

TEMA 2

Conceptos de Java para Estructuras de Datos

Fallos de ejecución: clasificación, representación y tratamiento

EJERCICIOS RESUELTOS

Ejercicio 1.- Completa la clase *Primitiva* (que almacena 6 números del 1 al 49):

```
public class Primitiva {
    private Integer numeros[];
    private int numGuardados;
    public Primitiva() { ... }
    public void guardarNumero(Integer n)
        throws ExcepcionNumeroDuplicado, ExcepcionNumeroFueraRango,
               ExcepcionCombinacionCompleta { ... }
}
```

Solución:

```
public class Primitiva {

    private Integer numeros[];
    private int numGuardados;

    public Primitiva() {
        numGuardados = 0;
        numeros = new Integer[6];
    }

    public void guardarNumero(Integer n)
        throws ExcepcionNumeroDuplicado, ExcepcionNumeroFueraRango,
               ExcepcionCombinacionCompleta {
        if (numGuardados > 5)
            throw new ExcepcionCombinacionCompleta("Combinación completa");
        if (n.intValue() < 1 || n.intValue() > 49)
            throw new ExcepcionNumeroFueraRango("El número " + n
                + " no está entre 1 y 49");
        for (int i = 0; i < numGuardados; i++) {
            if (numeros[i].equals(n))
                throw new ExcepcionNumeroDuplicado("El número " + n +
                    " ya está en la combinación");
        }
        numeros[numGuardados++] = n;
    }
}
```

Ejercicio 2.- Escribe un programa que lea por teclado seis números y los guarde en un objeto de clase *Primitiva*. El programa debe tratar las posibles excepciones que puedan aparecer

Solución:

```
import java.util.*;

public class TestPrimitiva {

    public static void main(String args[]) {
        Scanner teclado = new Scanner(System.in);
        Primitiva combinacion = new Primitiva();
        int indice = 1;
        while (indice <= 6) {
            System.out.print("Escriba el número " + indice +
                " de la combinación:");

            try {
                Integer n = new Integer(teclado.nextInt());
                combinacion.guardarNumero(n);
                indice++;
            } catch (InputMismatchException e1) {
                System.out.println("Número no válido. Vuelva a teclear");
                teclado.nextLine();
            } catch (Exception e2) {
                System.out.println(e2.getMessage() + ". Vuelva a teclear");
            }
        }
        System.out.println("Primitiva completa");
    }
}
```

Ejercicio 3.- Dada la siguiente definición de clases:

```
public class ImposibleAbrirBD extends Exception {
    public ImposibleAbrirBD() { super(); }
    public ImposibleAbrirBD(String msg) { super(msg); }
}

public class BaseDeDatos {
    public BaseDeDatos(String nombre){ ... }
    private void abrir() throws ImposibleAbrirBD { ... }
    public void inicializar() throws ImposibleAbrirBD {
        abrir();
        ...
    }
}

public class TestBaseDeDatos {
    public static void main (String args[]) throws ImposibleAbrirBD {
        BaseDeDatos bd = new BaseDeDatos("MiBD");
        bd.inicializar();
    }
}
```

Se pide modificar convenientemente el código de la clase *TestBaseDeDatos* para que haga tres intentos de inicializar la base de datos en caso de que se produzca la excepción *ImposibleAbrirBD*. Si al tercer intento no se logra inicializar la base de datos, deberá notificar al usuario que no ha sido posible abrir la base de datos sacando por pantalla un mensaje de error.

Solución:

```
public class TestBaseDeDatos {
    public static void main(String args[]) {
        int contador = 0;
        boolean hayError = true;
        BaseDeDatos bd = new BaseDeDatos("MiBD");
        do {
            try{
                bd.inicializar();
                hayError = false;
            } catch(ImposibleAbrirBD e) {
                contador++;
            }
        } while (contador < 3 && hayError);
        if (hayError)
            System.out.println("Imposible abrir la Base de Datos");
    }
}
```