LAPORAN PRAKTIKUM
TEKNIK PEMROGRAMAN

JUNIT TEST DRIVEN DEVELOPMENT

MINGGU KE-9

NAMA: FAUZI ISMAIL

NIM: 241524042

KELAS: D4-1B

PROGRAM STUDI SARJANA TERAPAN

TEKNIK INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2025

DAFTAR ISI

https://github.com/mailvlous/teknikPemrograman/tree/main/Week9/flight-management-app

JUNIT TEST DRIVEN DEVELOPMENT FLIGHT MANAGEMENT APPLICATION

Pada program Flight Management App terdapat Test Driven Development dalam file AirportTest.java.

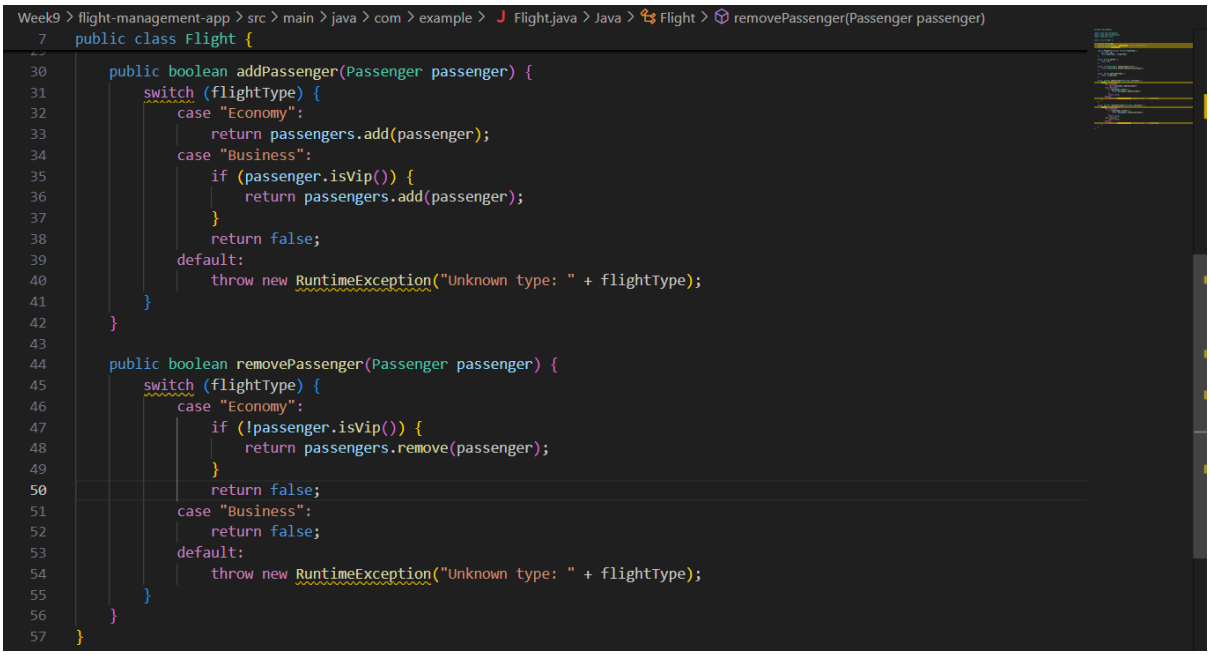
Terdapat 3 Class utama dalam Flight Management Application ini :

1. Passenger

```java
package com.example;

/**
 * Hello world!
 *
 */
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;
    }

}
```

