conservar los datos de un día para otro, pero los programadores estudiantes pensaron, según parece, que bastaba con obtener los puntos del dominio de las matrices e ignorar la entrada/salida de ficheros.

Esto podría tener sentido en un prototipo, pero es difícil imaginarse escribir cualquier tipo de software de procesamiento de datos que pierda todos sus datos cuando se desconecte la máquina. De nuevo, un usuario real con necesidades reales y al que se consulte activamente podría identificar estos objetivos para el alumno. El alumno, como se argumenta anteriormente, no está obligado a hacer todo lo que desea el usuario, pero debería seleccionar cuidadosamente las 5 o 6 restricciones mencionadas anteriormente.

Criterio A3

Una vez que se identifican clara e inequívocamente los principales objetivos y límites de la solución propuesta para el problema, debería ser posible identificar los elementos principales de la solución, así como una planificación de la misma.

Esto lleva a un diseño modular inicial para un prototipo que se puede mostrar posteriormente al usuario para que lo comente. Los comentarios del usuario deberían documentarse, pero no hay que seguirlos siempre al pie de la letra, sobre todo si las sugerencias conducen a una solución demasiado extensa.

Un prototipo ayudará a que el alumno se forme una imagen clara del "producto" en su propia mente.

En esta convocatoria muchos alumnos no incluyeron un diseño inicial o información del usuario en el prototipo. A menudo se presentaron prototipos no funcionales, útiles, creados mediante varios paquetes gráficos.

Puede resultar beneficioso si un prototipo no funcional se escribe en Java, ya que los alumnos lo comprenderán mejor, lo cual resulta útil para el diseño de los algoritmos y, a la vez, satisface sus ansias de continuar y escribir código desde un primer momento. Esto, obviamente, no se debería llevar hasta el extremo. Un prototipo en Java con métodos para las principales funciones es un punto de partida muy útil para una posterior codificación.
A menudo, los comentarios de los usuarios resultaron realmente triviales o no se incorporaron para modificar el prototipo. Por lo general, los alumnos no respondieron a los comentarios del usuario sobre el prototipo.

El prototipo se debería usar para descomponer el programa en secciones lógicas desde el punto de vista del usuario. Esto conlleva, probablemente, elegir elementos de menú y campos de datos en un formulario. El diseño inicial puede estar más centrado en el programa, e incluir estructuras de datos y algoritmos. Si los alumnos se aproximan al "diseño inicial" y el "prototipo" desde estos dos puntos de vista diferentes, tiene sentido preguntarse si ambos están relacionados correctamente (criterios A3).

Criterio B1

En el Nivel Medio, las estructuras de datos principales serán las matrices o los archivos. Los alumnos que usen archivos deberían ilustrarlos, proporcionar datos de ejemplo del dominio del problema y justificar la elección de los tipos de datos.

Como en anteriores versiones de los criterios del dossier, los ejemplos son extremadamente valiosos y necesarios para obtener las máximas puntuaciones. Las ilustraciones deberían usar datos del problema que el alumno está intentando resolver. El uso de diagramas generales sobre el funcionamiento abstracto de las matrices o la E/S en archivos no satisface este criterio.

Los alumnos no están obligados a usar una biblioteca, funciones de utilidad o código que no hayan escrito o modificado ellos mismos para conseguir el aspecto de dominio. En el Nivel Medio, el principal problema a este respecto es la codificación usando Java.util.arrayList, que no se puede usar para obtener el dominio de las matrices como se espera de un tipo estático. Los archivos se deberían usar tanto para entrada como para salida. En una sección posterior se discuten aspectos de dominio específicos.

Criterio B2

Los algoritmos deberían estar lo suficientemente detallados como para se pueda crear la solución a partir de ellos. No tienen que ser perfectos o completos, pero deberían estar presentes todos los módulos esenciales identificados en B3 o A3. No todos están de acuerdo en que ésta es una forma útil de enfocar la escritura de código; sin embargo, sigue siendo un requisito en programa de Informática.

Como siempre, algunos alumnos presentaron código que derivaba claramente de la solución y fueron penalizados en consecuencia. Los moderadores detectan esta práctica muy fácilmente.

Criterio B3

Tal como se indica en la Guía de la asignatura, en esta convocatoria se aceptan varios tipos de presentaciones, que suelen depender del enfoque utilizado. Sin embargo, todas estas variaciones deberían mostrar las conexiones entre módulos o clases y algoritmos y estructuras de datos. A menudo, esta sección muestra un "gran cuadro" de la solución propuesta, tanto si está realizada como diseño modular descendente, diseño orientado a objetos o enfoque ascendente.

Se usaron varias presentaciones, pero el fallo más común fue no identificar las conexiones con las estructuras de datos y los algoritmos.

Criterio C1

En su mayoría respondieron bien y usaron una sangría correcta y unos nombres identificadores claros. Sin embargo, en muchos listados resultó difícil averiguar dónde empezaban y terminaban las clases y determinar la relación entre las mismas. Esto es más difícil cuando la solución se desvía del diseño, lo cual está permitido.

A ser posible, una clase debe comenzar en una nueva página o estar claramente separada de las clases precedentes, así como contar con una cabecera que describa su función. Es realmente útil contar con una lista de clases y los números de página al principio del código. Algunos IDE proporcionan diagramas de relación, que también resultan útiles para navegar a través del código.

Los alumnos deben asegurarse de que los comentarios son legibles si se escribe en blanco y negro; algunos IDE presentan los comentarios en gris y son difíciles de leer.

En esta sección **se debe** entregar un listado completo del programa, no basta con usar la sección B2 (algoritmo) u otras partes de la documentación como listado de código.

Criterio C2

Los alumnos realizaron correctamente esta sección en líneas generales. Muchos eligieron usar un enfoque de resumen, haciendo referencia a la salida impresa que se proporcionaba como parte de la sección D1. Esto funcionó mejor que el texto descriptivo sin ilustraciones o ejemplos específicos.

Esta sección debería hacer referencia a los criterios A2 explícitamente, y esto no se hizo en muchos casos. De nuevo, la idea que subyace a los criterios de facilidad de uso de A2 es escribirlos de forma que en esta sección se pueda demostrar que se han alcanzado. De ahí que sea difícil demostrar una "correcta facilidad de uso", mientras que resulta más fácil demostrar que "todos los menús tendrán una opción de ayuda".

Criterio C3

Respondieron correctamente en esta sección, que no ha cambiado desde el último programa, salvo que los criterios son un poco más específicos. Los mejores alumnos proporcionan ejemplos de código de gestión de errores, junto con referencias a salidas impresas en las que se muestran evidencias de la captura de dichos errores.

A menudo los alumnos realizan tablas de datos de prueba, que se incluyen en esta sección o en D1 y a las que posteriormente hacen referencia en esta sección. Ambos enfoques suelen recibir muy buenas puntuaciones y demuestran que los alumnos se organizan muy bien.

Criterio C4

Hubo unos pocos casos en los que los profesores otorgaron puntuaciones elevadas a pesar de que los alumnos no proporcionaron salida para el criterio D1. Si no hay pruebas de que un programa funciona, la nota para ese criterio es 0. Ciertamente, no basta con que el profesor exija que el programa funcione adecuadamente, deben ofrecerse evidencias en todos los casos.

La salida a la que se haga referencia en esta función también debería estar relacionada con los objetivos argumentados en A2, que se deberían argumentar de nuevo aquí, junto con cualquier evidencia que ayude a demostrar que se han conseguido dichos objetivos.

