SISTEMAS INFORMÁTICOS I **Septiembre 2003 - Reserva**
INGENIERÍA EN INFORMÁTICA - Código Carrera 55 - Código Asignatura 077 **Tipo de Examen: C**
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 C.

1.- Indique cuál de las siguientes respuestas corresponde con el resultado de la ejecución del método *main* de la clase *CreaElementos*:

```
public class Elemento {
   public static int numElementos=0;
   public Elemento() {
      ++numElementos;
      }
   protected void finalize() {
      numElementos--;
      }
}

public class CreaElementos {
   public static void main (String args[]) {
     for (int i=1; i<=10000; i++) {
       new Elemento();
       }
     System.out.println("Num: "+Elemento.numElementos);
     }
}
```

A) `Num: 0`
B) `Num: 10000`
C) `Num: 1`
D) `El resultado puede ser diferente de una ejecución a otra.`

2.- El rango del tipo de dato simple *short* (entero corto) es –32768 ... 32767. Indique cuál de las siguientes afirmaciones, acerca del resultado de la ejecución del método *main* siguiente, es correcta:

```
public static void main (String args[]) {
   short i=32766;
   for (short j=0;j<4;j++) {
      System.out.print(i+j+" ");
      }
   }
```

A) Java realiza una comprobación de los rangos de las variables. Por este motivo, la ejecución del método escribe los dos primeros números de la secuencia, 32766 y 32767, y a continuación aborta mostrando un mensaje de error.
B) La secuencia obtenida de la ejecución es: 32766 32767 -32768 -32767. Esto es debido a que el resultado i+j es de tipo *short* y a que Java no realiza una comprobación de los rangos.
C) Las operaciones aritméticas entre dos variables de tipo *short* genera como resultado un *int*. Por este motivo, el resultado de la ejecución es: 32766 32767 32768 32769.
D) Ninguna de las afirmaciones anteriores es correcta.

3.- Indique cuál de las siguientes respuestas corresponde con el resultado de la ejecución del método *main* de la clase *Parametros*:

```
public class Objeto {
   int variable;
   public Objeto( int var ) {
      variable = var;
      }
}

public class Parametros {
   public static void main(String args[]) {
      int var1 = 2;
      final int var2 = 10;
      Objeto obj1 = new Objeto(var2);
      final Objeto obj2 = new Objeto(var2);
      modifica(var1, obj1, obj2);
      System.out.println(var1+" "+obj1.variable+" "+obj2.variable);
      }
   static void modifica (int ii, Objeto vv, Objeto VV) {
      ii++;
      vv.variable++;
      VV.variable++;
      }
}
```

A) 3 11 10 B) 2 11 11 C) 2 11 10 D) 3 10 10

4.- Indique cuál de las siguientes respuestas corresponde con el resultado de la ejecución del método *main* de la clase *ClaseB*:

```
abstract class ClaseA {
   String v;
   void metodo() {
      v = "Padre";
      }
}

public class ClaseB extends ClaseA {
   String v;
   void metodo() {
     v = "Hija";
     super.metodo();
     System.out.print(v+" ");
     System.out.println(super.v);
     }
  public static void main( String args[] ) {
     ClaseB b = new ClaseB();
     System.out.print(b.v+" ");
     b.metodo();
     }
}
```

A) null Hija Padre
B) null Padre Padre
C) Padre Padre
D) Hija Padre Padre

5.- ¿Qué es un *constructor por defecto* de una cierta clase?

A) Un constructor con el mismo nombre que la clase y sin parámetros.
B) Un constructor que devuelve la clase base.
C) Un constructor sobrecargado.
D) Un constructor que crea elementos de esa clase.

6.- Si un hilo está creado como *hilo demonio* (*daemon*), los hilos que cree serán siempre:

A) Un hilo demonio no puede crear hilos.
B) Obligatoriamente no serán hilos de tipo demonio.
C) Serán todos hilos de tipo demonio.
D) Depende de cómo se creen.

