SISTEMAS INFORMÁTICOS I — **Junio 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: _____

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.- En la interfaz `InterfA` se declara el método *metodoInterfA*. La clase `ClaseA` implementa la interfaz `InterfA`, y además define el método *metodoClaseA*. En el siguiente fragmento de código, se pretende que una referencia a la interfaz `InterfA` sirva de referencia para un objeto de la clase `ClaseA`.

```
InterfA a;
a = new ClaseA( );
```

Indique, de las siguientes afirmaciones, cual es la correcta:

A) Una referencia a la interface InterfA no puede servir de referencia para un objeto de la clase ClaseA. Por ello, el fragmento anterior de código es incorrecto.
B) El objeto al que referencia a puede ser usado con el método *metodoInterfA*, pero no con el método *metodoClaseA*.
C) El objeto al que referencia a puede ser usado con el método *metodoClaseA*, pero no con el método *metodoInterfA*.
D) El objeto al que referencia a puede ser usado con el método *metodoInterfA* y con el método *metodoClaseA*.

2.- Un applet que no sea del tipo de confianza puede acceder a un disco local para:
A) Sólo escribir
B) Sólo lectura
C) Para leer y escribir
D) No puede acceder al disco local ni para leer ni para escribir.

3. ¿Cuál de las siguientes afirmaciones es falsa?

A) Un programa no puede tener una mezcla de *daemon* y *non-daemon* threads.
B) Un programa puede tener un único *daemon* thread y varios *non-daemon* threads.
C) Un programa puede tener una mezcla de *daemon* y *non-daemon* threads.
D) Un programa puede tener varios *daemon* threads y un único *daemon* threads.

4. Para conseguir *multithreading* en una clase que hereda de otra clase que no sea la clase *Thread* hay que:

A) Implementar la interfaz *Runnable*.
B) Heredar de la clase *Multithread*.
C) Implementar la interfaz *Thread*.
D) Recurrir al modificador *threaded* para etiquetar los métodos.

5. El método *start* de un applet puede llamarse:

A) Cuando se considere necesario.
B) Una única vez al igual que el método *init*.
C) Exclusivamente cuando se va a llamar después al método *stop*.
D) Siempre después de *init* ya que es obligatorio.

6. Para permitir la ejecución de otros threads que están esperando se recurre al método:

A) *yield*.

B) *changePriority*.
C) *resume*.
D) *changeThread*.

7. Para conectar a través del subprotocolo ODBC con una base de datos identificada como *people*, la expresión que construye la JDBC URL necesaria es:

A) `String dbUrl = "jdbc:odbc:people";`
B) `String dbUrl = "jdbc://odbc.people";`
C) `String dbUrl = "jdbc:odbc://people";`
D) `String dbUrl = "jdbc-odbc:people";`

8.- Se declara una variable dentro de un bloque (es decir, dentro de unas llaves {}) con el mismo nombre que una variable miembro (campo) de la misma clase. Indique cual de las siguientes afirmaciones es correcta:

A) Este tipo de declaración no está permitido en Java. Nunca puede darse el mismo nombre a una variable local de un bloque y a una variable miembro (campo) de la misma clase.
B) La variable declarada dentro del bloque oculta a la variable miembro en ese bloque. No es posible acceder en ese bloque a la variable miembro (campo) oculta.
C) La variable declarada dentro del bloque oculta a la variable miembro en ese bloque. Para acceder en ese bloque a la variable miembro oculta será preciso utilizar el operador *this*.
D) La variable declarada dentro del bloque oculta a la variable miembro en ese bloque. Para acceder en ese bloque a la variable miembro oculta será preciso utilizar el operador *super*.

9.- Señale el valor que asignan a `z` las sentencias siguientes: `x=1; y=10; z=(x<y)?x+3:y+8;`

A) `z=0` B) `z=18` C) `z=11` D) `z=4`

10.- Se desea obtener exactamente la siguiente salida:

```
i=1 j=15
i=2 j=25
i=3 j=35
i=4 j=45
```

Indique qué fragmento de código permite obtenerla:

A)
```
for (int i=1, j; i<5; i++, j=10*i+5) {
   System.out.println("i="+i+" j="+j);
   }
```
B)
```
for (int i=1; i<5; i++) {
    System.out.println("i="+i+" j="+i+5);
    }
```
C)
```
int i=1;
 do {
   System.out.println("i="+i+" j="+(10*i+5));
   } while ((i++)<5);
```

D) Todos los fragmentos de código anteriores producen exactamente la salida indicada en el enunciado.

11.- En Java se denomina *encapsulación* a:

A) La posibilidad de acceder a clases privadas
B) La posibilidad de acceder a clases y métodos privados
C) Al hecho de poder envolver u ocultar datos y miembros dentro de las clases
D) Ninguna de las anteriores

12.- Indique si las siguientes afirmaciones son verdaderas:

I) Las variables miembro de las clases, que no son inicializadas por el programador, son inicializadas por defecto.
II) Las variables locales de los métodos, que no son inicializadas por el programador, son inicializadas por defecto.

A) I: Sí; II: Sí.

B) I: Sí; II: No.

C) I: No; II: Sí.

D) I: No; II: No.

13.- Al llamar a un método sobrecargado, Java sigue unas reglas para determinar el método concreto que debe llamar. Indique si Java aplica cada una de las siguientes dos reglas:

I) Si no existe un método que se ajuste exactamente al tipo de los argumentos de la llamada, siempre se produce un error.

II) El valor de retorno influye en la elección del método sobrecargado. Por ello, es posible crear dos métodos sobrecargados, es decir con el mismo nombre, que sólo difieran en el valor de retorno.

A) I: Sí; II: Sí.

B) I: Sí; II: No.

C) I: No; II: Sí.

D) I: No; II: No.

14.- Indique si las siguientes afirmaciones acerca del paso de argumentos a los métodos son correctas.

I) Los tipos primitivos se pasan siempre por valor. El método recibe una copia del argumento actual; si el método modifica esta copia, el argumento original que se incluyó en la llamada no queda modificado.

II) Las referencias se pasan también por valor, no pudiéndose modificar, a través de ellas, los objetos referenciados.

A) I: Sí; II: Sí.

B) I: Sí; II: No.

C) I: No; II: Sí.

D) I: No; II: No.

15.- Considérese el siguiente fragmento de código, correspondiente a la definición de la clase *DosNumeros*. Con el fin de poder hacer referencia a ellas al plantear esta pregunta, se han numerado las sentencias del 1 al 10:

```
1       public class DosNumeros {
2           public double x, y;
3           public DosNumeros(double x, double y, int A) {
4              this.x = A;
5              this.y = y;
6           }
7           public DosNumeros() {
8              final double PI = 3.14159;
9              this(PI,0.0,0);
10      }
```

El fragmento de código anterior contiene un error. Indique cual de las siguientes respuestas es la correcta:

A) Error en la línea 4, al realizar la conversión de tipos. El programador debe hacer un *cast* explícito.

B) Error en el constructor definido en las líneas 3 a 6: el argumento x no se usa en la definición del constructor.

C) Error en la línea 8. En un constructor no se puede definir una variable *final*.

D) Error en la línea 9. Un constructor de una clase puede llamar a otro constructor previamente definido en la misma clase por medio de la palabra *this*. En este contexto, la palabra *this* sólo puede aparecer en la primera sentencia de un constructor.

16.- Señale cual de las siguientes afirmaciones acerca del funcionamiento del recolector de basura (*garbage collector*) es más correcta:

A) Puesto que es normal que varias variables del tipo referencia apunten al mismo objeto, Java lleva internamente un contador de cuántas referencias hay sobre cada objeto. En el preciso instante en que el número de referencias a un objeto es cero, el objeto es borrado por el recolector de basura.

B) Se puede forzar que se produzca la operación de recolección de basura, en determinado punto de la ejecución del programa fijado por el programador, llamando al método *System.gc( )*. Este método activa el recolector de basura, liberando en ese instante la memoria de los objetos que han perdido su referencia.

C) El recolector de basura sólo libera la memoria reservada con *new*.

D) Todas las respuestas anteriores son correctas.

17.- Cuando se compila un fichero Java se obtiene

A) Un único fichero con el mismo nombre pero extensión **.class**.

B) Un fichero **.class** por cada clase del fichero **.java**.

C) Un fichero con extensión **.jar**.

D) Un fichero **.jar** por cada clase del fichero **.java**.

18.- En Java las excepciones son:

A) Métodos.

B) Datos.

C) Objetos.

D) Ninguna de las anteriores.

19.- ¿Cuál de estas funciones no pertenece a la clase **Arrays** en Java?

A) equals().

B) fill().

C) setVisible().

D) binarySearch().

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.