

**PRUEBA 3 PROGRAMACIÓN  
Mayo 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 qué dos clases de la librería estándar de java se utilizan para leer y escribir ficheros de texto de manera secuencial.

Para leer: `java.io.FileReader`

Para escribir: `java.io.FileWriter`

---

**Pregunta 2 (1 Punto).**- Explicar brevemente por qué aunque en un método pueda lanzarse una excepción de tipo `NullPointerException` no es necesario declararlo explícitamente en la definición del método mediante la cláusula *throws*.

En java no es necesario declarar que se pueden lanzar explícitamente (mediante la sentencia *throws*) todas las excepciones que heredan de la clase `Error` o de la clase `RuntimeException`.

`NullPointerException` hereda directamente de `RuntimeException`, por lo tanto no es necesario.

---

**Pregunta 3 (1 Punto).**- En java, todo objeto de una clase que herede de la clase *Exception* puede ser capturado por una sentencia *catch*.

Verdadero. En java, todo objeto de una clase que herede de `Throwable` puede ser capturado por una sentencia *catch*. La clase `Exception` hereda de `Throwable`, por tanto todos los objetos de las subclases de `Exception` podrán ser capturados.

**Pregunta 4 (1 Punto).**- Dado el siguiente código java:

```
public static void leer() throws IOException {
    String linea;
    FileReader fr = null;
    BufferedReader br = null;
    try{
        fr = new FileReader("entrada.txt");
        br = new BufferedReader(fr);
        linea = br.readLine();
        while(linea != null) {
            System.out.println("Leido: " + linea);
            int numero = Integer.parseInt(linea);
            for(int i=0; i<numero; i++) {
                br.readLine();
            }
            linea = br.readLine();
        }
        br.close();
    }
    catch(Exception e){
        System.out.println("Se ha producido una excepción");
    } finally {
        if(br != null) {
            br.close();
        }
    }
}
```

y dado el fichero “entrada.txt” con el contenido siguiente:

---

```
1
hipopotamo
2
1
unicornio
jirafa
2
1
```

---

Escribir cuál sería la salida por pantalla tras invocar al método *leer()*

Respuesta:

Leido: 1

Leido: 2

Leido: jirafa

Se ha producido una excepción

---

**Pregunta 5 (1 Punto).- Dada la siguiente clase java:**

```
public class PreguntaCinco {  
    public static void main(String[] args) {  
        try {  
            metodol();  
        } catch(Exception e) {  
            System.out.println("Paso A");  
        }  
    }  
  
    public static void metodol() {  
        Integer b = null;  
        try {  
            int[] c = new int[] {1, 2};  
            System.out.println("Paso B");  
            c[2] = 3;  
            int a = 10 / 0;  
            b.toString();  
            System.out.println("Paso C");  
        } catch(NullPointerException npe) {  
            System.out.println("Paso D");  
        } catch(ArrayIndexOutOfBoundsException aiob) {  
            System.out.println("Paso E");  
        } finally {  
            System.out.println("Paso F");  
        }  
        System.out.println("Paso G");  
    }  
}
```

Escribir cuál sería la salida por pantalla tras ejecutar la clase PreguntaCinco.

Paso B

Paso E

Paso F

Paso G



**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.

```
import java.io.*;

public class Leopardo extends Felino implements Serializable {
    private boolean pelajeManchado;
    private String habitat;

    public Leopardo(String esp, int ed, String color, boolean
pelajeManchado, String habitat) {
        super(esp, ed, color);
        this.pelajeManchado = pelajeManchado;
        this.habitat = habitat;
    }

    public Leopardo() {
        this("leopardo", 0, "amarillo", true, "sabana");
    }

    public void guardar(String fichero) throws IOException {
        FileOutputStream fos = null;
        ObjectOutputStream oos = null;
        try {
            fos = new FileOutputStream(fichero);
            oos = new ObjectOutputStream(fos);
            oos.writeObject(this);
        } catch (IOException e) {
            System.out.println("Excepción intentando guardar el
Leopardo");
        } finally {
            if(oos != null) {
                oos.close();
            }
        }
    }

    public static Leopardo leer (String fichero) throws IOException {
        FileInputStream fis = null;
        ObjectInputStream ois = null;
        try {
            fis = new FileInputStream(fichero);
            ois = new ObjectInputStream(fis);
            Leopardo leo = (Leopardo) ois.readObject();
            return leo;
        } catch(Exception ioe) {
            System.out.println("Excepción intentando leer el
leopardo");
            return null;
        } finally {
            if(ois != null) {
                ois.close();
            }
        }
    }
}
```

**PROBLEMA2 (2 Puntos)**

Dado el siguiente código java:

```
public abstract class Mueble {
    public String nombre;
    public int numero;

    Mueble(String nombre){
        this.nombre = nombre;
        this.numero = 0;
    }
}
```

---

```
public interface Numerable {
    // Debe establecer el numero
    public int establecerNumero(int numero);
    // Debe devolver el numero
    public int devolverNumero();
    // Debe incrementar el numero
    public void incrementarNumero();
}
```

Hacer una clase **Silla** que herede de la clase Mueble, y que implemente la interfaz Numerable. Además:

1. Hacer un constructor que reciba como parámetro el nombre.
2. Hacer otro constructor que reciba como parámetros el nombre y el número.
3. Se debe permitir la posibilidad de serializar los objetos de la clase en flujos de bytes.

```
import java.io.Serializable;

public class Silla extends Mueble implements Numerable, Serializable {
    public Silla(String nombre) {
        super(nombre);
    }

    public Silla(String nombre, int numero) {
        super(nombre);
        this.numero = numero;
    }

    public int devolverNumero() {
        return numero;
    }

    public int establecerNumero(int numero) {
        this.numero = numero;
        return numero;
    }

    public void incrementarNumero() {
        numero++;
    }
}
```