Criterio D1

En la situación ideal, las pruebas se ejecutan sistemáticamente, están bien planificadas y se ejecutan de forma estructurada. Las anotaciones deberían indicar qué parte de la solución (tal como se esboza en los objetivos) se está probando. No hay que repetirlo en la sección C4, pero debería aparecer en C4 o D1. Es factible considerar una sección que incorpore C4 y D1 bajo una sección titulada "Salida impresa con anotaciones", tal como se indica en la Guía de la asignatura, en el formulario 5/PDCS.

Esta sección es difícil de realizar eficazmente a menos que se enumeren objetivos claros y específicos en la sección A2. Aun cuando este fuera el caso, muchos alumnos no los probaron explícitamente en esta sección.

Esta sección tiene como objetivo demostrar el funcionamiento del programa en situaciones "normales"; muchos alumnos pusieron demasiado énfasis en la detección de errores y dejaron de lado todo lo demás. Cuando sea posible, se deben incluir ejecuciones que demuestren que las secciones del código para las que se reclama el dominio funcionan realmente. Un ejemplo común en el NM sería el correcto funcionamiento de las inserciones y eliminaciones en archivos y matrices.

Criterio D2

Se espera que el alumno reflexione sobre los logros del proyecto, aunque no debería olvidarse de todo lo demás. Tal como se indica en la introducción, el proceso es más importante que la complejidad de la aplicación que se esté creando. Por tanto, la eficacia de la solución debería discutirse sinceramente; cómo funciona para el usuario (en relación con los criterios argumentados en A2) en lugar de su calidad en términos de codificación (esto se puede discutir desde el punto de vista de la eficacia).

Además de cualquier mejora posible, el alumno debería examinar el proceso global que se ha llevado a cabo. Para ello, se puede ayudar de notas tomadas durante el proyecto, ya sea electrónicamente (por ejemplo, un blog) o en un cuaderno. Los resultados más valiosos de un dossier de Informática del BI podrían ser los defectos del proceso de diseño y cualquier otra lección que haya aprendido el alumno. En muy pocas ocasiones los alumnos tuvieron en cuenta alternativas al propio proceso de diseño.

Criterio D3

Parece que muchos profesores no tienen en cuenta aquello de "ejecutar el programa en otra máquina sin usar el IDE" a la hora de conceder 3 puntos en esta sección. Por lo general, son los mejores alumnos los que realizan las instrucciones o el uso más claro e ilustrado de los programas. Los alumnos menos preparados muestran evidencias de haberse precipitado en esta sección. Con una planificación detallada de las ejecuciones de prueba es posible realizar capturas de pantalla que proporcionen documentación para D1 y D3.

Criterio E

Da la impresión de que la nota holística se concedió justamente y sólo unos pocos colegios optaron por conceder los 3 puntos máximos, como se podría haber temido. Con frecuencia, los profesores tuvieron la amabilidad de explicar el porqué de la concesión de esta nota, lo cual ofrece confianza a los moderadores sobre la integridad de los colegios y los profesores.

Aspectos de dominio

Éstas son sólo directrices, se espera que los profesores usen su criterio profesional para la puntuación de los aspectos de dominio.

En la convocatoria de 2006 se puso de manifiesto alguna variación en la interpretación de los aspectos de dominio según los profesores, colegios y alumnos. Concretamente, algunos aspectos no se reconocen bien o no están lo suficientemente bien definidos en la Guía de la asignatura.

Esta sección del informe está diseñada para ilustrar la forma en que los alumnos deberían enfocar los aspectos de dominio y cómo deberían puntuarlos profesores.

Debemos insistir, no obstante, en que hay muchas formas de programar y de conseguir el dominio. Ninguno de los ejemplos que se presentan en esta sección deben tomarse como plantilla para que los profesores o los alumnos la apliquen a sus propios problemas.

Tal vez el mayor problema con los aspectos de dominio sea la frecuente falta de documentación o las inexactitudes en la misma. En estas carencias se incluyen varios aspectos como errores administrativos, falta de salida impresa y ausencia de discusiones sobre la necesidad de la solución para algunos aspectos de dominio.

Los números de página escritos por los estudiantes son a veces incorrectos, y el profesor debería revisarlos cuidadosamente antes de enviar el dossier y el formulario 5/PDCS. No se espera que los moderadores revisen las páginas con el listado de código para encontrar factores de dominio relevantes.

Los alumnos y los profesores, si lo prefieren, pueden hacer referencia a listas más completas de aspectos de dominio en el dossier. La ubicación de dichas páginas debería estar claramente indicada en el formulario 5/PDCS.

Si el profesor no completa los formularios adecuadamente, se podría devolver los dossieres a los colegios para su corrección, lo que produciría retrasos considerables en los resultados del diploma de los alumnos.

Documentación de aspectos de la fase de diseño

Entre los problemas típicos se encuentran los aspectos de dominio que no se discuten en la sección de estructuras de datos o que no se identifican algoritmos o módulos como manipuladores de estructuras de datos. Todavía hay otro aspecto que aparece misteriosamente en el código. A menudo se incluye una ordenación cuando aparentemente no hay necesidad de ello, y algunos alumnos reclaman el dominio en el Nivel Medio.

Uso de algoritmos estándares

Se permite el uso de algoritmos estándares, incluso se anima a que se incluyan en el dossier. Cualquier algoritmo estándar de alguna fuente pública se podría reconocer en condiciones normales. Sin embargo, los moderadores tienen que asegurarse de que el alumno entiende dichos algoritmos. Esto se puede demostrar de varias formas:

- El alumno discute las modificaciones necesarias para que el algoritmo estándar cumpla sus propios requisitos o explica claramente la función de dichos algoritmos en el contexto de su problema/solución concretos.
- El alumno demuestra al profesor una comprensión suficiente, que lo confirma realizando observaciones en 5/PDCS o 5/IACS y firmando los formularios.

Al igual que con otra materia del BI, copiar código de un libro, sitio Web u otra fuente (como por ejemplo otro compañero) se considera plagio.

Bibliotecas y utilidades de Java

Java posee amplias colecciones y rutinas de biblioteca incorporadas, y hay disponibles otras muchas de terceros. Aunque se podrían usar en las soluciones, ello no implica que reciban aspectos de dominio.

Esto se aplica concretamente a estructuras como ArrayList.

Matrices

La clase java.io.ArrayList u otras implementaciones similares a los vectores deberían evitarse, ya que este aspecto se refiere a matrices estáticas tradicionales. Se espera que los alumnos sean capaces de controlar la actualización de los elementos de una matriz, al mismo tiempo que evitan los errores típicos, como acceder a elementos nulos o fuera de los límites de la matriz.

Los alumnos, por tanto, deberían usar matrices estándares con subíndices entre corchetes.

Un buen ejemplo y bastante común podría ser introducir una lista de nombres y almacenarlos en una matriz de String. Luego, dejar que el usuario edite la matriz y finalmente los guarde en un fichero de texto.

Objetos definidos por el usuario

Se espera que los alumnos realicen la definición de una clase y que usen una instancia de dicha clase, así como sus miembros dato y/o métodos en otra clase. Sin embargo, en el Nivel Medio no es necesario que los alumnos comprendan los conceptos clásicos de la POO, como herencia y encapsulación.

No se permite crear el programa principal como una clase, ni tampoco crear objetivos a partir de clases estándar de Java. Un ejemplo podría ser la creación de una clase que pueda almacenar (y usar un método para validar) un número de teléfono que se use posteriormente dentro del programa para almacenar y validar número de teléfono.

Objetos como registros de datos

Esto implica usar una clase con miembros dato, que se corresponda con los campos de un registro. Para evitar un enfoque trivial, los campos deberían ser de tipos diferentes, según los datos que se estén almacenando. Una clase de este tipo podría incluir la validación tanto en el método constructor como modificador (setter) si el alumno es capaz (aunque no es estrictamente necesario).