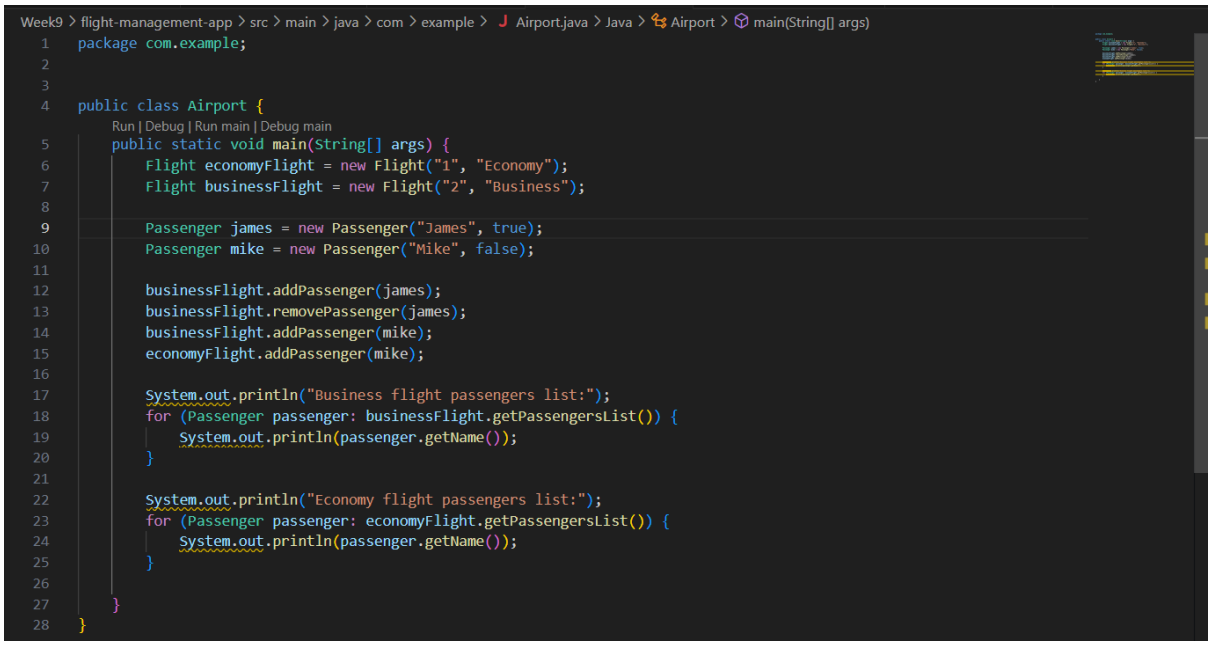
2. Flight

```
Week9 > flight-management-app > src > main > java > com > example > Flight.java > Java > Flight > removePassenger(Passenger passenger)
1  package com.example;
2
3  import java.util.ArrayList;
4  import java.util.Collections;
5  import java.util.List;
6
7  public class Flight {
8
9      private String id;
10     private List<Passenger> passengers = new ArrayList<>();
11     private String flightType;
12
13     public Flight(String id, String flightType) {
14         this.id = id;
15         this.flightType = flightType;
16     }
17
18     public String getId() {
19         return id;
20     }
21
22     public List<Passenger> getPassengersList() {
23         return Collections.unmodifiableList(passengers);
24     }
25
26     public String getFlightType() {
27         return flightType;
28     }
```



```
Week9 > flight-management-app > src > main > java > com > example > Flight.java > Java > Flight > removePassenger(Passenger passenger)
7  public class Flight {
30     public boolean addPassenger(Passenger passenger) {
31         switch (flightType) {
32             case "Economy":
33                 return passengers.add(passenger);
34             case "Business":
35                 if (passenger.isVip()) {
36                     return passengers.add(passenger);
37                 }
38                 return false;
39             default:
40                 throw new RuntimeException("Unknown type: " + flightType);
41         }
42     }
43
44     public boolean removePassenger(Passenger passenger) {
45         switch (flightType) {
46             case "Economy":
47                 if (!passenger.isVip()) {
48                     return passengers.remove(passenger);
49                 }
50                 return false;
51             case "Business":
52                 return false;
53             default:
54                 throw new RuntimeException("Unknown type: " + flightType);
55         }
56     }
57 }
```

