Unidad-9: Gestión de Excepciones.
ACTIVIDADES I

1. Hacer un método que recibe un entero, la edad de una persona, y devuelve la cantidad de años que le faltan a dicha persona para jubilarse. Imaginamos que la gente aún se jubila a los 65 años. El método puede provocar dos errores:

- EsMenorDeEdadException, si la edad introducida es de un menor de edad.
- YaEstaJubiladoException, si la edad introducida es mayor de 65

Crear el método y probarlo desde un método main, donde la edad se pida por Scanner

MEJORA 1: El método no devuelve el numero de los años que faltan para jubilarse, sino que devuelve el valor de un enumerado.

- El enumerado tiene los valores APRENDIZ, EXPERTO, NOTEFALTAMUCHO.
- El método devuelve APRENDIZ si faltan más de 20 años para jubilación.
- El método devuelve NOTEFALTAMUCHO si faltan menos de 5 años para la jubilación.
- El método devuelve EXPERTO en los demás casos

2. Crear un método que recibe dos números y calcula la diferencia entre ambos. Emite excepciones si:

- Algún número es menor que cero (MenorQueCeroException).
- La diferencia entre ambos números es mayor de 50 (DiferenciaErrorException)

Probar el método mediante la llamada desde otra clase.

3. Hacer un método que recibe un *array*, pide al usuario dos posiciones del mismo, y devuelve la división de los números que estén en dichas posiciones en el *array*. Deben controlarse en el método estas circunstancias:

- Debe controlar que al pedir el valor por teclado, sea realmente un numero.
- Debe controlar que las posiciones del *array* existen y tienen valor.
- Debe controlar que no se puede dividir por 0

Si se produce alguna de estas 3 circunstancias, el método devuelve 0.

4. Hacer varios métodos sumar, restar, multiplicar y dividir, que reciben dos enteros por parámetros y devuelven la operación correspondiente con esos dos números. Todos los métodos deben devolver una excepción personalizada si alguno de los parámetros es negativo. El método dividir debe devolver una *Exception* (de la clase Java *Exception*) con mensaje personalizado, cuando el divisor sea un 0.

5. Se tiene la siguiente clase:

```
import java.util.Scanner;
public class Excepciones {
	public static void main(String[] args){ final int NUM = 5;
		int[] enteros = new int[NUM];
		int posicion = 0;
		Scanner sc = new Scanner(System.in);
		int cont = 0;
		int divisor = 0;
		while(cont < NUM){
			System.out.print("Introduce una posición del array:");
			posicion = Integer.parseInt(teclado.nextLine());
			System.out.print("Introduce un divisor:");
			divisor = Integer.parseInt(teclado.nextLine());
			enteros[posicion] = 5 / divisor;
			cont++;
		}
		System.out.println("El contenido del array de enteros es:");
		for(int valor : enteros){
			System.out.println(valor);
		}
	}
}
```

Modificar el código para que el programa capture las excepciones que se pudieran producir, mostrando los mensajes correspondientes a cada error para después continuar con la ejecución del programa.

6. Realizar un método leerInt() que, utilizando la clase Scanner, permita repetir la petición de datos por teclado de un número hasta que lo introducido por el usuario sea realmente un número entero válido. Guardar el método como estático en una clase de Utilidades.

Crear una clase Prueba con un main que llame y compruebe el funcionamiento del método anterior.

7. Crear la clase Alumno con los siguientes atributos

```
private String nombre;
private int edad;
private double nota;
```

Y el siguiente constructor, que deberá lanzar una excepción cuando la edad sea inferior a 0.

```
public Alumno(String nombre, int edad) ...
```

La clase tendrá los siguientes métodos:

```
public double getNota()
public void setNota(double nota) public String getNombre()
public int getEdad()
```

El método setNota no deberá asignar notas inferiores a 0 o superiores a 10, y se protegerá lanzando una excepción cuando la nota sea errónea.

Crear una clase con un método main que lea desde teclado un *array* de 5 alumnos con sus notas y posteriormente muestre la información de cada uno de ellos.

Se deberán capturar todas las excepciones que se pudieran generar y mostrar los mensajes correspondientes a cada tipo de excepción.

MEJORA 1: Añade las siguientes excepciones:

- EdadNoValidaException : Añadirla al código anterior para que sea lanzada cuando se intenta construir un alumno con edad inferior a 0. Asígnale un mensaje de error.
- NotaNoValidaException, que será lanzada cuando la nota pasada como parámetro al método setNota no sea válida (inferior a 0 o superior a 10). Asígnale un mensaje de error.
- Modificar los bloques try-catch del método main del código anterior para que sean capturadas.