LAPORAN PRAKTIKUM PBO

Test Driven Development Junit 5

Logo, company name

Description automatically generated

Disusun oleh :

Muhamad Rafli Nur Ikhsan

201511048

D-3 Teknik Informatika 2B

Jurusan Teknik Komputer dan Informatika

Program studi D3 Teknik Informatika

Politeknik Negeri Bandung

1. Listing 20.1 Passanger Class

* Source code

public class Passenger {  
 private String name;  
 private boolean vip;  
  
 public Passenger(String name, boolean vip) {  
 this.name = name;  
 this.vip = vip;  
 }  
 public String getName() {  
 return name;  
 }  
 public boolean isVip() {  
 return vip;  
 }  
}

* Fungsi / tujuan class

Class ini berfungsi untuk menginisiasi String name dan Boolean vip.

1. Listing 20.2 Flight Class

* Source code

public class Flight {  
 private String id;  
 private List<Passenger> passengers = new ArrayList<Passenger>();  
 private String flightType;  
  
 public Flight(String id, String flightType) {  
 this.id = id;  
 this.flightType = flightType;  
 }  
 public String getId() {  
 return id;  
 }  
 public List<Passenger> getPassengersList() {  
 return Collections.*unmodifiableList*(passengers);  
 }  
 public String getFlightType() {  
 return flightType;  
 }  
 public boolean addPassenger(Passenger passenger) {  
 switch (flightType) {  
 case "Economy":  
 return passengers.add(passenger);  
  
 case "Business":  
 if (passenger.isVip()) {  
 return passengers.add(passenger);  
 }  
 return false;  
 default:  
 throw new RuntimeException("Unknown type: " + flightType);  
 }  
 }  
 public boolean removePassenger(Passenger passenger) {  
 switch (flightType) {  
 case "Economy":  
 if (!passenger.isVip()) {  
 return passengers.remove(passenger);  
 }  
 return false;  
 case "Business":  
 return false;  
 default:  
 throw new RuntimeException("Unknown type: " + flightType);  
 }  
 }  
}

* Fungsi / tujuan Class

Class ini berfungsi untuk menginisiasi String id, List passanger dan String flightType

1. Listing 20.3 Airport Class, including the Main Method

* Source code

public class Airport {  
 public static void main(String[] args) {  
  
 Flight economyFlight = new Flight("1", "Economy");  
 Flight businessFlight = new Flight("2", "Business");  
  
 Passenger james = new Passenger("James", true);  
 Passenger mike = new Passenger("Mike", false);  
  
 businessFlight.addPassenger(james);  
 businessFlight.removePassenger(james);  
 businessFlight.addPassenger(mike);  
 economyFlight.addPassenger(mike);  
 System.*out*.println("Business flight passengers list:");  
 for (Passenger passenger: businessFlight.getPassengersList()) {  
 System.*out*.println(passenger.getName());  
 }  
 System.*out*.println("Economy flight passengers list:");  
 for (Passenger passenger: economyFlight.getPassengersList()) {  
 System.*out*.println(passenger.getName());  
 }  
 }  
}

* Fungsi / tujuan Class

Ini adalah Main Class untuk mengeksekusi program dari Class-Class yang sebelumnya sudah dibuat.

* SS akhir program

Text

Description automatically generated

1. Listing 20.4 Junit 5 Dependencies added to the pom.xml

* Source code

<dependencies>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-api</artifactId>  
 <version>5.6.0</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>5.6.0</version>  
 <scope>test</scope>  
 </dependency>  
</dependencies>

* Fungsi / tujuan Class

Class ini berfungsi sebagai library dari junit yang nantinya akan dipakai pada saat melakukan unit testing.

1. Listing 20.5 Testing the business logic for an economic flight

* Source Code

@DisplayName("Given there is an economy flight")  
@Nested  
class EconomyFlightTest {  
 private Flight economyFlight;  
 @BeforeEach  
 void setUp() {  
 economyFlight = new Flight("1", "Economy");  
 }  
 @Test  
 public void testEconomyFlightRegularPassenger() {  
 Passenger mike = new Passenger("Mike", false);  
 *assertEquals*("1", economyFlight.getId());  
 *assertEquals*(true, economyFlight.addPassenger(mike));  
 *assertEquals*(1, economyFlight.getPassengersList().size());  
 *assertEquals*("Mike",  
 economyFlight.getPassengersList().get(0).getName());  
 *assertEquals*(true, economyFlight.removePassenger(mike));  
 *assertEquals*(0, economyFlight.getPassengersList().size());  
 }  
 @Test  
 public void testEconomyFlightVipPassenger() {  
 Passenger james = new Passenger("James", true);  
 *assertEquals*("1", economyFlight.getId());  
 *assertEquals*(true, economyFlight.addPassenger(james));  
 *assertEquals*(1, economyFlight.getPassengersList().size());  
 *assertEquals*("James",  
 economyFlight.getPassengersList().get(0).getName());  
 *assertEquals*(false, economyFlight.removePassenger(james));  
 *assertEquals*(1, economyFlight.getPassengersList().size());  
 }  
}

* Fungsi / tujuan Class

Class ini adalah tempat dilakukannya testing dari Economy Flight

* SS akhir program

Text

Description automatically generated

1. Listing 20.6 Testing the business logic for an business flight

* Source Code
* @DisplayName("Given there is a business flight")  
  @Nested  
  class BusinessFlightTest {  
   private Flight businessFlight;  
   @BeforeEach  
   void setUp() {  
   businessFlight = new Flight("2", "Business");  
   }  
   @Test  
   public void testBusinessFlightRegularPassenger() {  
   Passenger mike = new Passenger("Mike", false);  
   *assertEquals*(false, businessFlight.addPassenger(mike));  
   *assertEquals*(0, businessFlight.getPassengersList().size());  
   *assertEquals*(false, businessFlight.removePassenger(mike));  
   *assertEquals*(0, businessFlight.getPassengersList().size());  
   }  
   @Test  
   public void testBusinessFlightVipPassenger() {  
   Passenger james = new Passenger("James", true);  
   *assertEquals*(true, businessFlight.addPassenger(james));  
   *assertEquals*(1, businessFlight.getPassengersList().size());  
   *assertEquals*(false, businessFlight.removePassenger(james));  
   *assertEquals*(1, businessFlight.getPassengersList().size());  
   }  
  }
* Fungsi / tujuan Class

Class ini adalah tempat dilakukannya testing dari Business Flight

* SS akhir program

Text

Description automatically generated

1. Listing 20.7 Abstract Flight class, the basis of the hierarchy

* public abstract class Flight {  
    
   private String id;  
   List<Passenger> passengers = new ArrayList<Passenger>();  
   public Flight(String id) {  
   this.id = id;  
   }  
   public String getId() {  
   return id;  
   }  
   public List<Passenger> getPassengersList() {  
   return Collections.*unmodifiableList*(passengers);  
   }  
   public abstract boolean addPassenger(Passenger passenger);  
   public abstract boolean removePassenger(Passenger passenger);  
  }
* Fungsi / tujuan Class

Class ini berfungsi untuk menginisiasi String id dan List Passenger. Dan juga pendeklarasian abstract method Boolean addPassenger dan Boolean removePassenger.

1. Listing 20.8 EconomyFlight class, extending the abstract Flight class

* Source Code

public class EconomyFlight extends Flight{  
  
 public EconomyFlight(String id) {  
 super(id);  
 }  
  
 @Override  
 public boolean addPassenger(Passenger passenger) {  
 return passengers.add(passenger);  
 }  
 @Override  
 public boolean removePassenger(Passenger passenger) {  
 if (!passenger.isVip()) {  
 return passengers.remove(passenger);  
 }  
 return false;  
 }  
}

* Fungsi / tujuan Class

Class ini berisikan inisiasi untuk Flight yang berjenis Economy.

1. Listing 20.9 BusinessFlight class, extending the abstract Flight class

* Source Code

public BusinessFlight(String id) {  
 super(id);  
 }  
 @Override  
 public boolean addPassenger(Passenger passenger) {  
 if (passenger.isVip()) {  
 return passengers.add(passenger);  
 }  
 return false;  
 }  
 @Override  
 public boolean removePassenger(Passenger passenger) {  
 return false;  
 }  
}

* Fungsi / tujuan Class

Class ini berisikan inisiasi untuk Flight yang berjenis Business.

1. Listing 20.10 Refactoring propagation into the AirportTest class

* Source Code Test Economy

@DisplayName("Given there is an economy flight")  
@Nested  
class EconomyFlightTest {  
 private EconomyFlight economyFlight;  
 @BeforeEach  
 void setUp() {  
 economyFlight = new EconomyFlight("1");  
 }  
 @Test  
 public void testEconomyFlightRegularPassenger() {  
 Passenger mike = new Passenger("Mike", false);  
 *assertEquals*("1", economyFlight.getId());  
 *assertEquals*(true, economyFlight.addPassenger(mike));  
 *assertEquals*(1, economyFlight.getPassengersList().size());  
 *assertEquals*("Mike",  
 economyFlight.getPassengersList().get(0).getName());  
 *assertEquals*(true, economyFlight.removePassenger(mike));  
 *assertEquals*(0, economyFlight.getPassengersList().size());  
 }  
 @Test  
 public void testEconomyFlightVipPassenger() {  
 Passenger james = new Passenger("James", true);  
 *assertEquals*("1", economyFlight.getId());  
 *assertEquals*(true, economyFlight.addPassenger(james));  
 *assertEquals*(1, economyFlight.getPassengersList().size());  
 *assertEquals*("James",  
 economyFlight.getPassengersList().get(0).getName());  
 *assertEquals*(false, economyFlight.removePassenger(james));  
 *assertEquals*(1, economyFlight.getPassengersList().size());  
 }  
}

* Fungsi / tujuan Class

Class ini adalah tempat dilakukannya testing dari Economy Flight dan Business Flight. Namun codingan di atas hanya melakukan testing pada Economy Flight.

* SS akhir program

Text

Description automatically generated

* Source Code Business Flight
* @DisplayName("Given there is a business flight")  
  @Nested  
  class BusinessFlightTest {  
   private Flight businessFlight;  
    
   @BeforeEach  
   void setUp() {  
   businessFlight = new BusinessFlight("2");  
   }  
    
   public void testBusinessFlightRegularPassenger() {  
   Passenger mike = new Passenger("Mike", false);  
   *assertEquals*(false, businessFlight.addPassenger(mike));  
   *assertEquals*(0, businessFlight.getPassengersList().size());  
   *assertEquals*(false, businessFlight.removePassenger(mike));  
   *assertEquals*(0, businessFlight.getPassengersList().size());  
   }  
    
   @Test  
   public void testBusinessFlightVipPassenger() {  
   Passenger james = new Passenger("James", true);  
   *assertEquals*(true, businessFlight.addPassenger(james));  
   *assertEquals*(1, businessFlight.getPassengersList().size());  
   *assertEquals*(false, businessFlight.removePassenger(james));  
   *assertEquals*(1, businessFlight.getPassengersList().size());  
   }  
  }
* Fungsi / tujuan Class

Class ini adalah tempat dilakukannya testing dari Economy Flight dan Business Flight. Namun codingan di atas hanya melakukan testing pada Business Flight.

* SS akhir program

Text

Description automatically generated