**Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and**

**Teardown Methods in JUnit**

**JUnit\_Basic Testing**

Vaishnavi

28/6/2025

This Java project uses JUnit to test a ShoppingCart class by applying the Arrange-Act-Assert (AAA) pattern. It demonstrates the use of test fixtures, including @Before for setup and @After for teardown, to ensure clean, consistent testing for shopping cart operations.

**Objective:**

* **Follow AAA Pattern:** Clearly separate test logic into three stages – Arrange (prepare data), Act (execute methods), and Assert (verify outcomes).
* **Use Setup/Teardown Methods:** Apply @Before to initialize the cart before each test and @After to clean up after each test to maintain test isolation.
* **Verify Functional Accuracy:** Test shopping cart operations like adding, removing, clearing items, and calculating totals to ensure expected behavior.

**Implementation:**

### Create a New Java Project

**IntelliJ IDEA**: File → New → Project → Java

### Add JUnit to Your Project

#### By using ****Maven****: add the dependencies in pom.xml

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

**Create a ShoppingCart Java Class and Test Class**

**ShoppingCart.java:**

import java.util.\*;

public class ShoppingCart {

private List<String> items = new ArrayList<>();

private Map<String, Double> priceList = new HashMap<>();

public void addItem(String item, double price) {

items.add(item);

priceList.put(item, price);

}

public void removeItem(String item) {

items.remove(item);

priceList.remove(item);

}

public double calculateTotal() {

return items.stream().mapToDouble(item -> priceList.getOrDefault(item, 0.0)).sum();

}

public int getItemCount() {

return items.size();

}

public void clear() {

items.clear();

priceList.clear();

}

public boolean isEmpty() {

return items.isEmpty();

}

}

**ShoppingCartTest.java:**

import static org.junit.Assert.\*;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

public class ShoppingCartTest {

private ShoppingCart cart;

@Before

public void setUp() {

cart = new ShoppingCart();

System.out.println("New cart created.");

}

@After

public void tearDown() {

cart.clear();

System.out.println("Cart cleared.\n");

}

@Test

public void testAddSingleItem() {

String item = "Book";

cart.addItem(item, 150.0);

assertEquals(1, cart.getItemCount());

assertEquals(150.0, cart.calculateTotal(), 0.001);

System.out.println("Added 1 item. Total: ₹" + cart.calculateTotal());

}

@Test

public void testAddMultipleItems() {

cart.addItem("Pen", 10.0);

cart.addItem("Notebook", 40.0);

cart.addItem("Pencil", 5.0);

assertEquals(3, cart.getItemCount());

assertEquals(55.0, cart.calculateTotal(), 0.001);

System.out.println("3 items added. Total: ₹" + cart.calculateTotal());

}

@Test

public void testRemoveItem() {

cart.addItem("Perfume", 500.0);

cart.addItem("Soap", 50.0);

cart.removeItem("Perfume");

assertEquals(1, cart.getItemCount());

assertEquals(50.0, cart.calculateTotal(), 0.001);

System.out.println("Removed 'Perfume'. Remaining total: ₹" + cart.calculateTotal());

}

@Test

public void testClearCart() {

cart.addItem("Bag", 800.0);

cart.addItem("Shoes", 1200.0);

cart.clear();

assertTrue(cart.isEmpty());

assertEquals(0.0, cart.calculateTotal(), 0.001);

System.out.println("Cart cleared manually. Total: ₹" + cart.calculateTotal());

}

@Test

public void testIsEmptyInitially() {

assertTrue(cart.isEmpty());

System.out.println("Cart is empty initially.");

}

@Test

public void testItemCountAfterAddAndRemove() {

cart.addItem("Charger", 300.0);

cart.addItem("Mouse", 700.0);

cart.removeItem("Charger");

assertEquals(1, cart.getItemCount());

System.out.println("Added 2, removed 1. Items left: " + cart.getItemCount());

}

@Test

public void testTotalCalculationAccuracy() {

cart.addItem("Pen Drive", 999.99);

cart.addItem("Headphones", 2000.50);

double total = cart.calculateTotal();

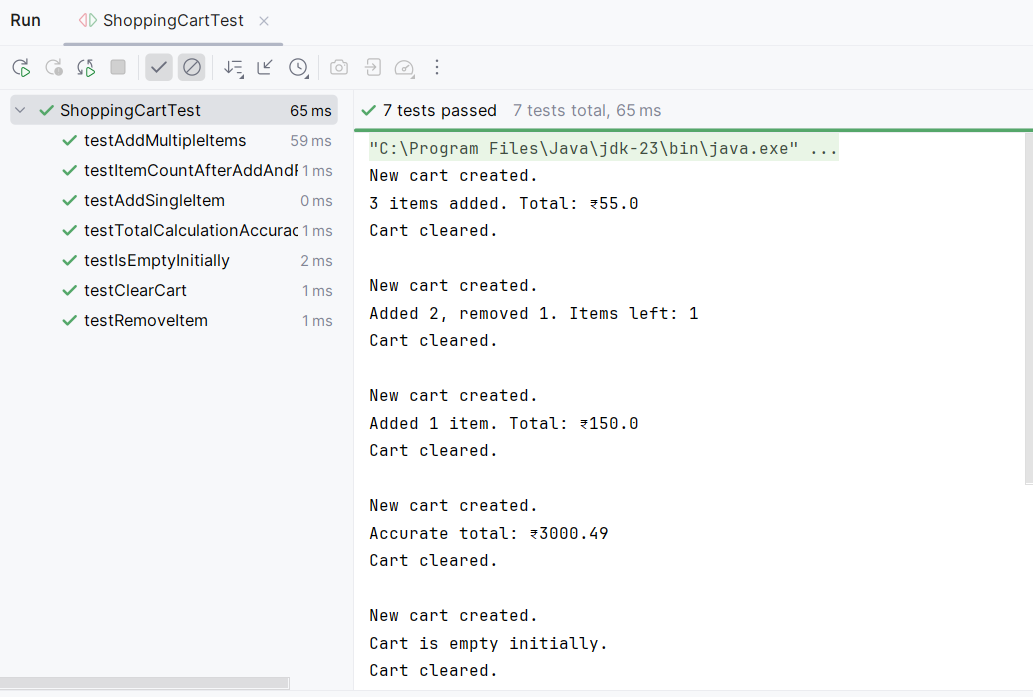
assertEquals(3000.49, total, 0.01);

System.out.println("Accurate total: ₹" + total);

}

}

**Output:**

****