Homework Hustlers: https://discord.gg/aJ55rZBV

- Wizard.

Create a simple test method using @test for the class MathUtilsTest

```
~

import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class MathUtilsTest {

    @Test
    public void testAdd() {
        MathUtils ms = new MathUtils();
        assertEquals(5, ms.add(2, 3));  // test the `add` method
    }
}
```

```
~

public class MathUtils {
    public int add(int a, int b) {
        return a + b;
    }
    public static void main(String[] args) {

    }
}
```

```
~

[wizard@archlinux tuto]$ java-compile-test
[wizard@archlinux tuto]$ java-test MathUtilsTest
JUnit version 4.13.2
..
Time: 0.008

OK (2 tests)

[wizard@archlinux tuto]$
```

Use @before and @after annotations to setup and clean up resources beofre and after each test

```
~

import org.junit.Before;
import org.junit.After;
import org.junit.Test;

public class DatabaseConnectionTest {

    private DatabaseConnection ms;

    @Before
    public void testConnection() {
        ms = new DatabaseConnection();
    }

    @Test
    public void testConnectToDatabase() {
        ms.connectToDatabase(); // This would call the method you're testing
    }

    @After
    public void closeConnection() {
        System.out.println("Closed Connection");
    }
}

class DatabaseConnection {
    public void connectToDatabase() {
        System.out.println("Established connection");
    }
}
```

```
~

[wizard@archlinux tuto]$ java-compile-test
[wizard@archlinux tuto]$ java-test DatabaseConnectionTest
JUnit version 4.13.2
..Established connection
Closed Connection

Time: 0.01

OK (2 tests)

[wizard@archlinux tuto]$
```

Write test for converting temperature from Celcius to Fernheight and vice versa.

```
~

import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class TempConversionTest {

    @Test
    public void testCelsiusToFahrenheit() {
        assertEquals(32.0, tmpConversion.celciusToFer(0), 0.01);
        assertEquals(212.0, tmpConversion.celciusToFer(100), 0.01);
    }

    @Test
    public void testFahrenheitToCelsius() {
        assertEquals(0.0, tmpConversion.ferToCel(32), 0.01);
        assertEquals(100.0, tmpConversion.ferToCel(212), 0.01);
    }
}

class tmpConversion {
    public static double celciusToFer(double celsius) {
        return (celsius * 9/5) + 32;
    }

    public static double ferToCel(double fahrenheit) {
        return (fahrenheit - 32) * 5/9;
    }

}
```

```
~

[wizard@archlinux tuto]$ java-compile-test
[wizard@archlinux tuto]$ java-test TempConversionTest
JUnit version 4.13.2
...
Time: 0.011

OK (3 tests)

[wizard@archlinux tuto]$
```

Banking System - Account Balance

```
~
```

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

class BankAccount {
    private double balance;

    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        balance -= amount;
    }

    public double getBalance() {
        return balance;
    }
}

public class BankAccountTest {

    @Test
    public void testDeposit() {
        BankAccount account = new BankAccount(100);
        account.deposit(50);
        assertEquals(150, account.getBalance(), 0.01);
    }

    @Test
    public void testWithdraw() {
        BankAccount account = new BankAccount(100);
        account.withdraw(30);
        assertEquals(70, account.getBalance(), 0.01);
    }
}
```

```
[wizard@archlinux tuto]$ java-compile-test
[wizard@archlinux tuto]$ java-test BankAccountTest
JUnit version 4.13.2
...
Time: 0.008

OK (3 tests)

[wizard@archlinux tuto]$
```

Write tests for product class that calculates the total price after applying discount

```
~

import org.junit.Test;
import static org.junit.Assert.assertEquals;

class Product {
    private double price;
    private double discount;

    public Product(double price, double discount) {
        this.price = price;
        this.discount = discount;
    }

    public double calculateTotalPrice() {
        return price - (price * discount / 100);
    }
}

public class ProductTest {

    @Test
    public void testCalculateTotalPrice() {
        Product product = new Product(100, 10);
        assertEquals(90, product.calculateTotalPrice(), 0.01);

        product = new Product(200, 25);
        assertEquals(150, product.calculateTotalPrice(), 0.01);
    }
}
```

```
~

[wizard@archlinux tuto]$ java-compile-test
[wizard@archlinux tuto]$ java-test ProductTest
JUnit version 4.13.2
..
Time: 0.009

OK (2 tests)

[wizard@archlinux tuto]$
```

Test Driven Development:

What is Test-Driven-Development (TDD), and how does it differ from traditional Development practices?

TDD is a development process where you write tests before you write the code. Unlike traditional development, you make the test first, then the code.

What are the three main phases in TDD? Describle each phase.

Write a failing test: Write a test that fails because the functionality doesn’t exist yet.
Make the test pass: Write just enough code to make the test pass.
Refactor: Clean up the code while ensuring the test still passes.

What is the role of JUnit in Test-Driven Development?

Unit is a framework that helps write and run tests in Java. In TDD, it is used to create tests, and run them checking wether the tests pass or not