7.- Cuál de las siguientes afirmaciones es falsa. Un hilo puede estar bloqueado porque:

A) Se ha puesto a dormir llamando a sleep()
B) Se ha suspendido la ejecución con suspend()
C) Se ha suspendido la ejecución con wait()
D) Se ha suspendido al recibir un mensaje resume()

8 .- Qué resultado produce el código siguiente:

```
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.

9.- Cuando se crea un *Server Socket*:

A) Se asigna una dirección IP.
B) Se asigna un número de puerto.
C) Se bloquea un servidor.
D) Ninguna de las anteriores.

10.- Cuando se crea un *socket*, se debe indicar:

A) Una dirección IP y un número de puerto.
B) Sólo un número de puerto.
C) Sólo una IP.
D) Un socket no necesita ni una dirección IP ni un número de puerto.

11.- En Java 1.2 y versiones superiores, para poder recibir los eventos del tipo XXX producidos por un componente es necesario crear el componente y:

A) Registrar el componente como un listener mediante el método addXXXListener y recurrir al método action chequeando qué componente ha producido el evento y si se trata del evento XXX.
B) Recurrir a la clase Event especificando que hay que permanecer a la escucha de eventos del tipo XXX.
C) Implementar la interfaz del listener correspondiente al evento XXX y utilizar el método handleXXXEvent preguntando por el componente y el tipo de evento que se ha producido.
D) Implementar la interfaz del listener correspondiente al evento XXX y registrar el componente como un listener mediante el método addXXXListener.

12.- Un archivo Jar (extensión .jar) permite que:

A) Una aplicación sea interpretada con mayor velocidad debido al factor de compresión aplicado al contenido del archivo Jar.
B) Un applet sea transmitido mediante una única petición al servidor HTTP al incluir el archivo Jar todos los ficheros necesarios.
C) Varios ficheros sean compilados simultáneamente al haberlos comprimido previamente en un único archivo con la extensión .jar.
D) Un applet se transmita mediante varias peticiones al servidor HTTP pero se ejecute en el destino con mayor rapidez ya que el formato Jar es un formato de fichero .class compilado con optimización avanzada.

13.- La carga e inicialización del driver JDBC-ODBC se realiza mediante la expresión:

A) Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
B) Class.loadDriver ("sun.jdbc.odbc.JdbcOdbcDriver");
C) Class.loadName ("sun.jdbc.odbc.JdbcOdbcDriver");
D) Class.forDriver ("sun.jdbc.odbc.JdbcOdbcDriver");

14.- Una vez que se han realizado los pasos para efectuar la conexión a la base de datos, es decir, se ha preparado la URL y se ha inicializado el driver JDBC-ODBC, hay que:

A) Llamar al método Connection.createEstament() e indicar la URL de la base de datos como parámetro.
B) Llamar al método Connection.getConnection( ) y pasar como parámetros la URL, el nombre de usuario y la contraseña de acceso a la base de datos.
C) Llamar al método DriverManager.createEstament ( ) y pasar como parámetros la URL de la base de datos.
D) Llamar al método DriverManager.getConnection( ) y pasar como parámetros la URL, el nombre de usuario y la contraseña de acceso a la base de datos.

15.- Si se dispone de una cadena de caracteres y se desea realizar un análisis de las palabras que componen la cadena, lo más adecuado es:

A) Crear un objeto StreamTokenizer y obtener las palabras del objeto mediante el método nextToken().
B) Convertir la cadena de caracteres en un InputStream y utilizar el objeto InputStream creado como argumento del constructor de StreamTokenizer.
C) Crear un objeto StringTokenizer y obtener las palabras recurriendo al método nextToken().
D) Convertir la cadena de caracteres en un InputStream y utilizar el objeto InputStream creado como argumento del constructor de StringTokenizer.

16.- ¿Cómo se nombra un archivo/clase interna?

A) Con el nombre de la clase contenedora, seguido de %, seguido del nombre de la clase interna.
B) Con el nombre de la clase contenedora, seguido de &, seguido del nombre de la clase interna.
C) Con el nombre de la clase contenedora, seguido de $, seguido del nombre de la clase interna.
D) Ninguna de las anteriores.

17.- En el tratamiento con excepciones, la clausula *finally*:

A) Siempre se ejecuta.
B) Se ejecuta en función de las condiciones.
C) Nunca se ejecuta.
D) Ninguna de las anteriores.

18.- ¿Qué es un *applet*?

A) Un programa que se ejecuta sin control.
B) Un programa que se ejecuta dentro del navegador web.
C) Un programa ejecutable desde el entorno de programación.
D) Ninguna de las anteriores.

19.- ¿Qué es un *socket*?

A) Una conexión física entre dos máquinas.
B) La abstracción software para representar los terminales de una conexión entre dos máquinas.
C) La abstracción software para representar los enlaces de una conexión entre dos máquinas.
D) Ninguna de las anteriores.

20.- 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.