Este aspecto tiene la ventaja de cumplir el aspecto de dominio anterior simultáneamente.

Almacenar los datos en el programa principal no satisface este aspecto. La clase escrita tiene que usarse en el programa principal. Un ejemplo podría ser crear una clase CD que contuviera campos como título, artista y longitud/número de pistas. Por lo general, el uso de datos que no sean String es un requisito. La clase CD se debería usar para crear objetos CD en el programa principal.

Selección simple

Implica el uso de una o varias instrucciones if o else-if para controlar la ejecución del código. El número exacto es difícil de especificar, aunque es deseable que sea más de uno.

```
if (puntuaciones[x] < notaMinima)
{
  resultado[x] = “suspenso”;
}
```

Las instrucciones de selección aparecen repetidamente en casi todos los programas. No es necesario documentar todos los ejemplos, es suficiente documentar un par de ejemplos claros.

Un bloque try..catch no cuenta como selección. Para demostrar el dominio es necesario proporcionar más de un ejemplo de instrucción if en un programa.

Selección compleja

Un ejemplo de selección compleja podría ser el uso de una instrucción if con múltiples condiciones (y operadores lógicos), instrucciones if anidadas, varias instrucciones if-else anidadas o una instrucción switch.

Los alumnos que sepan manejar la selección compleja también pueden reclamar el dominio de la selección simple.

Ejemplo 1

```
if ( (x < 8) && (x >0) )
{
  ok = true;
}
```

Ejemplo 2

```
if (x < 8)
{
  if (x > 0)
  {
    ok = true;
   }
}
```

Ejemplo 3

```
if (opc == 1)
{
  agregarArticulo(Comida articulo)
}
else if (opc == 2)
{
  eliminarArticulo(Comida articulo)
}
```

Aunque `switch` no forma parte de JETS, se puede usar igualmente en el último ejemplo.

Normalmente, este aspecto no se dará sin la documentación detallada. Un bloque try..catch no cuenta como selección compleja.

Bucles

Cualquier tipo de bucle (obviamente debe funcionar) servirá para este criterio. Este aspecto aparece con frecuencia en dossieres no triviales. Un bloque try-catch no se puede considerar un bucle. Se necesitará más de un ejemplo en un programa para obtener el dominio de este aspecto.

Se recomienda que los alumnos eviten condiciones complejas en los bucles for, ya que se puede producir fácilmente bucles infinitos (con lo cual no se demuestra dominio). Una búsqueda secuencial en una matriz es un buen ejemplo del uso de bucles y muestra también el aspecto de la búsqueda (cuando se documente adecuadamente en la sección B). Los bucles que nunca se ejecutan no demuestran dominio, por ejemplo:

```
for(x=0;x<0;x++)
```

Bucles anidados

Los bucles anidados son aquellos bucles que se ubican correctamente dentro de otros bucles y producen el resultado esperado, que se ha documentado en alguna sección.

```
while ( cp != (SIZE - 1) )  //mientras no se llegue al final de
la matriz
{
  cp = cp + 1    // incrementar actual (no ordenados)
  pt = cp;        // puntero a la parte ordenada – comienza al
principio
  tm = datos[cp]; // almacenamiento temporal para el elemento a
insertar
  // mientras no esté al principio, y el valor siguiente sea
demasiado elevado,
  // incrementar el elemento actual en 1
  while ( (pt > 0) && (datos[pt-1] > tm) )
  {
    datos[pt] = datos[pt - 1];
    pt = pt - 1;
  }
  // insertar el valor temporal en la parte ordenada
  datos[pt] = tm;
}
```

Hay que tener en cuenta que este bucle forma parte de una rutina de ordenación y, por tanto, incluye tres aspectos de dominio. No obstante, la **necesidad** de ordenar **debería** haberse establecido en el **diseño** de la solución y **debería** haberse demostrado su funcionamiento mediante una **copia impresa**.

El ejemplo siguiente muestra un simple bucle y una simple selección, NO bucles anidados o selecciones complejas:

```
for (int pos = inicio + 1; pos <= fin; pos = pos + 1)
      {
          if (datos[pos] < minimoProvisional)
          {
            // se ha encontrado un nuevo mínimo
            minimoProvisional = datos[pos];
            posMin = pos;
          }
      }
      return posMin;
    }
```

Métodos definidos por el usuario

Las buenas técnicas de programación llevarán, de forma natural, al uso de pequeños segmentos de código que realizan una única tarea y que se combinan en un método. Es posible que estos métodos no devuelvan ningún valor.

La documentación de un método **debería** especificarse en la sección de diseño de algoritmos (B2), para que se puntúe este aspecto de dominio.

El uso de public static void main no cuenta como método definido por el usuario, como tampoco cuenta un método constructor simple para un programa principal. Los métodos creados por el IDE o requeridos por la implementación de interfaces Java estándar (e.g. ActionListener) no son válidos.

Un método que sólo se invoca una vez no cuenta como dominio de este aspecto. Algunos ejemplos de usos correctos de métodos: búsqueda, ordenación, validación de la entrada de usuario y formateo de una cadena de texto.

Métodos con parámetros definidos por el usuario

Los parámetros se pasarán a un método y se usarán dentro del cuerpo. Un ejemplo clásico podría ser imprimir líneas blancas en el terminal de salida:

```
public void imprimirBlancos(int n)
{
  for (int x = 0; x < n; x++)
    System.out.println("");
}
```

De nuevo, esto debería documentarse en alguna sección de B2. Si el alumno sigue usando instrucciones múltiples como:

```
System.out.print("\n\n\n\n\n\n\n\n\n");
```

en lugar de

```
imprimirBlancos(8);
```

indica que tal vez no domina el concepto de método con parámetros.

Si el anterior es el único ejemplo de método del programa, resulta verdaderamente trivial, ya que se espera que la mayoría de los programas no triviales hagan un uso correcto y eficaz de métodos de todo tipo.

Como moderadores, esperamos que no se use este método en casi todos los dossieres de la próxima convocatoria, ya que se trata de un ejemplo de muestra, no de una plantilla para que la usen los alumnos.

Los métodos creados por el IDE o requeridos por la implementación de interfaces Java estándar (e.g. ActionListener) no son válidos.

Métodos definidos por el usuario que devuelven valores

Estos métodos no son de tipo void, sino que devuelven algún tipo de valor mediante una instrucción return. Sin embargo, se requiere algo más que una simple instrucción return. He aquí un ejemplo trivial (y una práctica de programación pésima):

```
public void devolverCuarentaydos(int y)
{
  d = 42;
  return;
}
```

No se usa el parámetro, d no es de ámbito local y la instrucción return es redundante.

De nuevo, un problema no trivial del dossier al que se supone que debe enfrentarse el alumno, debería ofrecer una gran oportunidad para que los alumnos desarrollen métodos adecuados que cumplan dichos criterios.

Los métodos creados por el IDE o requeridos por la implementación de interfaces Java estándar (e.g. ActionListener) no son válidos. No se tiene en cuenta un método al que sólo se llama una vez.

Ordenación

Una de las razones para contar con más de 10 de los aspectos de dominio requeridos es que los alumnos no tengan que buscar problemas que requieran un aspecto concreto. Por tanto, se espera que los alumnos introduzcan una ordenación por alguna razón que esté relacionado con el dominio del problema en lugar de una razón que simplemente satisfaga un requisito de dominio.

Un problema relacionado es que una ordenación reciba tres aspectos de dominio y, por tanto, el moderador necesite estar convenido, de lo contrario el alumno podría poner en peligro más de un aspecto de dominio.

