

**PRUEBA 3 PROGRAMACIÓN
Junio 2007
INGENIERÍA INFORMÁTICA**



UNIVERSIDAD CARLOS III DE MADRID

LEA ATENTAMENTE ESTAS INSTRUCCIONES ANTES DE COMENZAR LA PRUEBA:

- Rellene todas las hojas a bolígrafo, tanto los datos personales como las respuestas. No use bolígrafo rojo.
- No olvide rellenar el NIA y el grupo real al que pertenece.
- El tiempo máximo de realización es de 1 hora.
- El único material permitido sobre la mesa es la hoja de test y un bolígrafo

NO PASE DE ESTA HOJA, hasta que se le indique

Apellidos	Nombre	
Firma	NIA	Grupo

PARTE 1: CUESTIONES

Pregunta 1 (1 Punto).- Indicar si la siguiente afirmación es cierta, y explicar brevemente por qué.

“El constructor de la clase File se puede usar para crear tanto directorios como ficheros”

Pregunta 2 (1 Punto).- Indicar si la siguiente afirmación es cierta, y explicar brevemente por qué.

“Se produzca o no una excepción, todas las sentencias de código de un bloque try/catch se ejecutarán”

Pregunta 3 (1 Punto).- Indicar si la siguiente afirmación es cierta, y explicar brevemente por qué.

“El orden en el que se colocan las sentencias catch dentro de un bloque try es irrelevante ya que Java buscará la más conveniente en caso de producirse una excepción.”

Pregunta 4 (1 Punto).- Implementar un método que copie el contenido de un fichero que se recibe como parámetro de entrada en un nuevo fichero llamado "nuevo.dat". Se deben gestionar las excepciones pertinentes.

Pregunta 5 (1 Punto).- Dado el siguiente método

```
public class ExceptionA extends Exception {}
public class ExceptionB extends Exception {}
public class ExceptionC extends ExceptionA {}
public class ExceptionD extends ExceptionA {}

public class Cuestion {
    public void metodo1(int x) throws ExceptionA, ExceptionB{
        if (x%2==0) metodo2(x);
        else metodo3(x);
    }
    public void metodo2(int x) throws ExceptionA{
        if (x<0) throw new ExceptionC();
        else if (x>6) throw new ExceptionD();
    }
    public void metodo3(int x) throws ExceptionB{
        if (x<0) throw new ExceptionB();
    }
}

public class Cuestion3 {
    public static void main (String [] args){
        Cuestion c = new Cuestion();
        try{
            c.metodo1(7);
            c.metodo1(8);
            c.metodo1(-2);
        }
        catch (ExceptionD e){
            System.out.println("D");
        }catch (ExceptionC e){
            System.out.println("C");
        }catch (ExceptionA e){
            System.out.println("A");
        }catch (ExceptionB e){
            System.out.println("B");
        }finally {
            System.out.println("adios");
        }
    }
}
```

Explicar la salida por pantalla tras ejecutar la clase Cuestion3.

PARTE 2: PROBLEMAS

Problema 1 (3 Puntos).- Dado el siguiente código java:

```
public class Felino {  
    private String especie;  
    private int edad;  
    private String colorPelaje;  
    protected static int numeroEjemplares;  
    private int id;  
  
    public Felino (String esp, int ed, String color){  
        especie = esp;  
        edad = ed;  
        colorPelaje = color;  
        numeroEjemplares++;  
        id = numeroEjemplares;  
    }  
  
    public Felino (){  
        this ("indeterminado",0,"indeterminado");  
    }  
}
```

1. Crear la clase `Leopardo` que hereda de `Felino` y tiene dos atributos privados: `pelajeManchado` de tipo booleano que indica si el leopardo tiene o no manchas en el pelaje, y un atributo `habitat` de tipo `String`.
2. Crear un constructor para la clase `Leopardo` que recibe valor para todos los parámetros (incluidos los heredados). Hacer otro constructor por defecto sin parámetros que use el constructor con parámetros anterior.
3. Modificar la clase `Leopardo` y la clase `Felino` para que se puedan serializar los objetos de estas clases.
4. Crear un método `public void guardar (String fichero)` en la clase `Leopardo` que guarde un objeto de la clase `Leopardo` en el fichero que se le pase por parámetro. Se deberán gestionar las excepciones pertinentes.
5. Crear un método `public Leopardo leer (String fichero)` de la clase `Leopardo` que lea un objeto de la clase `Leopardo` del fichero que se le pase por parámetro. Se deberán gestionar las excepciones pertinentes.

Problema 2 (2 Puntos).- Dado el siguiente código java:

```
import java.io.*;
public class Problema3 {
    public static void main(String [] args){
        Jarra j1 = new Jarra();
        String linea;
        int litros;
        try{
            BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
            System.out.println(" Con cuantos litros quiere llenar la
jarra: ");
            linea = br.readLine();
            litros = Integer.parseInt(linea);
            j1.llenar(litros);
        }catch (IOException e) {
            System.out.println(e.toString());
        }catch (NumberFormatException e){
            System.out.println(e.toString());
        }
    }
}
```

1. Crear una excepción propia denominada `VolumenNegativo` que tenga como atributo privado un entero llamado `volumen`.
2. Crear un constructor para esta excepción que reciba como parámetro el valor del atributo `volumen`.
3. Crear una clase `Jarra` con dos atributos enteros que indiquen la máxima capacidad en volumen que tiene un objeto `Jarra` y la cantidad de litros que contiene actualmente una `Jarra`.
4. Crear dos constructores de la clase `Jarra`. Uno de ellos sin parámetros tal que cree una jarra vacía con una capacidad máxima de 3 litros. El otro constructor deberá crear la jarra vacía con una capacidad máxima dada por parámetro.
5. Crear dos métodos de la clase `Jarra`: `vaciar` y `llenar`. El primero vaciará la jarra completamente y el segundo llenará la jarra con el volumen que se le indique por parámetro. Si el volumen es negativo, este método lanzará una excepción de tipo `VolumenNegativo`. Si el volumen no es negativo, la jarra se llenará teniendo en cuenta los litros que ya tuviera y hasta un máximo de su capacidad.
6. Debido al nuevo código creado en los apartados anteriores, el código java facilitado no compila. ¿Por qué? Añadir el código que falta.