**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

**Scenario:**

You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup

and teardown methods.

**Steps:**

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods.

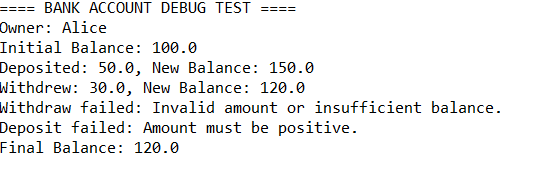
**STEP 1 – CREATING A NEW CLASS**

* Created a new class named “Testing4.java” under the Junit testing project. Used bank account as a use case.
* Main code logic is given below.

**Code:**

|  |
| --- |
| package test;  public class Testing4 {  private String owner;  private double balance;  public Testing4(String owner, double initialBalance) {  this.owner = owner;  this.balance = initialBalance;  }  public String getOwner() {  return owner;  }  public double getBalance() {  return balance;  }  public void deposit(double amount) {  if (amount > 0) {  balance += amount;  System.out.println("Deposited: " + amount + ", New Balance: " + balance);  } else {  System.out.println("Deposit failed: Amount must be positive.");  }  }  public boolean withdraw(double amount) {  if (amount > 0 && amount <= balance) {  balance -= amount;  System.out.println("Withdrew: " + amount + ", New Balance: " + balance);  return true;  }  System.out.println("Withdraw failed: Invalid amount or insufficient balance.");  return false;  }  public static void main(String[] args) {  System.out.println("==== BANK ACCOUNT DEBUG TEST ====");  Testing4 account = new Testing4("Alice", 100.0);  System.out.println("Owner: " + account.getOwner());  System.out.println("Initial Balance: " + account.getBalance());  account.deposit(50.0);  account.withdraw(30.0);  account.withdraw(200.0);  account.deposit(-10.0);  System.out.println("Final Balance: " + account.getBalance());  }  } |

**Output:**

****

**STEP 2 – CREATING A NEW TEST CLASS**

Created a new class named “Testing4test.java” under the Junit testing project.

Code:

|  |
| --- |
| **package** test;  **import** org.junit.After;  **import** org.junit.Before;  **import** org.junit.Test;  **import** **static** org.junit.Assert.\*;  **public** **class** Testing4Test {  **private** Testing4 account;  @Before  **public** **void** setUp() {  System.***out***.println("Setting up test account...");  account = **new** Testing4("Alice", 100.0);  }  @After  **public** **void** tearDown() {  System.***out***.println("Cleaning up test account...");  account = **null**;  }  @Test  **public** **void** testDepositIncreasesBalance() {  **double** depositAmount = 50.0;  account.deposit(depositAmount);  *assertEquals*(150.0, account.getBalance(), 0.01);  }  @Test  **public** **void** testDepositNegativeAmountDoesNothing() {  **double** depositAmount = -30.0;  account.deposit(depositAmount);  *assertEquals*(100.0, account.getBalance(), 0.01);  }  @Test  **public** **void** testWithdrawDecreasesBalance() {  **double** withdrawAmount = 40.0;  **boolean** success = account.withdraw(withdrawAmount);  *assertTrue*(success);  *assertEquals*(60.0, account.getBalance(), 0.01);  }  @Test  **public** **void** testWithdrawMoreThanBalanceFails() {  **double** withdrawAmount = 200.0;  **boolean** success = account.withdraw(withdrawAmount);  *assertFalse*(success);  *assertEquals*(100.0, account.getBalance(), 0.01);  }  @Test  **public** **void** testWithdrawNegativeAmountFails() {  **double** withdrawAmount = -20.0;  **boolean** success = account.withdraw(withdrawAmount);  *assertFalse*(success);  *assertEquals*(100.0, account.getBalance(), 0.01);  }  @Test  **public** **void** testGetOwner() {  String owner = account.getOwner();  *assertEquals*("Alice", owner);  }  } |

Output:

