PROGRAMACIÓN ORIENTADA A OBJETOS **1/7/2009**

Examen de Java

Nombre:_____

DNI:_____ **Titulación:** _____

1. Sea una aplicación Java donde todas las clases están compiladas y empaquetas en un solo fichero JAR (`programa.jar`). ¿Qué sería necesario para conseguir ejecutar el fichero JAR con el siguiente comando: `>java –jar programa.jar`?

2. Indica los problemas del siguiente código. Justifica la respuesta.

```
1.  public class EJ2{
2.     public static void main(String[] args){
3.       Integer[] tablaEnteros = new Integer[25];
4.       int valor = tablaEnteros[0];
5.       tablaEnteros[1] = valor+1;
6.     }
7.  }
```

3. Sea una clase `B` que hereda de otra clase `A`. Si todos los constructores de la clase `A` son privados, ¿habría algún problema en la definición de los constructores de la clase `B`? Justifica la respuesta.

4. Dado el código sin errores de compilación que se muestra a continuación, responde de manera justificada a las siguientes preguntas: a) señala los *errores de diseño* o *conceptuales* del código y b) ¿qué se imprimirá en las líneas 7 y 8?

```
 1.  class Principal {
 2.    public static void main(String[] args) {
 3.      Contador conta = new Contador();
 4.      conta.valor = 100;
 5.      int newValor = 10;
 6.      conta.update(conta, newValor);
 7.      System.out.println(conta.valor);
 8.      System.out.println(newValor);
 9.    }
10.  }
11.  class Contador {
12.    public int valor;
13.    public void update(Contador c, int v) {
14.       v = v+1;
15.       c.valor = v;
16.    }
17   }
```

5. ¿Son equivalentes las siguientes declaraciones de métodos? ¿En qué casos se utilizaría cada una de ellas? Justifica la respuesta.

```
public void addAllEntidades(Collection<Entidad> lista){…}

public void addAllEntidades(Collection<? extends Entidad> lista){…}
```

6. Sea un método declarado con el modificador `static`, ¿tiene alguna limitación para el acceso (lectura y modificación) al siguiente atributo declarado en la misma clase?: `protected final int x;`

7. Supuesto una clase `A` que contiene la definición de un método `met` con visibilidad por defecto, si la clase `B` hereda de `A` y redefine el método `met` ¿Qué modificadores de visibilidad se pueden utilizar en la redefinición de `met` en `B`? Justifica la respuesta.

8. Si `(obj.getClass()==A.class)` es igual a `true` ¿Es posible que el tipo dinámico de `obj` sea distinto de `A`? ¿Y el tipo estático? Justifica la respuesta.

9. Indica dos clases de la biblioteca de colecciones de Java que implementan la interfaz `Set`. ¿Qué diferencia la implementación de ambas clases?

10. Dado el siguiente código sin errores de compilación, ¿existe algún problema en la redefinición del método `equals`? En el caso de que lo haya indica cómo se arreglaría. Sabiendo que en la clase `A` está redefinido el método clone llamando a la copia superficial de la clase `Object` ¿Cuál es el tipo dinámico de la variable `copia` en el método `clone`? Justifica las respuestas.

```
public class B extends A{
   private C at = new C();

   public boolean equals (B otroObj){
      return super.equals(otroObj) &&
               otroObj.at.equals(at);
   }

   public A clone(){
      A copia = super.clone();
      return copia;
   }
}
```

11. Sabiendo que la clase `A` es una clase abstracta ¿es necesario que la clase `A` tenga métodos abstractos? ¿es necesario que `A` defina algún constructor? Justifica las respuestas.

12. ¿Qué papel tiene el método `setUp` en las pruebas `JUnit`? Justifica la respuesta.

13. Sabiendo que `FileNotFoundException` es subclase de `IOException` y se lanza cuando no se encuentra un fichero. ¿Cuál es el resultado de la ejecución del siguiente código si el fichero `"escenario.txt"` no existe? ¿y en el caso de que haya un error de configuración? Justifica las respuestas.

```
package aplicacion.util;

public class Configurador {

    public static Map<String,String> procesar (String fichero) throws
                  ErrorConfiguracion, FileNotFoundException{…}

}
```

```
import aplicacion.util.*;

public class Controlador {
   private Map<String,String> mapa;

   public void cargarConfiguracion(String fichero){
      try{
         mapa = Configurador.procesar(fichero);
      }catch (IOException e1){
          System.out.println("Error entrada/salida");
      }catch (Exception e2){
          System.out.println("Error al procesar fichero");
      }
      finally{
          System.out.println("revise el fichero");
      }
   }

   public static void main (String[] args){
       Controlador control  = new Controlador();
       control.cargarConfiguracion("escenario.txt");
   }
}
```

14. Dada la siguiente interfaz, ¿puede una clase implementar la interfaz y modificar `valor`? ¿Qué visibilidad tiene el método `met()`? Justifica las respuestas.

```
package examen;

public interface IPrueba {

   int valor = 3;
   void met();
}
```

15. Supuestas las clases `Clase1` y `Clase2` definidas en ficheros diferentes dentro del mismo paquete, señala los errores, si los hay, del siguiente código. Justifica la respuesta.

```
1. public class Clase1 {
2.   public float met(float a, float b)throws IOException {
3.       ...
4.   }
5. }
```

```
1. public class Clase2 extends Clase1 {
2.   public float met(float p, float q) {...}
3.   public int met(int a, int b) throws Exception {...}
4.    ...
5. }
```