La ordenación debería usar bucles o recursividad y no, por ejemplo, una serie de instrucciones de selección (if) sobre un pequeño conjunto de datos.

Búsqueda

No hay restricciones particulares sobre la necesidad de realizar una búsqueda de otros elementos que no sean los indicados. Una simple búsqueda lineal requerirá un bucle y una instrucción de selección, y posiblemente también el uso de un centinela.

Éste es un proceso más bien simple, con el que obtener 4 aspectos de dominio y, por tanto, la documentación del diseño debería ser adecuada y la salida impresa debería demostrar la corrección. Además, si la búsqueda es la única parte del programa en que aparece la selección, la iteración y los indicadores, se considerará trivial.

La búsqueda debería usar bucles o recursividad y no, por ejemplo, una serie de instrucciones de selección (if) sobre un pequeño conjunto de datos.

Entrada y salida mediante archivos

Los datos deben escribirse y leerse en un archivo (podría ser un simple archivo de texto o un objeto OutputStream) y deben persistir tras las ejecuciones del programa, y también es necesario demostrarlo mediante salida impresa.

Se permite que los alumnos del Nivel Medio usen una base de datos SQL para satisfacer este requisito (por ejemplo, Access) y funciones de java.sql para actualizar la base de datos (debe haber operaciones de adición, edición y eliminación). Con esto también se cumplirán los criterios para el uso de bibliotecas adicionales.

En este caso, SQL no se puede usar para demostrar el dominio de la búsqueda. El uso de una base de datos SQL no implica dominio de los objetos como registros de datos (el alumno ha usado la base de datos, no Java, para crear el conjunto de registros).

No se puede usar un comando SQL fijo, los valores se deben recuperar de manera flexible; por ejemplo, el comando SQL para actualizar el conjunto de datos debe crearse de forma flexible, de tal manera que se pueda pasar cualquier valor, no un conjunto predefinido mediante programación.

Una buena forma de usar el aspecto de esta forma podría ser mediante la creación de un método para guardar una matriz de valores en una base de datos SQL y otro método para recuperar dichos valores y copiarlos en la matriz.

Al igual que con el resto de aspectos de dominio, no se puede conceder si no se proporciona ninguna razón para su uso o explicación de su funcionamiento en la sección B1, además de ejemplos de prueba

e ilustraciones. La solución debe tener un requisito real para almacenamiento a largo plazo (al igual que muchas aplicaciones) y es necesario argumentar explícitamente este requisito.

Uso de bibliotecas adicionales

Math y String forman parte de JETS, así que se debería usar bibliotecas distintas de Abstract Windows Toolkit, Swing, una colección integrada o una utilidad como StringTokenizer. Se puede implementar una interfaz o importar una biblioteca y, por tanto, ActionListener es un ejemplo válido (con tal de que los listeners se establezcan y detecten realmente).

Es importante que, en la fase de diseño, se demuestre y documente la importancia de contar con tales bibliotecas. Si la utilidad proporciona almacenamiento u otras funciones, el correcto funcionamiento se debe demostrar también mediante salida impresa.

No se tiene en cuenta el uso de clases creadas por el alumno.

No basta con usar código generado automáticamente por el IDE (por ejemplo, Swing), el alumno también debe escribir código por sí mismo que use la biblioteca, y dicho código debería estar identificado claramente como propio.

En esta convocatoria, no siempre se reclamó este aspecto de dominio, cuando parece evidente que debería haber sido así.

Uso de centinelas o indicadores

El centinela podría ser, por ejemplo, el marcador final de un conjunto de datos, aunque no forma parte de los datos. No es un valor que tiene que escribir el usuario (por ejemplo “sí” o “no”). No es un valor booleano devuelto por algún método.

A menudo, se declara un indicador en una búsqueda o como devolución de un método concreto (por ejemplo, una validación). Normalmente, este aspecto formará parte de la mayoría de los programas no triviales del dossier y aparecerá, con toda probabilidad, en más de una sección.

El indicador no tiene por qué ser un tipo de datos booleano.

Un buen ejemplo podría ser marcar el final de una lista de datos en una matriz con un valor ficticio (“xxx” o 999) y programar una búsqueda para detenerse cuando alcance este valor.

Recomendaciones para la enseñanza de alumnos futuros

Muchas recomendaciones serán evidentes a partir de la sección anterior sobre criterios específicos; sin embargo, todas las secciones del dossier deberían beneficiarse de que los alumnos se aseguran que su elección del tema:

- Incluye un usuario concreto.
- Implica algo que entiende.
- Se puede resolver con el lenguaje de programación de que disponen.
- Es aprobado por el profesor.
- Satisfará los aspectos de dominio requeridos para el NM.
- Es investigado y analizado ANTES de empezar a escribir el programa.

Al igual que en sesiones anteriores, muchos profesores no completan el reverso del Formulario 5/IACS y no incluyen comentarios con los ejemplos enviados. Los comentarios ayudan al moderador

a evaluar la nota otorgada por el profesor. El moderador desea confirmar la nota concedida por el profesor siempre que sea razonable hacerlo.

En algunos casos parece que los profesores no concedieron factores de dominio aun habiendo evidencias en dossier de que se habían alcanzado. El moderador, por lo general, corroborará la opinión del profesor, que es el más adecuado para otorgar esos puntos y no modificará la nota.

Los comentarios del profesor y una adecuada documentación por parte del alumno podrían evitar penalizaciones en los casos en los que no se entendieron correctamente los aspectos de dominio y, por tanto, no se obtuvieron cuando podrían haberse conseguido.

Se han cometido algunos errores al calcular la nota final en el Formulario 5/IACS. Los profesores deberían leer atentamente las instrucciones en el Vademécum a la hora de calcular y registrar las notas.

Los profesores deberían incentivar a los alumnos a mostrar el dominio en todos los aspectos, ya que pueden perder muchos puntos si no lo hacen.

La discusión en la evaluación debería realizarse desde el punto de vista de la programación, no desde los logros personales como programadores.

Los profesores deben discutir los criterios y directrices de evaluación con los alumnos antes de comenzar el dossier de trabajo personal. Los alumnos deben tener claro lo que cada criterio requiere de ellos. En el Nivel Superior, los profesores deberían tener cuidado en esbozar los aspectos de dominio requeridos y discutir el uso trivial frente al no trivial.

El desarrollo del dossier debería seguir las fases siguientes:

- Análisis
- Diseño
- Desarrollo
- Documentación

En concreto, tal como se advierte anteriormente, el análisis y el diseño no deberían completarse después de haber finalizado la solución, ya que sería una pérdida de tiempo y esfuerzo por parte del alumno.

Una estrategia excelente para la enseñanza es insistir a los alumnos sobre la realización de las secciones A y B antes de llevar a cabo ningún desarrollo. Esto no sólo creará una base sólida para una mejor calificación final en el dossier, sino que también mejorará e informará sobre el desarrollo de la solución real del alumno. En términos matemáticos, teniendo en cuenta que estas secciones valen 24 puntos, los alumnos podrían haber conseguido una puntuación suficiente para continuar, sin tener que alcanzar la perfección en ningún criterio posterior para obtener una calificación final elevada. No obstante, esto es extremadamente difícil de conseguir a menos que se curse el programa en dos años.

Los moderadores podrán no confirmar los aspectos de dominio y los alumnos serán penalizados si:

- No se documentan adecuadamente los aspectos de dominio en 5/PDCS (o cualquier otra sección del dossier).
- No se documentan los aspectos de dominio en la Sección B.
- No se documentan los aspectos de dominio en la sección D1 (si procede).
- El alumno utiliza algoritmos de un libro, biblioteca de utilidades o sitio Web sin especificarlo.

