Name: Palanivel M

Superset ID: 6373179

Mail ID: [727822tuec135@skct.edu.in](mailto:727822tuec135@skct.edu.in)

# Week 2 Hands-On Exercise

## Skill: TDD Using Junit5 and Mockito

### Exercise 1: setting Up Junit

### Code: Temperature Converter with manual test output

public class Main {

static class TemperatureConverter {

public double celsiusToFahrenheit(double c) {

return (c \* 9.0 / 5.0) + 32;

}

public double fahrenheitToCelsius(double f) {

return (f - 32) \* 5.0 / 9.0;

}

public double celsiusToKelvin(double c) {

return c + 273.15;

}

public double kelvinToCelsius(double k) {

if (k < 0) {

throw new IllegalArgumentException("Kelvin cannot be negative");

}

return k - 273.15;

} }

public static void main(String[] args) {

TemperatureConverter converter = new TemperatureConverter();

System.out.println("Test 1: Celsius to Fahrenheit");

System.out.println(Math.abs(converter.celsiusToFahrenheit(0) - 32.0) < 0.001 ? "Passed" : "Failed");

System.out.println("Test 2: Fahrenheit to Celsius");

System.out.println(Math.abs(converter.fahrenheitToCelsius(32) - 0.0) < 0.001 ? "Passed" : "Failed");

System.out.println("Test 3: Celsius to Kelvin");

System.out.println(Math.abs(converter.celsiusToKelvin(0) - 273.15) < 0.001 ? "Passed" : "Failed");

System.out.println("Test 4: Kelvin to Celsius");

System.out.println(Math.abs(converter.kelvinToCelsius(300) - 26.85) < 0.001 ? "Passed" : "Failed");

System.out.println("Test 5: Kelvin to Celsius with negative input");

try {

converter.kelvinToCelsius(-10);

System.out.println("Failed");

} catch (IllegalArgumentException e) {

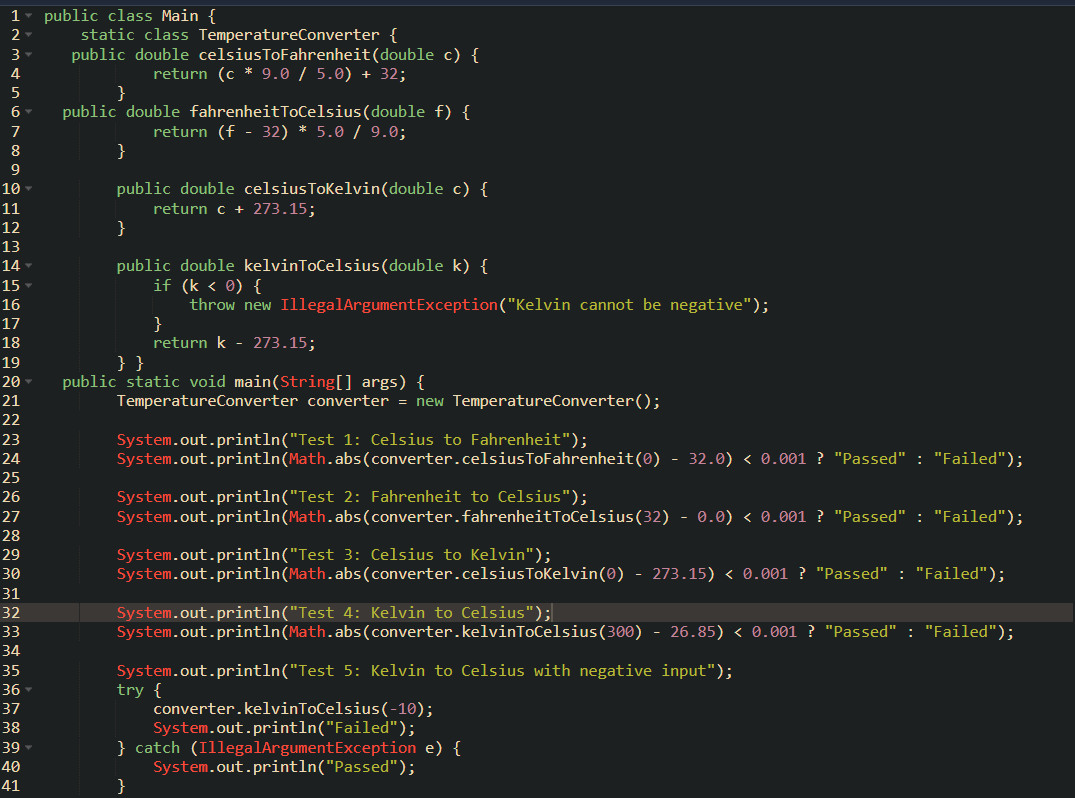
System.out.println("Passed");

