Universidad Carlos III de Madrid

Programación de Sistemas
Grado en Ingeniería Telemática

Leganés, 13 de marzo de 2013 — Examen parcial 1 (teoría)
Duración de la prueba: 15 min — Puntuación: 5 puntos sobre 10 del examen

Sólo una opción es correcta en cada pregunta de opción múltiple. Cada respuesta correcta suma 1/2 *puntos. Cada respuesta incorrecta resta* 1/6 *puntos. Las preguntas no contestadas no suman ni restan puntos.*

- Marca la respuesta a cada pregunta con una equis ("X") en la tabla de abajo.
- Si marcas más de una opción, o ninguna opción, la pregunta se cuenta como no contestada (ni suma ni resta).
- Rellena **tus datos personales** antes de comenzar a realizar el examen.

Nombre:	Grupo:

Firma:

NIA:

	A	B	C	D		A	B	C	D
1					6				
2					7				
3					8				
4					9				
5					10				

1.- Para añadir un nuevo botón a una ventana *JFrame* referenciada por la variable *frame*, se debe escribir:

(a) ***

```
frame.getContentPane().add(new JButton())
```

(b) `(new JButton()).getContentPane().add(frame)`

(c) `frame.getContentPane().add(JButton)`

(d) `getContentPane().add(new JButton())`

2.- Dado el siguiente código, selecciona la opción cierta:

```
public interface Printable {
    public void print();
}
public class Animal implements Printable {
    private String name;
}
```

(a) *** La clase *Animal* no compila, pero compilaría si se declarase como abstracta.

(b) La asignación siguiente es correcta: `Animal a = new Printable();`.

(c) Las instancias de la clase *Animal* dispondrán de una implementación del método *print()* que heredan de *Printable*.

(d) Como *Animal* implementa *Printable*, no puede sobreescribir el método *print()*.

3.- Si se invoca a un método pasándole una referencia a un objeto, y el método cambia el valor de algún atributo del objeto:

(a) El cambio será visible también desde el código que esté fuera del método, salvo que el atributo cuyo valor se cambia sea de un tipo primitivo, en cuyo caso no lo será.

(b) El cambio no será visible desde el código que esté fuera del método, salvo que el objeto que se pasa como parámetro sea abstracto, en cuyo caso lo será.

(c) El cambio no será visible desde el código que esté fuera del método.

(d) *** El cambio será visible también desde el código que esté fuera del método.

4.- Si se declara una clase con la palabra reservada *final*:

(a) Puede haber clases que hereden de dicha clase, siempre que no sobreescriban ninguno de sus métodos.

(b) *** No puede haber clases que hereden de dicha clase.

(c) Puede haber clases que hereden de dicha clase, siempre que no sobrecarguen ninguno de sus métodos.

(d) No puede haber instancias de dicha clase.

5.- Indica cuál de las siguientes afirmaciones es cierta:

(a) *** A los objetos de la clase *JPanel* se les puede añadir otros componentes mediante el método *add*.

(b) Los administradores de diseño (*layouts*) se usan para cambiar la decoración de la barra de título de las ventanas.

(c) Las instancias de *JFrame* son visibles por defecto. Se mostrarán en el momento en que se creen, salvo que se invoque a *setVisible* con parámetro *false* antes de crearlas.

(d) Una etiqueta es siempre textual, no puede mostrar una imagen.

6.- Dadas las siguientes declaraciones de clases, ¿cuál de las siguientes asignaciones no es correcta por ser los tipos de datos incompatibles?:

```
public class Estudiante extends Persona {...}
public class Profesor extends Persona {...}
public class Becario extends Estudiante {...}
```

(a) `Object o = new Profesor();`

(b) ***

```
Profesor p = new Persona();
```

(c) `Estudiante e = new Estudiante();`

(d) `Persona p = new Becario();`

7.- La sobrecarga de métodos consiste en:

(a) Programar métodos llamados igual que reciben el mismo número y tipo de parámetros, pero devuelven datos de tipos distintos.

(b) Reemplazar un método heredado por otro definido en la propia clase.

(c) Utilizar referencias de una clase para apuntar a objetos de clases que heredan de ella.

(d) *** Programar en una clase métodos que se llaman igual, pero reciben distinto número o tipo de parámetros.

8.- Se desea que un botón escuche los eventos que él mismo genera cuando el usuario lo presiona. ¿Qué hay que pasar como parámetro a *addActionListener* en el código siguiente?

```
public class MiBoton extends JButton implements ActionListener {
    private int contador;
    public MiBoton() {
        super("Me han pinchado 0 veces.");
        contador = 0;
        addActionListener(.........);
    }
    public void actionPerformed(ActionEvent e) {
        contador++;
        setText("Me han pinchado " + contador + " veces.")
    }
}
```

(a) `***`

`this`

(b) `JButton`

(c) `actionPerformed()`

(d) `new Escuchador()`

9.- Dada la clase *A* siguiente, se crea una instancia mediante `Object ref = new A()` y después se invoca el método `ref.toString()`. ¿Qué cadena de texto devuelve esta invocación?

(a) `"0, 0"`

(b) `***`

`"-1, 7"`

(c) Lo que devuelva *toString()* de la clase *Object*.

(d) `"-1, -1"`

```
public class A {
    private int a;
    private int b;
    public A(int a, int b) {
        this.a = a;
        this.b = b;
    }
    public A(int a) {
        this(a, 7);
    }
    public A() {
        this(-1);
    }
    public String toString() {
        return "" + a + ", " + b;
    }
}
```

10.- ¿Cuál de las siguientes afirmaciones acerca de interfaces es *falsa*?

(a) Una interfaz puede declarar constantes (como atributos).

(b) *** Una clase sólo puede implementar una interfaz.

(c) Si una clase implementa una interfaz, todas las clases que hereden de ella también la implementan, aunque no lo declaren explícitamente.

(d) Los métodos de una interfaz no pueden contener código.