- El profesor no ha confirmado la comprensión de algoritmos estándar por parte del alumno.

Nivel Superior Prueba 1

Bandas de calificación del componente

Nota final:	1	2	3	4	5	6	7
Puntuaciones:	0-15	16-30	31-41	42-51	52-61	62-71	72-100

Niveles de logro

Hay un gran rango de notas: pocas muy bajas (menos que en años anteriores), así como muchas muy altas. No hay diferencias notables entre los resultados obtenidos en las secciones A y B. No hay evidencias de que a los alumnos les haya faltado tiempo. Sólo algunos han respondido más preguntas que las indicadas en las instrucciones. En ambos casos sólo se han corregido las cuatro primeras preguntas de la sección B.

Áreas del programa y del examen que parecen haber resultado difíciles para los estudiantes

Los alumnos responden por lo general de forma correcta a preguntas sencillas, como introducir datos en un árbol binario de búsqueda, definir DMA y rastrear un algoritmo.

La representación en punto flotante, el ciclo de instrucción de máquina y los conceptos de OMR y *sondeo* son términos básicos de informática que no resultaron familiares a algunos alumnos.

Se entiende muy bien los términos *encapsulación*, *herencia* y *polimorfismo*, relacionados con la POO. Muchos alumnos pueden dar una definición, aunque los intentos de relacionar estas definiciones con una aplicación demuestran que no se entienden completamente. De forma similar, los ejemplos proporcionados para *interrupciones* eran usos válidos de *interrupciones*, pero guardan poca relación con la aplicación.

Puntos fuertes y débiles de los estudiantes al abordar las distintas preguntas

En líneas generales, el examen tenía un nivel adecuado, con unas pocas preguntas que ocasionaron problemas importantes a TODOS los alumnos o correctores.

La distribución de notas parece normal, habiendo un número razonable de alumnos con notas elevadas, distinguiéndose claramente de los 6 o 7 niveles de rendimiento más bajos.

Áreas del programa que parecen haber resultado difíciles para los estudiantes

En la sección A, las preguntas 2 y 13 resultaron difíciles, así como la 16 de la sección B. Estas preguntas tratan sobre representación de datos en binario y hexadecimal, junto con cuestiones aritméticas y los errores asociados. Éste es un tema en el que deberían centrarse los alumnos y los profesores.

La pregunta 8 (c) estaba fuera del alcance del curso y pocos alumnos respondieron correctamente.

Varios alumnos contestaron bastante mal a las partes del algoritmo en la pregunta 16 de la sección B. Estos algoritmos son razonablemente estándares y resulta decepcionante, dados los requisitos del dossier, que los alumnos no sean capaces de contestar a preguntas tan básicas. Los profesores y los alumnos deben prestar más atención al desarrollo de algoritmos.

Las respuestas a la pregunta 5 de la sección A son bastante mediocres, probablemente debido al cambio en la forma de tratar estos conceptos en el curso. La noción de argumento es el valor que se pasa a las variables declaradas en la signatura del método. Por ejemplo, en el caso de media(12, 20), tanto 12 como 20 son argumentos. Los parámetros se declaran en la signatura del método. Por ejemplo, en double average (double n, double m), las variables n y m son parámetros.

En el futuro, este tipo de pregunta usará los términos “signatura del método” y “llamada al método”. Los alumnos pueden esperar que se les pregunte sobre un aspecto concreto de la signatura de un método o sobre los componentes de la llamada a un método.

Niveles de conocimiento, comprensión y habilidad demostrados

En líneas generales, los alumnos muestran una comprensión razonable del curso, con la excepción de la representación binaria y los algoritmos.

En general, los alumnos no suelen responder las preguntas de tipo explicación o discusión que hacen énfasis en los requisitos de los verbos de acción. Los estudiantes deben tener claro el significado de estos términos. Además, por lo general, la aplicación de la regla "una puntuación n requiere n puntos" ayudaría a los alumnos a estructurar sus respuestas.

Puntos fuertes y débiles de los estudiantes al abordar las distintas preguntas

Algunos se han mencionado anteriormente, en (A).

Puntos fuertes

Los alumnos demuestran una comprensión adecuada de la teoría base. Hay buenas respuestas a las preguntas 1, 2, 4 o 6.

Respondieron correctamente a la pregunta 17 sobre lógica booleana. Muchos alumnos muestran capacidad para simplificar expresiones booleanas.

Una gran variedad de alumnos responde correctamente a la pregunta 18, mostrando una comprensión básica de los archivos secuenciales y de las cuestiones éticas y sociales asociadas con la obtención de datos.

Muchos estudiantes respondieron correctamente a la pregunta 19; muchos son capaces de definir acceso directo y construir un árbol binario para almacenar el índice.

La pregunta 20 demuestra que muchos estudiantes entendieron las ideas básicas de sondeo e interrupciones y que fueron capaces de recomendar la técnica adecuada. Responden bien a las preguntas (c) y (d), demostrando una buena comprensión de los distintos métodos de entrada y captura de datos. También saben determinar sus ventajas y desventajas.

Puntos débiles

Algoritmos: se espera que los estudiantes puedan representar la lógica de la solución de un problema básico en forma de algoritmo. Muchos estudiantes no son capaces de gestionar métodos que reciben

datos y posteriormente devuelven un valor. Java permite métodos void que actúan como procedimientos y no devuelven ningún valor. Java también admite métodos con un tipo de datos que indica que se devuelve un valor; dichos métodos operan igual que una función.

Se espera que los estudiantes sean capaces de escribir soluciones modulares y reutilizar métodos según sea necesario.

Tienen dificultades para comprender la representación en binario y hexadecimal. Los estudiantes deben ser conscientes de que se les puede realizar preguntas que exijan la comprensión de la representación de enteros positivos y negativos, la representación hexadecimal, los valores máximos y mínimos de n bits, la suma de enteros binarios y errores como el desbordamiento.

Acceso a archivos: muchos alumnos no saben responder la pregunta 19 (e). El acceso directo requiere el conocimiento de las direcciones relativas o físicas en disco. A menudo se usa un índice para almacenar la clave primaria y la posición relativa del registro. Se espera que los estudiantes puedan explicar cómo se usa una estructura de datos como, por ejemplo, una matriz, una lista enlazada o un árbol binario para conseguir acceder a los registros directamente usando una clave primaria. En esta instancia, la clave primaria (userID) se usa para acceder a los nodos del árbol binario y, a continuación, se usa la posición relativa del registro para buscar dicho registro en el archivo.

Las respuestas a la pregunta 20 (a) son bastante pobres. Los profesores y los alumnos deben saber representar las relaciones entre los componentes que se usan en un diagrama de flujo de sistemas. La dirección de la relación también es importante.

Recomendaciones y orientaciones para la enseñanza de futuros estudiantes

Los alumnos deben asegurarse de que sus respuestas están relacionadas con la informática. Las respuestas que muestran un "sentido común" adecuado pero ningún conocimiento técnico no obtienen ningún punto.

Muchos alumnos demuestran conocimientos técnicos, pero son incapaces de aplicarlos a la pregunta específica. Para evitar esta situación, tenemos dos posibles estrategias: en primer lugar, leer la pregunta atentamente antes de intentar responderla.

Una pequeña reflexión ayudaría a aplicarla correctamente. En segundo lugar, practicar con preguntas de exámenes anteriores. Cuanto más diversas sean las aplicaciones que se encuentre, mayor probabilidad tendrán de aplicar los conocimientos a nuevas situaciones.