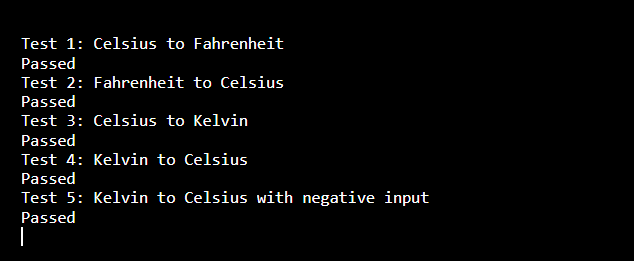
}

}

}

## Output:





## Exercise 3: Assertions in Junit

JUnit assertions like:

* assertEquals
* assertNotEquals
* assertTrue
* assertFalse
* assertNull
* assertNotNull
* assertArrayEquals

## StringUtils.java

## Code:

package com.example;

public class StringUtils {

public static boolean isEmpty(String str) {

return str == null || str.isEmpty();

}

public static String reverse(String str) {

if (str == null) return null;

return new StringBuilder(str).reverse().toString();

}

public static String[] splitWords(String sentence) {

if (sentence == null) return null;

return sentence.trim().split("\\s+");

}

}

## StringUtilsTest.java

## Code:

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class StringUtilsTest {

@Test

public void testIsEmpty() {

assertTrue(StringUtils.isEmpty(""));

assertTrue(StringUtils.isEmpty(null));

assertFalse(StringUtils.isEmpty("hello"));

}

@Test

public void testReverse() {

assertEquals("dcba", StringUtils.reverse("abcd"));

assertEquals("", StringUtils.reverse(""));

assertNull(StringUtils.reverse(null));

assertNotEquals("abcd", StringUtils.reverse("abcd")); // reversed!

}

@Test

public void testSplitWords() {

String[] expected = {"Hello", "world"};

String[] actual = StringUtils.splitWords("Hello world");

assertArrayEquals(expected, actual);

assertNull(StringUtils.splitWords(null));

String[] single = StringUtils.splitWords(" oneWord ");

assertEquals(1, single.length);

assertEquals("oneWord", single[0]);

}

@Test

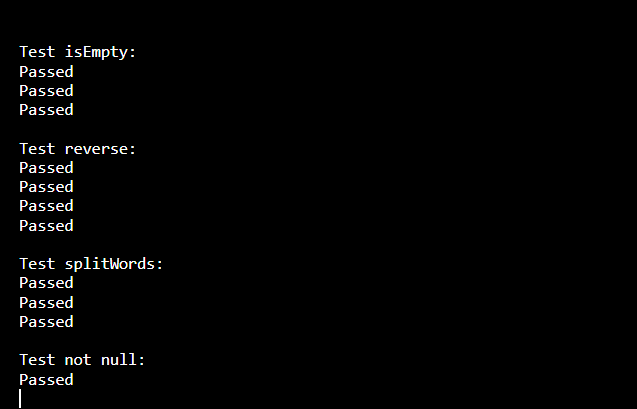
public void testNotNull() {

assertNotNull(StringUtils.reverse("test"));

}

}

## Output:



# Exercise 4:

## Arrange-Act-Asser(AAA) Pattern,Test Fixures,Setup and Teardown Method in Junit

# Code:

## BankAccount.java

package com.example;

public class BankAccount {

private String owner;

private double balance;

public BankAccount(String owner, double startingBalance) {

this.owner = owner;

this.balance = startingBalance;

}

public void deposit(double amount) {

if (amount <= 0) throw new IllegalArgumentException("Invalid deposit");

balance += amount;

}

public void withdraw(double amount) {

if (amount > balance) throw new IllegalArgumentException("Insufficient funds");

balance -= amount;

}

public double getBalance() {

return balance;

