**Ejercicio 1: Test de una Clase de Usuario**

**Descripción:** Crea una clase Usuario que contenga los siguientes atributos y métodos:

* String nombre
* int edad
* String obtenerNombre()
* void establecerNombre(String nombre)
* int obtenerEdad()
* void establecerEdad(int edad) (debe lanzar una excepción si la edad es negativa)

**Enunciado:**

1. Implementa la clase Usuario.
2. Escribe pruebas unitarias para verificar:
   * La correcta asignación y obtención del nombre.
   * La correcta asignación y obtención de la edad.
   * La excepción al intentar establecer una edad negativa.
   * Investiga BeforeAll y AfterAll

@BeforeAll es una anotación en JUnit 5 que se utiliza para indicar que un método debe ser ejecutado una vez antes de que se ejecuten todos los métodos de prueba en una clase de prueba. Este método suele usarse para realizar tareas de configuración que son costosas o que solo necesitan hacerse una vez, como establecer conexiones de base de datos, inicializar recursos compartidos, o configurar datos iniciales necesarios para todas las pruebas.

@AfterAll es una anotación en JUnit 5 que se utiliza para indicar que un método debe ser ejecutado una vez después de que se hayan ejecutado todos los métodos de prueba en una clase de prueba. Este método es útil para realizar tareas de limpieza o liberación de recursos que fueron utilizados durante las pruebas, como cerrar conexiones de base de datos, eliminar archivos temporales, o liberar memoria.

**Ejercicio 2: Test de una Clase Banco**

**Descripción:** Crea una clase Banco que maneje cuentas bancarias. Cada cuenta tiene un número de cuenta y un saldo. Los métodos incluyen:

* void agregarCuenta(String numeroCuenta, double saldoInicial)
* double consultarSaldo(String numeroCuenta) (debe lanzar una excepción si la cuenta no existe)
* void depositar(String numeroCuenta, double monto) (debe lanzar una excepción si la cuenta no existe o si el monto es negativo)
* void retirar(String numeroCuenta, double monto) (debe lanzar una excepción si la cuenta no existe, si el monto es negativo, o si el saldo es insuficiente)

**Enunciado:**

1. Implementa la clase Banco.
2. Escribe pruebas en JUnit para:
   * La adición de cuentas con saldo inicial.
   * La consulta de saldo de una cuenta existente y la excepción para cuentas inexistentes.
   * Los depósitos en cuentas existentes y la excepción para montos negativos.
   * Los retiros de cuentas existentes, incluyendo la excepción para montos negativos y saldos insuficientes.