**UNIT TEST CASES**

**1.Passenger class**

import org.junit.Test;

import static org.junit.Assert.\*;

public class PassengerTest {

@Test

public void testDeductBalance() {

Passenger passenger = new Passenger("John", 1, PassengerType.GOLD);

passenger.deductBalance(100);

assertEquals(90.0, passenger.getBalance(), 0.01); // Expected balance after 10% discount

}

@Test

public void testAddActivity() {

Passenger passenger = new Passenger("Emily", 2, PassengerType.STANDARD);

Activity activity = new Activity("Louvre Museum Visit", "Visit the Louvre Museum", 40, 40, new Destination("Paris"));

passenger.addActivity(activity);

assertTrue(passenger.getActivities().contains(activity)); // Check if activity is added

}

@Test

public void testGetters() {

Passenger passenger = new Passenger("David", 3, PassengerType.PREMIUM);

assertEquals("David", passenger.getName());

assertEquals(3, passenger.getPassengerNumber());

assertEquals(PassengerType.PREMIUM, passenger.getType());

}

}

Explanation of test cases for `Passenger` class:

1. `testDeductBalance`: This test checks if the balance is deducted correctly for a Gold passenger (with a 10% discount).

2. `testAddActivity`: Verifies if an activity is successfully added to the passenger's list of activities.

3. `testGetters`: Ensures that the getter methods return the expected values for the passenger's attributes.

**2. Activity Class**

import org.junit.Test;

import static org.junit.Assert.\*;

public class ActivityTest {

@Test

public void testGetCost() {

Activity activity = new Activity("Eiffel Tower Tour", "Guided tour of the Eiffel Tower", 50, 30, new Destination("Paris"));

assertEquals(50, activity.getCost(), 0.01); // Verify cost retrieval

}

@Test

public void testGetters() {

Activity activity = new Activity("Vatican Museum Visit", "Visit the Vatican Museums", 55, 35, new Destination("Rome"));

assertEquals("Vatican Museum Visit", activity.getName());

assertEquals("Visit the Vatican Museums", activity.getDescription());

assertEquals(35, activity.getCapacity());

assertEquals("Rome", activity.getDestination().getName());

}

@Test

public void testRemainingCapacity() {

Destination destination = new Destination("Paris");

Activity activity = new Activity("Eiffel Tower Tour", "Guided tour of the Eiffel Tower", 50, 30, destination);

assertEquals(30, activity.getCapacity());

destination.addActivity(activity);

assertEquals(29, activity.getCapacity()); // Verify remaining capacity after adding an activity

}

}

```

Explanation of test cases for `Activity` class:

1. `testGetCost`: Checks if the cost of the activity is returned correctly.

2. `testGetters`: Ensures that getter methods return the expected values for the activity's attributes.

3. `testRemainingCapacity`: Verifies if the remaining capacity of an activity is updated correctly when added to a destination.

**3.Destination Class**

import org.junit.Test;

import static org.junit.Assert.\*;

public class DestinationTest {

@Test

public void testAddActivity() {

Destination destination = new Destination("Paris");

Activity activity = new Activity("Eiffel Tower Tour", "Guided tour of the Eiffel Tower", 50, 30, destination);

destination.addActivity(activity);

assertTrue(destination.getActivities().contains(activity)); // Verify activity addition

}

@Test

public void testGetters() {

Destination destination = new Destination("Rome");

assertEquals("Rome", destination.getName());

assertNotNull(destination.getActivities());

}

@Test

public void testGetActivities() {

Destination destination = new Destination("Paris");

Activity activity = new Activity("Eiffel Tower Tour", "Guided tour of the Eiffel Tower", 50, 30, destination);

destination.addActivity(activity);

assertEquals(1, destination.getActivities().size()); // Verify activity retrieval

}

}

```

Explanation of test cases for `Destination` class:

1. `testAddActivity`: Verifies if an activity is successfully added to the destination's list of activities.

2. `testGetters`: Ensures that getter methods return the expected values for the destination's attributes.

3. `testGetActivities`: Checks if activities can be retrieved from the destination.

**4. TravelPackage Class**

import org.junit.Test;

import static org.junit.Assert.\*;

public class TravelPackageTest {

@Test

public void testAddPassenger() {

TravelPackage travelPackage = new TravelPackage("Europe Trip", 50);

Passenger passenger = new Passenger("John", 1, PassengerType.STANDARD);

travelPackage.addPassenger(passenger);

assertTrue(travelPackage.getPassengers().contains(passenger)); // Verify passenger addition

}

@Test

public void testAddDestination() {

TravelPackage travelPackage = new TravelPackage("Europe Trip", 50);

Destination destination = new Destination("Paris");

travelPackage.addDestination(destination);

assertTrue(travelPackage.getItinerary().contains(destination)); // Verify destination addition

}

@Test

public void testPrintItinerary() {

// This test prints the itinerary; manual inspection is required

TravelPackage travelPackage = new TravelPackage("Europe Trip", 50);

// Add destinations and activities

// Call printItinerary() and manually verify the printed output

travelPackage.printItinerary();

}

}

Explanation of test cases for `TravelPackage` class:

1. `testAddPassenger`: Checks if a passenger is successfully added to the travel package.

2. `testAddDestination`: Verifies if a destination is successfully added to the travel package's itinerary.

3. `testPrintItinerary`: This test prints the itinerary for manual inspection to ensure correct formatting and content. Manual verification is required.