SISTEMAS INFORMÁTICOS I **Septiembre 2003 - Original**
INGENIERÍA EN INFORMÁTICA - Código Carrera 55 - Código Asignatura 077 **Tipo de Examen: B**
NO SE PERMITE EL USO DE NINGÚN MATERIAL **Tiempo: 2 horas**

Apellidos: _____ **Nombre:** _____ **DNI:** _____

Centro Asociado en el que está MATRICULADO: _____ **Especialidad:** _____

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja, y **ENTRÉGUELA OBLIGATORIAMENTE** con el resto de hojas de su examen. **Cualquier examen que no venga acompañado de esta hoja de enunciados no será corregido**. Complete **TODOS** los datos que se piden en la hoja de lectura óptica o **en caso contrario su examen no será corregido.** El examen consta de 20 preguntas. Cada respuesta acertada tiene un valor de 0.5 puntos y cada respuesta incorrecta de -0.25 puntos. Para superar el examen es necesario obtener una puntuación de al menos 5 puntos.

Test : Conteste exclusivamente en una HOJA DE LECTURA ÓPTICA, no olvidando marcar que su tipo de examen es B.

1- El parámetro CODEBASE de la etiqueta <*applet*> se utiliza para

A) Permite especificar para las clases o archivos recopilatorios una ubicación diferente a la del archivo HTML que contiene la etiqueta <applet>.
B) Indica en qué lenguaje se ha escrito el applet.
C) No es un parámetro permitido en una etiqueta <applet>.
D) Ninguna de las anteriores es correcta.

2.- Qué significa el "." como valor del parámetro CODEBASE de una etiqueta <applet>

A) Significa que se bloquea el applet hasta que se pulse la tecla F4.
B) No tiene ningún significado especial.
C) Que el archivo HTML se encuentra en el mismo subdirectorio que los ficheros para las clases o archivos recopilatorios.
D) Ninguna de las anteriores es cierta.

3.- Para arrancar un hilo ¿qué método se usa?

A) Run.
B) Start.
C) Se inicializa por defecto.
D) Ninguna de las anteriores.

4.- Java en la programación de hilos utiliza como algoritmo de planificación de los mismos

A) Programación fija de prioridades. El hilo con la mayor prioridad será el que se esté ejecutando siempre.
B) No se usan prioridades. Se ejecutan en el orden de creación.
C) Utiliza un algoritmo *Round Robin.*
D) Se calcula la prioridad en función del número de bytes de código.

5.- El metódo run( ) de la clase Thread se utiliza

A) Para implementar la acción del hilo.
B) Arrancar la ejecución del hilo.
C) No es un método de la clase Thread.
D) Ninguna de las anteriores es correcta.

6. - ¿Dónde se encuentra en Java 2 la clase Arrays?

A) java.util
B) java.io
C) java.lang
D) Ninguna de las anteriores es correcta.

7. - ¿Cómo se puede saber si un hilo es demonio?

A) Llamando a isDaemon()
B) Llamando a setSaemon()
C) Llamando a notDaemon()
D) Ninguna de las anteriores es correcta.

8. - ¿Cómo se suspende la ejecución de un hilo?

A) Llamando a suspend()
B) Llamando a wait()
C) Las respuestas A y B son correctas.
D) Llamando a suspend()

9. - La salida de *javadoc* es un documento:

A) jpx.
B) java.
C) HTML.
D) Ninguna de las anteriores.

10.- En la entrada con formato desde memoria, para leer datos "con formato" se deben usar las clases

A) InputStream.
B) Reader.
C) Las dos a la vez.
D) Ninguna de las anteriores.

11.- La clase `Escribe` se define de la manera siguiente (se han numerado las sentencias para facilitar su localización):

```
1     public class Escribe {
2        public String Saludo() {
3           return "Hola";
4           }
5        public static void main(String arg[]) {
6           System.out.println(Saludo());
7           }
8     }
```

Indique cuál de las siguientes afirmaciones es correcta:

A) Se obtiene un error de compilación en la línea 2: la declaración del método no es válida, puesto que no se indica el tipo del retorno (invalid method declaration; return type required).
B) Se obtiene un error de compilación en la línea 6: el método no estático Saludo() no puede ser referenciado desde un contexto estático (non-static method Saludo() cannot be referenced from a static context)
C) Las dos afirmaciones anteriores son correctas.
D) La clase es correcta, por lo que puede ser compilada sin errores.