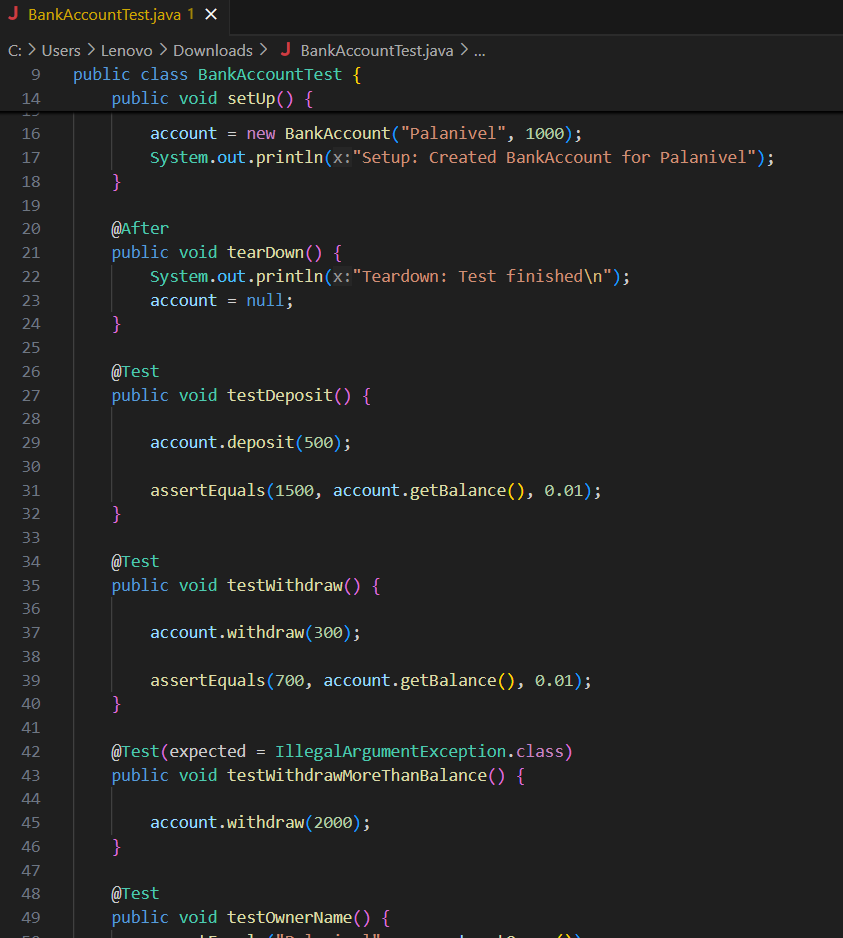
}

public String getOwner() {

return owner;

}

}



## BankAccountTest.java

package com.example;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class BankAccountTest {

private BankAccount account;

@Before

public void setUp() {

account = new BankAccount("Palanivel", 1000);

System.out.println("Setup: Created BankAccount for Palanivel");

}

@After

public void tearDown() {

System.out.println("Teardown: Test finished\n");

account = null;

}

@Test

public void testDeposit() {

account.deposit(500);

assertEquals(1500, account.getBalance(), 0.01);

}

@Test

public void testWithdraw() {

account.withdraw(300);

assertEquals(700, account.getBalance(), 0.01);

}

@Test(expected = IllegalArgumentException.class)

public void testWithdrawMoreThanBalance() {

account.withdraw(2000);

}

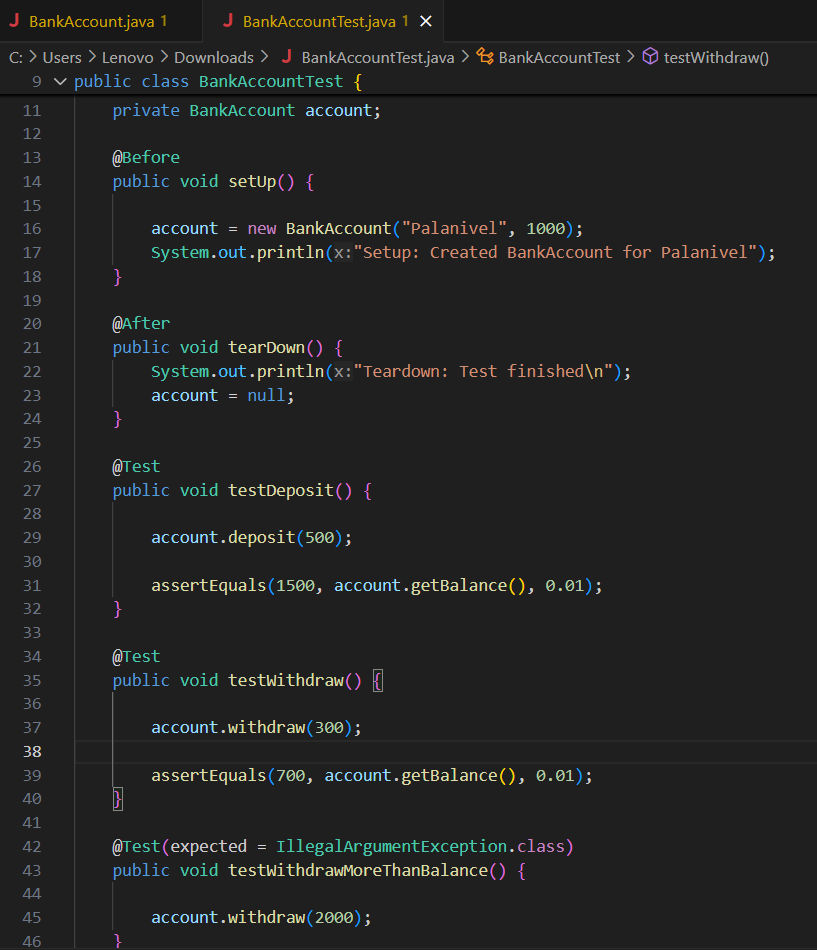
@Test

public void testOwnerName() {

assertEquals("Palanivel", account.getOwner());

}

}



## Output:

