Ejercicios Tema 5. Diseño y realización de pruebas

Bloque 1. Rompe el programa

Observa el siguiente método:

```
public static int calcularPrecioFinal(int precioBase, int descuento) {
   return precioBase - (precioBase * descuento / 100);
}
```

Ejercicio 1

Diseña al menos 7 casos de prueba distintos para el método calcularPrecioFinal

Para cada uno indica:

1 -

- Entrada: 100 y 0
- Resultado esperado: 100
- Oráculo: No hay descuento

2 -

- Entrada: 100 y 50
- Resultado esperado: 50
- Oráculo: Descuento del 50%

3 -

- Entrada: 200 y 25
- Resultado esperado: 150
- Oráculo: Descuento del 25% de 200

4 -

- Entrada: 100 y 100
- Resultado esperado: 0
- Oráculo: Descuento del 100%

5 -

- Entrada: 100 y -10
- Resultado esperado: 110
- Oráculo: Descuento del -10% por lo que aumenta el precio

6 -

- Entrada: 100 y 150
- Resultado esperado: -50
- Oráculo: Descuento del 150% que es mayor que 100 por lo que aumenta el precio

7 -

- Entrada: 0 y 50
- Resultado esperado: 0
- Oráculo: El resultado da 0 por el precio original de 0

Ejercicio 2

Contesta las siguientes preguntas relacionadas con el método calcularPrecioFinal

- ¿Qué ocurre si descuento es negativo? Que el precio aumenta
- ¿Y si es mayor que 100? El precio es negativo por lo que aumenta el precio
- ¿Crees que el método debería permitir valores negativos o mayores que 100? No, porque no sirven

Ejercicio 3

Propón una mejora del método calcularPrecioFinal **que haga que su comportamiento sea más seguro.**

```
public static int calcularPrecioFinal(int precioBase, int descuento) {
    if (descuento < 0 || descuento > 100) {
        throw new IllegalArgumentException("Descuento inválido");
    }
    return precioBase - (precioBase * descuento / 100);
}
```

Bloque 2. El cazador de bugs

Observa el siguiente método:

```
public static int maximo(int[] datos) {
  int max = 0;
  for (int i = 0; i < datos.length; i++) {
    if (datos[i] > max) {
      max = datos[i];
    }
  }
  return max;
}
```

Ejercicio 4

Encuentra al menos 3 entradas de datos donde el método maximo falle.

Para cada entrada indica:

1-

Le doy: [-5, -10, -3]

- Resultado obtenido: 0
- Resultado correcto: -3
- Tipo de fallo: Como el máximo es 0 al inicializarse da el error

2-

Le doy: [-1, -2, -8]

- Resultado obtenido: 0
- Resultado correcto: -1
- Tipo de fallo: El método no esta preparado para usar números negativos

3-

Le doy: []

- Resultado obtenido: 0
- Resultado correcto: Error
- Tipo de fallo: No tiene parte de control de excepciones

Ejercicio 5

Escribe una versión correcta del método maximo

```
public static int maximo(int[] datos) {
    if (datos.length == 0) {
        throw new IllegalArgumentException("Array vacío");
    }

    int max = datos[0];
    for (int i = 1; i < datos.length; i++) {
        if (datos[i] > max) {
            max = datos[i];
        }
    }
    return max;
}
```

Bloque 3. Diseña el oráculo

Observa el siguiente método:

```
public static int[] ordenar(int[] datos) {
  // algoritmo desconocido
}
```

Ejercicio 6

Define 3 propiedades que siempre debe cumplir el resultado del método ordenar.

Estar ordenado de menor a mayor, tener los mismos elementos que el array original, tener el mismo tamaño que el original.

Ejercicio 7

Explica por qué en el método ordenar es mejor usar oráculos por propiedades que valores concretos.

No se sabe el algoritmo interno, es más fácil comprobar reglas generales, sirve para cualquier tipo de datos.

Bloque 4. Caminos y decisiones

A la vista del siguiente método:

```
public static String clasificarEdad(int edad) {
  if (edad < 0) {
    return "ERROR";
  } else if (edad < 12) {
    return "NIÑO";
  } else if (edad < 18) {
    return "ADOLESCENTE";
  } else {
    return "ADULTO";
  }
}
```

Ejercicio 8

Calcula el número mínimo de tests necesarios para el método clasificarEdad. Después propón tantos tests como hayas obtenido.

4 opciones.

Si tiene -3 años da error.

Si tiene 8 años da niño.

Si tiene 17 años da adolescente.

Si tiene más como 67 da adulto.

Bloque 6. Prioriza como un profesional

Se quiere desarrollar una aplicación Web de compra online.

Ejercicio 10

Ordena de mayor a menor prioridad:

- Login 2
- Mostrar catálogo 3
- Pago con tarjeta 1
- Cambiar avatar 5
- Recuperar contraseña 4

```
package org.example;
import java.util.Scanner;
//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class Main {

    public static void main(String[] args){

        int res, precioBase, descuento;
        Scanner leer=new Scanner(System.in);

        System.out.println("Dame el precio inicial");
        precioBase=leer.nextInt();
        System.out.println("Dame el descuento");
        descuento=leer.nextInt();

        try {
             res = calcularPrecioFinal(precioBase, descuento);
            System.out.println("El precio final es: "+ res);
        } catch (IllegalArgumentException e) {
            System.out.println("ERROR FATAL: "+ e);
        }


    }

    public static int calcularPrecioFinal(int precioBase, int descuento) {

        if (precioBase < 0) {
            throw new IllegalArgumentException("Precio invalido");
        }
        if (descuento < 0 || descuento > 100) {
            throw new IllegalArgumentException("Descuento invalido");
        }
        return precioBase - (precioBase * descuento / 100);
    }


    public static int maximo(int[] datos) {
        if (datos.length == 0) {
            throw new IllegalArgumentException("Array vacío");
        }

        int max = datos[0];
        for (int i = 1; i < datos.length; i++) {
            if (datos[i] > max) {
                max = datos[i];
            }
        }
        return max;
    }
```

```
}
```