Nivel Superior Prueba 2

Bandas de calificación del componente

Nota final:	1	2	3	4	5	6	7
Puntuaciones:	0-15	16-30	31-40	41-50	51-61	62-71	72-100

Niveles de logro

El hecho de que sólo haya unos cuantos guiones muy pobres y algunos excepcionales confirma que el examen fue, en líneas generales, asequible para la mayoría de los alumnos. En general, el nivel demostrado fue muy satisfactorio. La mayoría de los alumnos hizo un gran esfuerzo en los algoritmos. Sólo unos cuantos dejaron sin responder alguna parte de esta pregunta y otros produjeron algoritmos excepcionalmente buenos. Muchos alumnos demostraron un buen conocimiento de la informática y

bastantes fueron capaces de mostrar una comprensión mayor de la teoría. No hay evidencias de que a los alumnos les haya faltado tiempo.

Áreas del programa y del examen que parecen haber resultado faciles o difíciles para los estudiantes

La construcción del algoritmo resultó especialmente correcta, aunque la llamada a un procedimiento o función previamente escritos, como en el caso de HAMMING e INTERCAMBIAR en la pregunta 1, ocasionan cierta dificultad a los alumnos.

En las respuestas a la pregunta sobre el estudio de caso se observa una falta de conocimientos informáticos y, a su vez, están basadas en afirmaciones repetitivas extraídas del estudio de caso (a veces se incluyen algunas irrelevantes).

Los diagramas de flujo de sistemas no resultaron familiares a muchos alumnos; algunos produjeron en su lugar diagramas de flujo de programación.

Los métodos para la recuperación de datos y los tipos de procesamiento no parecen haberse estudiado con el nivel suficiente para responder adecuadamente a las preguntas.

Áreas del programa que parecen haber resultado difíciles para los estudiantes

Parece que los alumnos fallaron en dos áreas principales: aquellas en las que había que interpretar, trazar y escribir algoritmos y aquellas en las que no. Algunos alumnos parecen conocer muy poco el lenguaje Java y de estar muy poco preparados para este examen.

Muchos alumnos mostraron muy pocos conocimientos sobre la conmutación de paquetes y sobre conceptos de redes relacionados. Los conocimientos sobre estructuras de fichero parecen haber mejorado en relación con años anteriores, aunque algunos alumnos obtuvieron resultados muy pobres en esta área.

Suelen responder bien al estudio de caso, y parece que los alumnos menos preparados obtienen la mayoría de sus puntos en esta área.

Hubo algunos alumnos que demostraron conoce bien los conceptos de programación y la asignatura en general, aunque obviamente esos son los alumnos que obtuvieron mejores puntuaciones.

Nivel de conocimiento, comprensión y habilidad demostrados

Los alumnos que se presentaron a este examen poseen una amplia base de conocimientos. El rendimiento de muchos alumnos ronda la media. Pocos alumnos demostraron niveles muy elevados tanto de conocimientos como de habilidades a la hora de responder a todo tipo de preguntas.

Parece que completar una tabla de rastreo recursiva es un problema para muchos alumnos. Parece que muchos alumnos tienen problemas con los conceptos relacionados con las funciones de tipo void en contraposición con aquellos que devuelven un valor. Muchos alumnos encuentran dificultades con los parámetros, lo que resulta sorprendente en el Nivel Superior.

Muchos alumnos responden de forma imprecisa y nada específica y no prestan atención suficiente a la asignación de la puntuación o a los verbos de acción que se usan en una pregunta.

Puntos fuertes y débiles de los estudiantes al abordar las distintas preguntas

Pregunta 1

Tal como se menciona anteriormente, fueron pocos los alumnos que obtuvieron puntuaciones medias en este pregunta. O sabían trabajar con algoritmos o no sabían. De hecho, parece que algunos alumnos saben muy poco de programación en Java, y nos preguntamos cómo consiguen completar un dossier de programa del Nivel Superior.

Es cierto que un algoritmo recursivo es complicado de rastrear, pero el resto de preguntas eran relativamente sencillas. Hubo muy pocas soluciones completas en la tabla de rastreo; muchos alumnos parecían completamente perdidos.

Por lo general, los alumnos que lo intentaron realizaron correctamente el algoritmo de la parte (b). A los alumnos les resultó difícil eliminar los duplicados de la matriz, aunque la mayoría pudo marcarlos (por ejemplo, con 0 o null). Hay problemas a la hora de realizar la transferencia y calcular correctamente la matriz resultante.

La mayoría de alumnos que intentaron hacer las partes (b) y (c) también pudieron completar la parte (d) correctamente.

Pregunta 2

Los alumnos que comprendieron los principios básicos de la conmutación de paquetes y los conceptos de redes relacionados respondieron, por lo general, bien a esta pregunta. Sorprendentemente, éste no es el caso de muchos colegios. De entre aquellos que respondieron correctamente, muchos se esforzaron para obtener todos los puntos de la parte (b) relacionada con la trasmisión de paquetes a través de Internet. Debe tenerse en cuenta que también hubo varias respuestas excelentes a esta parte.

En algunas ocasiones, a los alumnos les faltaron conocimientos sobre comprobación en la comunicación de datos, tal como se requería en la parte (d), aunque las respuestas relacionadas con los tipos de medios específicos (como fibra óptica), también se aceptaban.

Los alumnos respondieron bastante bien a la parte (e) y, por lo general, supieron identificar la encriptación como medida de seguridad. Algunos alumnos no se dieron cuenta de la necesidad de la desencriptación en la parte del receptor.

Pregunta 3

Unos cuantos alumnos confundieron las estructuras de archivo con los tipos abstractos de datos, lo que resultó decepcionante. La mayoría de alumnos pudo, al menos, identificar archivos en serie y secuenciales, y un buen número pudo identificar los cuatro tipos argumentados en la guía de la asignatura.

Los alumnos que sabían algo sobre archivos no siempre fueron claros y específicos en sus respuestas, que a menudo resultaron imprecisas. A menudo confundieron el concepto de indexación con la diferencia entre archivos completamente indexados (sin ordenar) y parcialmente indexados (ordenados); ambas se aceptaban como respuestas para la parte (c) si se explicaba adecuadamente. Unos cuantos alumnos también obtuvieron puntos por explicar el proceso de acceso directo mediante tablas hash para esta pregunta parcial.

La parte (e) requería que los alumnos eligieran un medio e indicaran cuatro o más puntos en apoyo de dicha elección. No todos los alumnos parecen reconocer que el número de puntos estaba relacionado con la asignación de puntos para esta pregunta, lo que sugiere que tal vez necesitarían más práctica en las técnicas de examen.

Pregunta 4

La pregunta sobre el estudio de caso sigue resultando difícil para muchos alumnos. Parece que el primer contacto que tienen con él es en la sala de exámenes el día de la prueba. A veces, las respuestas están formadas por citas aleatorias del estudio de caso, lo cual está bastante poco relacionado con esta pregunta.

El estudio de caso se introduce para permitir que los profesores exploren conceptos de informática dentro de una aplicación específica, en este caso, MIDI.

Las dos partes (a) y (b) sobre los procesos de grabación de música mediante MIDI y sampling son un ejemplo al respecto. Muchos alumnos parecen no entender las diferencias fundamentales entre estos dos procesos y ofrecen ejemplos del estudio de caso que no demuestran la comprensión de dichos procesos. De nuevo, los alumnos no aprecian el número de puntos requeridos para cada una de las partes de estas preguntas.

Respondieron mejor a la parte (c) sobre métodos de grabación, lo que sugiere que los alumnos tenían más conocimientos acerca de los métodos de grabación de lo que demostraron en las partes (a) y (b).

