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:

- Entrada
- Resultado esperado
- Oráculo

pruebas	entrada	Resultado esperado	oraculo
CP 1	100,20	80	Valor esperado: $100 - 20\% = 80$
CP 2	100,0	100	Un descuento de 0% no cambia el precio
CP 3	100,100	0	Un descuento del 100% deja el precio a 0
CP 4	100,-10	error	Un descuento no puede ser negativo
CP 5	100,110	error	No puede ser superor al 110%
CP 6	0,50	0	Si el precio es 0 no hay descuento
CP 7	-50,10	error	El precio no puede ser negativo

Ejercicio 2

Contesta las siguientes preguntas relacionadas con el método calcularPrecioFinal

- ¿Qué ocurre si descuento es negativo?

Si el descuento es negativo se restara un numero negativo por ejemplo 100-(-10) que es 110

- ¿Y si es mayor que 100?

si es mayor de 100 dara un valor negativo

- ¿Crees que el método debería permitir valores negativos o mayores que 100?

no ya no debe existir la posibilidad de que el descuento sume ni que el vendedor tenga que devolverle dinero al cliente

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 (precioBase < 0 || descuento < 0 || descuento > 100) {
        throw new IllegalArgumentException("Valores fuera de rango");
    }
    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:

Entrada (datos)	Obtenido	Correcto	Tipo de fallo
{-5, -10, -2}	0	-2	max=0 es mayor que cualquier negativo de los que estan en el array
{} (vacío)	0	Excepción	lanza error ya que no hay datos
null	Crash	Excepción	el programa no gestiona la ausencia de array

Ejercicio 5

Escribe una versión correcta del método maximo

```
public static int maximo(int[] datos) {
```

```
    // Comprobar si el array es null
    if (datos == null) {
        throw new IllegalArgumentException("El array no puede ser null");
    }

    // Comprobar si está vacío
    if (datos.length == 0) {
        throw new IllegalArgumentException("El array no puede estar vacío");
    }

    // Inicializar el máximo con el primer elemento
    int max = datos[0];

    // Recorrer desde la posición 1
    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.

1.Orden: los elementos tiene que estar ordenados

2.integridad: tiene que tener el mismo tamaño que el original

3.permutación:tiene que tener el mismo contenido

Ejercicio 7

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

Porque nos permite verificar si el algoritmo es correcto sin dependencia sobre datos específicos ya que si utilizamos valores concretos se considera caso aislado y no regla general

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.

Hay 4 tests necesarios

(edad<0) edad=-1 -->da error

(edad<12) edad=5 --> niño

(edad<18) edad=15 --->adolescente

(edad base) edad=-25 --->adulto

Bloque 6. Prioriza como un profesional

Se quiere desarrollar una aplicación Web de compra online.

Ejercicio 10

Ordena de mayor a menor prioridad:

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