12.- Indique si son correctas las siguientes afirmaciones acerca de las clases abstractas:

I) En una clase abstracta (abstract class) pueden declararse métodos, pero no pueden definirse .
II) En una clase abstracta, la declaración de los métodos abstractos puede no contener el tipo del retorno (return type). Por ejemplo, sería posible definir el método perimetro() de la forma siguiente: public abstract perimetro();. Esto facilita la sobrecarga de los métodos.

A) I: Sí; II: Sí
B) I: Sí; II: No
C) I: No; II: Sí
D) I: No; II: No

13.- Indique si son correctas las siguientes afirmaciones acerca de las clases abstractas (abstract classes) y las interfaces:

I) Los métodos de las clases abstractas y de las interfaces deben ser forzosamente abstractos.

II) Una clase no puede heredar más de una clase abstracta, pero si puede implementar varias interfaces.

A) I: Sí; II: Sí

B) I: Sí; II: No

C) I: No; II: Sí

D) I: No; II: No

14.- Las variables pueden ser tanto de tipos primitivos como referencias a objetos de alguna clase. Indique cuál de las siguientes afirmaciones acerca de las variables es correcta:

I) La memoria que ocupa una variable de un tipo primitivo, así como su rango de valores, puede depender del tipo de ordenador.

II) Una referencia a un objeto es una variable que indica dónde está en la memoria del ordenador el objeto.

A) I: Sí; II: Sí

B) I: Sí; II: No

C) I: No; II: Sí

D) I: No; II: No

15.- Indique cuál de las siguientes afirmaciones, relacionadas con el ámbito (*scope*) de las variables, es correcta:

A) Las funciones miembro de una clase B derivada de otra A (es decir, A es super-clase de B), tienen acceso a todas las variables miembro de A declaradas como *public* o *private*, pero no a las declaradas como protected.

B) Es posible declarar una variable dentro de un bloque (es decir, dentro de unas llaves {}) con el mismo nombre que una variable miembro. La variable declarada dentro del bloque oculta a la variable miembro en ese bloque. Para acceder a la variable miembro oculta será preciso utilizar el operador *super*.

C) Las dos afirmaciones son correctas.

D) Todas las anteriores son falsas.

16.- Qué resultado produce el siguiente código

```
InetAddress addr= InetAddress.getByname(null);
```

A) Asigna la IP 126.291.0.0

B) Asigna la dirección localhost.

C) No es código Java, luego no hace nada.

D) Ninguna de las anteriores.

17.- El contenido del bloque *finally*:

A) Sólo se ejecuta cuando la excepción ha sido previamente capturada y procesada.

B) Se ejecuta únicamente cuando la excepción lanzada no ha sido procesada por un bloque *catch*.

C) No se ejecuta si existe un bloque *catch* que capture la excepción que se ha producido.

D) Se ejecuta independientemente de que las excepciones sean o no capturadas.

18.- Un bloque *catch*:

A) Necesita un único argumento, esto es, la excepción que captura.

B) Puede especificar varios argumentos, esto es, excepciones separadas por comas.

C) Siempre tiene que ir acompañado de un bloque *finally*.

D) Puede ir sin argumentos si de pretende capturar cualquier tipo de excepción.

19.- ¿Cuál de las siguientes afirmaciones es correcta?

A) Para un bloque *try* pueden existir dos bloques catch que capturen la misma excepción. Uno se ejecutará siempre a continuación del otro.

B) Para un único bloque *try* nunca puede haber dos bloques catch que capturen la misma excepción. Se produce un error en tiempo de compilación ya que se trata de un error sintáctico.

C) Se genera un error durante la ejecución del programa si se produce una excepción y hay dos bloques *catch* pertenecientes al mismo bloque *try* que procesan la misma excepción.

D) Puede haber dos bloques *catch* capturando la misma expepción, siempre y cuando el primero de ellos relance la excepción mediante una clausula *throw* para que el siguiente *catch* la capture.

20.- Para conseguir ubicar los componentes en una interfaz gráfica de forma libre, esto es, sin ajustarse a un layout determinado, hay que:

A) Primero se utiliza el método setLayout(null) para fijar el layout del contenedor y, a continuación, setBound para establecer el tamaño y posición de los componentes a ubicar en el contenedor.

B) En Java es imposible utilizar posicionamiento libre en un layout. Siempre hay que ceñirse a un layout o a implementaciones Java de otros fabricantes que no sean la de Sun Microsystems.

C) Primero hay que liberar el contenedor con el método freeLayout(true) y luego se ubican los componentes mediante el método add.

D) Con independencia del layout que se haya establecido con el método setLayout se puede recurrir al método setBounds para fijar la posición y tamaño del componente grafico.