En las partes (e) y (f) se ofrecieron a menudo respuestas imprecisas, generales y ambiguas. De nuevo, los alumnos no supieron proporcionar el número de puntos necesarios, tal como indicaban el verbo de acción usado y la asignación de puntos.

Casi todos los alumnos respondieron correctamente a las preguntas de la parte final (g).

Tipo de sugerencias y recomendaciones que deberían ofrecer los profesores a futuros estudiantes

Muchos alumnos muestran una debilidad general en las técnicas de examen aunque, por otra parte, parecen mostrar aptitud en la asignatura. Una técnica útil es proporcionar a los alumnos guiones de ejemplo para que realicen anotaciones. Los profesores pueden escribirlos con el fin de demostrar errores comunes como:

- No proporcionar ningún tipo de respuesta.
- Responder a una pregunta diferente de la que se formula.
- Escribir demasiado para una simple pregunta de “argumentar”.
- Escribir poco para una pregunta que vale muchos puntos.

Se debería exponer a los alumnos a que escribieran algoritmos en condiciones similares a las de un examen, y un buen momento para hacerlo es antes de los exámenes de prueba y del mismo examen. No hay alternativa real a la práctica de la traza, interpretación y escritura de algoritmos con plazos de entrega.

Los alumnos deberían recibir una copia de la guía de la asignatura y contar con tiempo suficiente para evaluar sus conocimientos sobre la misma, así como de paliar las deficiencias más significativas antes del examen. Parece que muchos alumnos no sabían absolutamente nada sobre las áreas más extensas de la Guía de la asignatura.

Recomendaciones y orientaciones para la enseñanza de futuros estudiantes

Exponer a los alumnos a una variedad de diagramas de flujo de sistemas en las primeras fases del curso y animarlos posteriormente a representar sistemas de esta forma al explorar sistemas podría

ayudarles a comprender la diferencia entre los diagramas de flujo de sistemas y los diagramas de flujo de programación.

Se debería estudiar con frecuencia los diferentes tipos de procesamiento y los detalles sobre cómo y cuando se presentan. Concretamente, se debería animar a los alumnos a describir qué ocurre desde el punto de vista de la informática, no sólo desde el punto de vista de un usuario.

Las pruebas anteriores cronometradas deberían hacer ver que leer la pregunta atentamente es esencial para proporcionar una respuesta adecuada.

Nivel Medio Prueba 1

Bandas de calificación del componente

Nota final:	1	2	3	4	5	6	7
Puntuaciones:	0-12	13-24	25-30	31-37	38-43	44-50	51-70

Generalidades

El examen parece bastante claro. La mayor parte de los alumnos obtuvo buenos resultados en este componente y sólo unos pocos obtuvieron malas notas. Había oportunidades para obtener puntos fácilmente, pero aun así hubo pocas notas altas. También hubo una gran correlación entre las dos secciones. La mayoría de los alumnos que obtuvieron buenas puntuaciones en la Sección A también las obtuvieron en la Sección B, y aquellos que obtuvieron malos resultados lo hicieron en ambas secciones.

Áreas del programa y del examen que parecen haber resultado difíciles para los estudiantes

La mayoría de los alumnos realizaron correctamente al examen. Los alumnos peor preparados proporcionaron respuestas en las que había una falta evidente de detalle o de conceptos informáticos. Muchos de esos alumnos con poca preparación fueron incapaces de dar definiciones sin usar la palabra que estaban intentando definir (e.g. “Los errores lógicos son errores de la lógica”). Esos alumnos, por tanto, no ofrecieron correctamente algunas definiciones.

S encontraron carencias especialmente en la definición de errores, en la diferencia entre un índice y el dato de la matriz (P y X[P]), en la diferencia entre en línea y tiempo real y en el ciclo de vida del software. Otras respuestas en que se aprecia la carencia de detalles son: la definición de parámetro por referencia, la comprensión de cómo se pueden enlazar los archivos y la definición del tipo de procesamiento durante la cuenta de los nuevos usuarios (pregunta 14).

Áreas del programa o del examen en que los estudiantes demostraron estar bien preparados

Muy pocos alumnos dejaron preguntas sin resolver. La mayoría muestra un buen conocimiento sobre compiladores, compresión de datos, LAN y redes. Realizaron correctamente el rastreo de algoritmos y completaron correctamente la tabla de rastreo.

Puntos fuertes y débiles de los estudiantes al abordar las distintas preguntas

Sección A

Pregunta 1

Incluso los alumnos menos preparados fueron capaces de argumentar una función de los compiladores.

Pregunta 2

Se proporcionan algunas respuestas correctas a esta pregunta sobre algoritmos, aunque algunos alumnos aún confunden entre p y X[P]; no distinguen entre los datos de cada posición de la matriz y el índice. Hay algunos alumnos que no saben qué es la "inicialización".

Pregunta 3

Acceso a archivos. Se encontraron muchas respuestas que carecían de conceptos informáticos. Los alumnos mencionaron los beneficios para los usuarios en relación con el acceso directo. Se encontraron muchas respuestas imprecisas, como "es muy rápido".

Pregunta 4

Muchos alumnos no incluyeron ejemplos de "tiempo real" y "en línea". También ofrecieron respuestas usando las mismas palabras que intentaban definir.

Pregunta 5

La mayoría de alumnos fue capaz de explicar la diferencia entre validación y verificación.

Pregunta 6

Todos los alumnos, salvo los menos preparados, respondieron correctamente a la pregunta sobre representación de datos.

Pregunta 7

Salvo aquellos que usaron la palabra "comprimir" en su respuesta, ningún alumno tuvo problemas con la compresión de datos.

Pregunta 8

Pregunta sobre errores: sencilla, salvo para aquellos que, de nuevo, no encontraron las palabras para explicar "errores lógicos" o "errores sintácticos" sin usar las palabras "lógico" o "sintaxis". Algunos están convencidos de que una división por cero es un error lógico.

Pregunta 9

Respondieron correctamente a la pregunta sobre memoria primaria y secundaria, aunque algunos no se dieron cuenta de que se requerían dos funciones para cada tipo de memoria (se podían otorgar 4 puntos).

Pregunta 10

Los alumnos ofrecieron muchas respuestas diferentes a la pregunta sobre el ciclo de vida del software, y parece que muchos habían leído la pregunta muy rápidamente, ya que no tuvieron en cuenta las palabras "claramente" y "antes" a la hora de contestar.

Sección B

Pregunta 11

Rastreo del algoritmo. La mayoría de alumnos que respondieron a esta pregunta lo hicieron correctamente. Rastrearon el algoritmo y completaron la tabla correctamente. Sin embargo, no fueron capaces de proporcionar respuestas con el nivel de detalle requerido al explicar por qué la función necesitaba devolver un resultado de tipo real y al explicar qué se entiende por paso por referencia. Parece que estos alumnos no tuvieran los conocimientos para presentar las respuestas con el nivel de detalle necesario.

Pregunta 12

Esta pregunta relacionada con los archivos era muy fácil, pero ni siquiera los alumnos más preparados consiguieron una puntuación alta.

Casi todos los alumnos supieron explicar cómo se verifica el tiempo de entrada (parte a). En la parte b, muchos alumnos no relacionaron la respuesta con la informática y proporcionaron respuestas que implican "privacidad del cliente" y "seguridad para el cliente", para asegurarse de que nadie más use su cuenta. En la parte c, los alumnos no tienen muy claro el enlace de archivos y se observa, fundamentalmente, una falta de profundidad en sus repuestas. Muy pocos alumnos fueron capaces de esbozar correctamente el tipo de procesamiento de archivos necesario (parte c ii). Se han encontrado respuestas como "secuencial" o "directo", sin más detalles. Además, hay muchos alumnos que respondieron "procesamiento por lotes", y los peor preparados indicaron que el procesamiento de archivos era una "búsqueda".

