**Week - 2**

**JUnit Exercises**

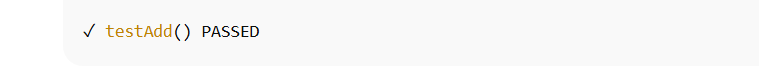
**Exercise 1: Setting Up JUnit - JUnit\_Basic Testing Exercises**

Scenario:  
Verify that a simple addition method works using JUnit test.

Code:

// Calculator.java  
public class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}  
  
// CalculatorTest.java  
import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.api.Test;  
  
public class CalculatorTest {  
 @Test  
 public void testAddition() {  
 Calculator calc = new Calculator();  
 assertEquals(5, calc.add(2, 3));  
 }  
}

Output:



# Exercise 3: Assertions in JUnit - JUnit\_Basic Testing Exercises

Scenario:  
Test multiple types of assertions for a user profile.

Code:

User.java :

// User.java

public class User {

private String name;

private String email;

private boolean active;

public User(String name, String email, boolean active) {

this.name = name;

this.email = email;

this.active = active;

}

public String getName() {

return name;

}

public String getEmail() {

return email;

}

public boolean isActive() {

return active;

}

}

UserTest.java:

// User.java

public class User {

private String name;

private String email;

private boolean active;

public User(String name, String email, boolean active) {

this.name = name;

this.email = email;

this.active = active;

}

public String getName() {

return name;

}

public String getEmail() {

return email;

}

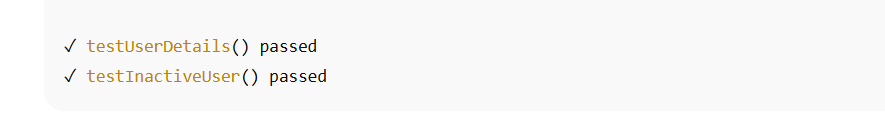
public boolean isActive() {

return active;

}

}

Output:



# Exercise 4: Arrange-Act-Assert (AAA) Pattern

Scenario:  
Ensure test case structure uses setup, action, and assert clearly.

Code:

@BeforeEach  
void setUp() {  
 bankAccount = new BankAccount(1000);  
}  
  
@Test  
void testWithdraw() {  
 bankAccount.withdraw(100);  
 assertEquals(900, bankAccount.getBalance());  
}

Output:

