PROGRAMA: Ingeniería de software		CURSO: Pruebas II	ACTIVIADAD: Trabajo Autónomo
Fecha de Entrega	**miércoles 03 de Septiembre 23:59**	Junit assertEquals, assertNotEquals, assertTrue, assertFalse, assertNull, assertNotNull	DESARROLLO: INDIVIDUAL
		FUNDACIÓN UNIVERSITARIA COMPENSAR – UCOMPENSAR	
"Los problemas no se pueden resolver con el mismo nivel de pensamiento que los creó" – Albert Einstein			

Pruebas Unitarias Básicas con JUnit 5

El obejtivo del taller es familiarizarse con la creación de pruebas unitarias en Java utilizando el framework JUnit 5 y Maven. A lo largo de este taller, aprenderás a utilizar las aserciones más comunes para validar el comportamiento del código.

Herramientas:

- JDK 21 o superior.
- Maven.
- Tu IDE favorito (**IntelliJ IDEA**, Eclipse, VSCode).

Instrucciones:

Para cada ejercicio, se proporciona una clase Java con un método específico. Tu tarea es crear una clase de prueba y escribir los tests necesarios para cubrir los requerimientos descritos, utilizando las aserciones de JUnit 5.

Entregables:

- En un único proyecto Java debes generar una clase para cada ejercicio con su respectiva prueba.
- La documentación o explicación para cada clase de prueba en un archivo PDF.

PROGRAMA: Ingeniería de software		CURSO: Pruebas II	ACTIVIADAD: Trabajo Autónomo
Fecha de Entrega	**miércoles 03 de Septiembre 23:59**	Junit assertEquals, assertNotEquals, assertTrue, assertFalse, assertNull, assertNotNull	DESARROLLO: INDIVIDUAL
		FUNDACIÓN UNIVERSITARIA COMPENSAR – UCOMPENSAR	
"Los problemas no se pueden resolver con el mismo nivel de pensamiento que los creó" – Albert Einstein			

compensar
fundación universitaria

1. **Se necesita verificar el funcionamiento de una calculadora básica que solo puede sumar dos números enteros.**

```
public class Calculadora {  no usages
    public int sumaEnteros (int a, int b) {
        return a + b;
    }
}
```

Requerimientos de la Prueba:

- Verifica la suma de dos números positivos.
- Usa assertEquals para comparar el resultado esperado con el obtenido.

2. **Se requiere una función que determine si una persona es mayor de edad (18 años o más).**

```
package edu.ucompensar;

public class ValidadorEdad {
    public boolean esMayorEdad(int edad) {
        return edad >= 18;
    }
}
```

Requerimientos de la Prueba:

- Prueba con una edad mayor a 18 (ej. 25).
- Prueba con una edad menor a 18 (ej. 17).

3. **Se tiene una función que concatena dos cadenas de texto. Se necesita verificar que la concatenación funciona como se espera y que no produce resultados incorrectos.**

```java
package edu.ucompensar;

public class comparadorCadenas {
    public String Concatenar(String str1, String str2) {
        return str1.concat(str2);
    }
}
```

Requerimientos de la Prueba:

- Verifica que el resultado de concatenar "JUnit" y "5" es igual a "JUnit5".
- Verifica que el resultado de concatenar "hola" y "mundo" **no es igual** a "hola mundo ".

4. **Se desea una función que devuelva un nombre de usuario. Sin embargo, si el nombre de usuario es "admin", la función debe devolver null por razones de seguridad.**

```java
public class ProcesadorNombresUsuario {
    public String nombre(String name) {
        if ("admin".equalsIgnoreCase(nombre)) {
            return null;
        }
        return nombre;
    }
}
```

Requerimientos de la Prueba:

- Prueba que si se ingresa "admin", el resultado es nulo.
- Prueba que si se ingresa cualquier otro nombre (ej. "john"), el resultado **no es** nulo.

5. **Una tienda aplica un descuento del 10% a compras mayores o iguales a $100000 "Cien mil". Se necesita una función que calcule el precio final.**

```java
package edu.ucompensar;

public class CalcularDescuento {
    public int aplicarDescuento(BigDecimal precio) {
        if (precio.compareTo(new BigDecimal("10000")) >= 0) {
            BigDecimal descuento = precio.multiply(new BigDecimal("0.10"));
            return precio.subtract(discount);
        }
        return precio;
    }
}
```

Requerimientos de la Prueba:

- Verifica que a un precio de $200000 se le aplica el descuento correctamente.
- Verifica que a un precio de $90000 no se le aplica ningún descuento.
- Utiliza assertEquals para comparar los BigDecimal.

6. **Crear una función que verifique si una palabra es un palíndromo (se lee igual de izquierda a derecha y de derecha a izquierda), ignorando mayúsculas, minúsculas y espacios.**

```java
public class ValidarPalindromos {
    public boolean esPalindrome(String texto) {
        if (texto == null) {
            return false;
        }
        String limpiandoTexto = texto.replaceAll("\\s+", "").toLowerCase();
        String textoInverso = new StringBuilder(limpiandoTexto).reverse().toString();
        return limpiandoTexto.equals(reversedText);
    }
}
```

Requerimientos de la Prueba:

- Prueba con una palabra que es un palíndromo (ej. "Anita lava la tina").
- Prueba con una palabra que no es un palíndromo (ej. " Filokallianthropía").
- Prueba con un null. El resultado debería ser falso.

7. **Se tiene un gestor de tareas que puede agregar tareas a una lista. Se necesita una función que devuelva la lista de tareas. Si no se ha agregado ninguna tarea, la lista debe ser null.**

```java
package edu.ucompensar;

import java.util.ArrayList;
import java.util.List;

public class AdministradorTareas {
    private List<String> tareas;

    public void adicionarTarea(String tarea) {
        if (this.tareas == null) {
            this.tareas = new ArrayList<>();
        }
        this.tareas.add(tarea);
    }

    public List<String> optenerTarea() {
        return this.tareas;
    }
}
```

- Verifica que, sin agregar ninguna tarea, el método optenerTarea() devuelve null.
- Verifica que, después de agregar al menos una tarea, el método optenerTarea () **no** devuelve null.

8. **Se necesita una función que valide la fortaleza de una contraseña. Una contraseña es fuerte si tiene al menos 8 caracteres y contiene al menos un número.**

```java
package edu.ucompensar;

public class ValidadorPassword {
    public boolean fuerte(String password) {
        if (password == null || password.length() < 8) {
            return false;
        }
        boolean conNumeros = false;
        for (char c : password.toCharArray()) {
            if (Character.isDigit(c)) {
                conNumeros = true;
                break;
            }
        }
        return conNumeros;
    }
}
```

Requerimientos de la Prueba:

- Prueba una contraseña fuerte (ej. "password123").
- Prueba una contraseña débil por ser corta (ej. "pass1").
- Prueba una contraseña débil por no tener números (ej. "passwordlong").

9. Se requiere una función que calcule el factorial de un número. El factorial.

```java
package edu.ucompensar;

public class FactorialCalculator {
    public long factorial(int n) {
        if (n < 0) {
            throw new IllegalArgumentException("Con Negativos nos  Funciono. . .");
        }
        if (n == 0) {
            return 1;
        }
        long result = 1;
        for (int i = 1; i <= n; i++) {
            result *= i;
        }
        return result;
    }
}
```

- Verifica que el factorial de 5 es 120.
- Verifica que el factorial de 0 es 1.
- Verifica que el factorial de 5 **no es** 121.

10. Se necesita una función que busque un usuario por su correo electrónico en una lista. Si el usuario no se encuentra, debe devolver null.

```java
package edu.ucompensar;

public class User {
    private String username;
    private String email;
    public User(String username, String email) {this.username = username; this.email = email;}
    public String getEmail() {return email;}
    public String getUsername() {return username;}
}
```

```java
package edu.ucompensar;

import java.util.List;

public class EmailFinder {
    public User findUserByEmail(List<User> users, String email) {
        if (users == null || email == null) {
            return null;
        }
        for (User user : users) {
            if (email.equals(user.getEmail())) {
                return user;
            }
        }
        return null;
    }
}
```

PROGRAMA: Ingeniería de software		CURSO: Pruebas II	ACTIVIADAD: Trabajo Autónomo
Fecha de Entrega	**miércoles 03 de Septiembre 23:59**	Junit assertEquals, assertNotEquals, assertTrue, assertFalse, assertNull, assertNotNull	DESARROLLO: INDIVIDUAL
		FUNDACIÓN UNIVERSITARIA COMPENSAR – UCOMPENSAR	
"Los problemas no se pueden resolver con el mismo nivel de pensamiento que los creó" – Albert Einstein			

Requerimientos de la Prueba:

- Crea una lista de usuarios de prueba.
- Prueba que se puede encontrar un usuario con un correo existente y que su nombre de usuario es el correcto.
- Prueba que la búsqueda de un correo electrónico que no existe devuelve null.