**AAA Pattern & Test Fixtures in JUnit**

**1. Create the Class to Test**

package com.example;

public class BankAccount {

private double balance;

private String accountHolder;

public BankAccount(String accountHolder, double initialBalance) {

this.accountHolder = accountHolder;

this.balance = initialBalance;

}

public void deposit(double amount) {

if (amount <= 0) throw new IllegalArgumentException("Amount must be positive");

balance += amount;

}

public void withdraw(double amount) throws InsufficientFundsException {

if (amount <= 0) throw new IllegalArgumentException("Amount must be positive");

if (amount > balance) throw new InsufficientFundsException();

balance -= amount;

}

public double getBalance() {

return balance;

}

public String getAccountHolder() {

return accountHolder;

}

}

class InsufficientFundsException extends Exception {}

**2. Test Class with AAA Pattern & Fixtures**

java

Copy

Download

// src/test/java/com/example/BankAccountTest.java

package com.example;

import org.junit.\*;

import static org.junit.Assert.\*;

public class BankAccountTest {

// Test fixtures (shared test data)

private BankAccount account;

private static final String TEST\_HOLDER = "John Doe";

private static final double INITIAL\_BALANCE = 1000.00;

// Setup method runs before each test

@Before

public void setUp() {

account = new BankAccount(TEST\_HOLDER, INITIAL\_BALANCE);

System.out.println("Creating fresh account for each test");

}

// Teardown method runs after each test

@After

public void tearDown() {

System.out.println("Test completed. Balance: " + account.getBalance());

}

// Test 1: Successful deposit

@Test

public void testDeposit\_increasesBalance() {

// Arrange

double depositAmount = 500.00;

double expectedBalance = INITIAL\_BALANCE + depositAmount;

// Act

account.deposit(depositAmount);

// Assert

assertEquals(expectedBalance, account.getBalance(), 0.001);

}

// Test 2: Successful withdrawal

@Test

public void testWithdraw\_decreasesBalance() throws InsufficientFundsException {

// Arrange

double withdrawAmount = 200.00;

double expectedBalance = INITIAL\_BALANCE - withdrawAmount;

// Act

account.withdraw(withdrawAmount);

// Assert

assertEquals(expectedBalance, account.getBalance(), 0.001);

}

// Test 3: Withdrawal with insufficient funds

@Test(expected = InsufficientFundsException.class)

public void testWithdraw\_throwsWhenInsufficientFunds() throws InsufficientFundsException {

// Arrange

double withdrawAmount = INITIAL\_BALANCE + 100;

// Act & Assert (in one line due to expected exception)

account.withdraw(withdrawAmount);

}

// Test 4: Invalid deposit amount

@Test

public void testDeposit\_negativeAmountThrowsException() {

// Arrange

double invalidAmount = -100;

try {

// Act

account.deposit(invalidAmount);

fail("Should have thrown exception");

} catch (IllegalArgumentException e) {

// Assert

assertEquals("Amount must be positive", e.getMessage());

}

}

}