Ejercicios

Ejercicio1. Valida Números

Escribir una clase InOK con los métodos que permitan:

a) Leer un número entero (LeeInt).

b) Leer un número entero sólo positivo (LeeIntPos).

c) Leer un número entero en un rango determinado (LeeIntRango).

d) Leer un número real (LeeDou).

e) Leer un número real en un rango determinado (LeeDouRango).

Se deben prever todos los errores posibles que puedan suceder y tratarlos en los métodos correspondientes.

Ejercicio2. Valida String

Ampliar la clase InOK del ejercicio 1 para que lea un String y que compruebe que coincide con alguno de los elementos de un String[] añadidos anteriormente.

En caso de que el String leído exista en el array, el método deberá devolver el índice con la posición en el array del String leído o, en caso de no existir, deberá lanzar la excepción **ElementoNoExistente** definida previamente.

Añade esta declaración constante a la clase:

```
public static final String[] COMPOSITORES = {"Bach", "Haydn", "Mozart", "Beethoven",
"Brahms", "Mahler", "Bartok"};
```

El método deberá, utilizar el array COMPOSITORES. Dentro del método debe leer un String desde el teclado y, o devolverá la posición del texto encontrado en el array o lanzará la excepción correspondiente (ElementoNoExistente).

Ejercicio3. Pieza

Creamos una clase llamada Pieza que contiene los atributos String forma, String color y el método double area(). Añadimos el siguiente **método equals** :

```
public boolean equals(Object o) {
	Pieza p = (Pieza) o;
	return this.color.equals(p.color) &&
		this.nombre.equals(p.nombre) &&
		this.area() == p.area();
}
```

Este método compara dos objetos Pieza usando el método equals en Strings (forma y color) y comparando los valores Double de las áreas.

Si probamos el siguiente código falla:

```
Pieza p1 = new Pieza("cuadrado","rojo");
Pieza p2 = new Pieza("cuadrado","rojo");
Double d = new Double(1.0);
String k = "Hola";

boolean b1 = p1.equals(p2);
boolean b2 = d.equals(k);
boolean b3 = k.equals(p2);
boolean b4 = p1.equals(d);
```

Averigua porqué falla y soluciónalo tratando la excepción

Ejercicio4. Division

Crea un programa que intente realizar en el main una división entre 2 números enteros introducidos por teclado y que capture cualquier tipo de excepción (por ejemplo, que el usuario no teclee números enteros o que el denominador sea distinto de 0).

Usa la excepción de forma genérica con:

```
catch(Exception  e) {
	System.out.println("Problemas con la division");
}
```

Crea un segundo programa modificando el anterior. Ahora deberá imprimir un mensaje diferente para los diferentes errores:

- Cuando se teclee un dato no numérico → "InputMismatchException"
- Cuando el denominador sea distinto de 0 → "ArithmeticException"
- Otros posibles errores → mostrara el mensaje genérico

Piensa en cómo se deben poner el tratamiento de las tres excepciones.

Ejercicio5. Throws

Modifica el programa anterior para que la excepción de la división entre 0 se traslade a un subprograma que se llamará desde el programa principal.

Para ello, acuérdate que este subprograma tendrá un bloque try-catch donde hará la división y, en caso de producirse la excepción, incluirá la palabra reservada "throw" en el cuerpo del subprograma y "throws" en la cabecera para indicar al programa principal que el módulo ha fallado.

Ejercicio6. Vector

Realiza un programa que inicialice los valores de un vector de 5 enteros (indexado por tanto de 0 a 4) y que lance una excepción cuando intentemos asignar un valor en la posición 3. Si no conoces la excepción, ejecútalo para saber cuál es.