Pregunta 13

Pregunta sobre redes de área local. Pregunta fácil con muy buenas respuestas. Casi todos los alumnos que respondieron a esta pregunta incluyeron un diagrama correcto de la red y fueron capaces de argumentar la topología (partes a y b). La mayor parte de alumnos conocía la función de un concentrador y la explicaron correctamente (parte c). Al relacionar las ventajas de una LAN (parte d), algunos alumnos no relacionaron sus respuestas con los usuarios, y sólo proporcionaron respuestas relacionadas con características generales de las redes. No tienen problemas para esbozar una medida de seguridad (parte e) ni argumentar el dispositivo hardware necesario para proporcionar acceso a Internet (parte f). Sin embargo, algunos alumnos que respondieron "pasarelas" o "encaminadores" no conocían exactamente la función de cada uno.

Pregunta 14

Ésta es, quizá, la pregunta más difícil. No tuvieron problemas para indicar los tipos de software necesarios para acceder al servicio (parte a), salvo aquellos alumnos que sólo incluyeron nombres comerciales. Casi todos los alumnos supieron argumentar el procesamiento por lotes (parte b) y elaboraron correctamente su respuesta. Muchos alumnos fueron capaces de esbozar las características del sistema informático necesario (parte c), e indicaron la necesidad de contar con servidores o con entornos multiusuario. Sin embargo, los alumnos peor preparados mostraron alguna confusión entre multitarea y multiusuario. No hay ningún problema con el restablecimiento de datos en caso de fallo (parte d). Aunque muchos alumnos comprendieron lo que ocurre cuando se cuentan los nuevos usuarios (parte e), pocos pudieron esbozar el "tipo de procesamiento informático". Se encontraron muchas preguntas que carecían de conceptos informáticos.

Recomendaciones y orientaciones para la enseñanza de futuros estudiantes

- Se debe animar a los alumnos a que lean detenidamente las preguntas, que sean precisos en sus respuestas y se aseguren que incluyen términos de informática en las mismas. Esto se puede conseguir mediante la realización de exámenes anteriores.
- Animar a los alumnos a dar definiciones sin usar la palabra que se está intentando definir.

A menudo ocurre que los alumnos muestran una buena comprensión de los conceptos básicos, pero cuando se presenta un conocimiento más técnico, muchos de ellos son incapaces de realizar comparaciones válidas o encuentran difícil proporcionar respuestas con el nivel de detalle requerido. Los conceptos mencionados en la Guía de la asignatura deberían estudiarse en mayor profundidad.

Nivel Medio Prueba 2

Bandas de calificación del componente

Nota final:	1	2	3	4	5	6	7
Puntuaciones:	0-11	12-22	23-29	30-35	36-41	42-47	48-70

Generalidades

Es evidente que el examen resultó más difícil que el año pasado. Pocos alumnos consiguieron puntuaciones elevadas. Esto puede deberse a dos factores: La ausencia de elección en el nuevo formato de examen y el aumento del número de preguntas sobre algoritmos, un tema que ha resultado difícil para muchos estudiantes de este nivel.

Hay algunos colegios en los que es evidente que no se ha enseñado completamente el curso de Informática y han entrado tras estudiar un curso intensivo de TIC. La capacidad de construir y comprender algoritmos es fundamental en el curso. Aún hay alumnos que no consiguen ningún punto en estas preguntas. Por otra parte, hay estudiantes que muestran una gran capacidad para el razonamiento lógico y producen soluciones elegantes. El éxito o el fracaso en este tema es específico de cada colegio.

Puntos fuertes y débiles de los estudiantes al abordar las distintas preguntas

Pregunta 1 (Algoritmo)

Hubo notas de todo tipo. (a) y (b) trataban sobre teoría básica de la programación (**signaturas** y **alcance**). Algunos entienden qué información contiene la signatura, pero pocos comprenden qué se entiende por ámbito. Para muchos alumnos, y de forma incorrecta, ámbito significa rango de valores. Para otros, la signatura es el objetivo del algoritmo. El ámbito es de particular importancia cuando se usan objetos y clases, y su comprensión debería mejorar a medida que los alumnos se acostumbren a Java. Es obvio que muchos alumnos construyen programas en sus colegios sin comprender claramente la teoría que sustenta sus programas.

El rastreo del algoritmo propuesto tiene diversos resultados. Los dos algoritmos que tenían que construir los alumnos fueron una buena prueba y se obtuvieron algunas soluciones muy buenas. Tal como se indica anteriormente, hay varios colegios cuyos estudiantes apenas intentan realizar los algoritmos y son incapaces de formular correctamente incluso las estructuras más básicas. Sigue siendo un misterio cómo los mismos alumnos han presentado dossieres completos para la evaluación interna.

Un problema común en el uso de matrices es la confusión entre el índice del último elemento y el número de entradas de la matriz.

Pregunta 2

Respondieron bastante bien a la pregunta sobre prototipos (a). Hay un solapamiento entre esta pregunta y el proyecto, donde los alumnos tienen que proporcionar ejemplos de prototipos.

A la pregunta sobre el uso de sensores (b) respondieron sorprendentemente muy mal. Muchos fueron incapaces de describir un sensor adecuado para el ejemplo del ferrocarril. Muchos argumentaron un sensor concreto, pero luego se equivocaron al describir su funcionamiento. Repetir la pregunta en la respuesta es un error común de este nivel. Obviamente, argumentar que "el sensor captura la posición del tren" no merece ningún punto.

Muchos respondieron bien a la pregunta sobre tipos de procesamiento (c), aunque las preguntas sobre algoritmos (d) y (e) de nuevo resultaron difíciles. Muchos no supieron diferenciar entre estructuras de datos y tipos de datos (d) y, por tanto, tuvieron problemas a la hora de construir el algoritmo (e). El algoritmo real fue una buena prueba para los mejores alumnos. Pocos encontraron soluciones adecuadas.

Pregunta 3 (Estudio de caso)

Es obvio, según las repuestas, en qué colegios se ha estudiado el tema con detalle y en cuáles no. Aunque no se realicen preguntas sobre aspectos del estudio de caso que, en cierta medida, aparecen en el propio estudio de caso, un conocimiento previo de la materia proporciona a los estudiantes una ventaja considerable sobre el resto.

Es necesario que los alumnos presten especial atención a los verbos de acción que se usan, y algunos colegios tienen que preparar a sus alumnos mejor en este aspecto, ya que muchos respondían de la misma forma (es decir, simplemente "argumentando") incluso cuando el verbo de acción indicaba claramente una respuesta más profunda y extensa.

Varios alumnos no entienden por qué se usa hexadecimal, creen que en cierta manera se ahorra espacio en memoria.

Recomendaciones y orientaciones para la enseñanza de futuros estudiantes

La capacidad para comprender y formular algoritmos es fundamental en este curso, y el nuevo formato de examen hace que sea imposible suprimir algunos temas. Por tanto, hay que tratar los algoritmos como un tema continuo, que se debe enseñar durante todo el programa, y no sólo como consecuencia de los programas del alumno. Debido a que los contenidos algorítmicos han aumentado no es posible conceder puntuaciones elevadas en este examen si no se logra competencia en esta destreza.

Se debería revisar cuidadosamente la forma de responder a los verbos de acción, debido a que muchos alumnos dejan de obtener la máxima puntuación en determinados tipos de preguntas ("explique" o "discuta") a causa de la brevedad de sus respuestas.