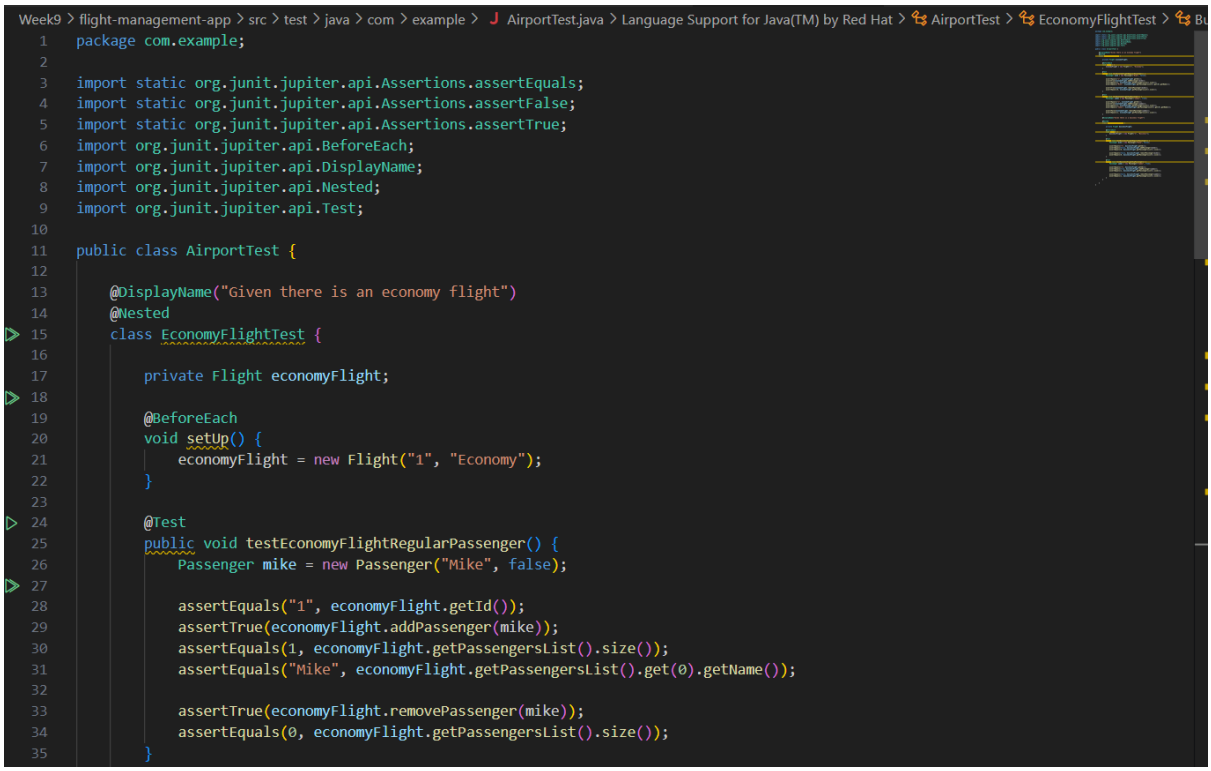
3. Airport/Main



```java
package com.example;


public class Airport {
    Run | Debug | Run main | Debug main
    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());
        }

    }
}
```

Class class tadi akan di tes di dalam

AirportTest.java



```java
package com.example;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertFalse;
import static org.junit.jupiter.api.Assertions.assertTrue;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Nested;
import org.junit.jupiter.api.Test;

public class AirportTest {

    @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());
            assertTrue(economyFlight.addPassenger(mike));
            assertEquals(1, economyFlight.getPassengersList().size());
            assertEquals("Mike", economyFlight.getPassengersList().get(0).getName());

            assertTrue(economyFlight.removePassenger(mike));
            assertEquals(0, economyFlight.getPassengersList().size());
        }
```

```
Week9 > flight-management-app > src > test > java > com > example > AirportTest.java > Language Support for Java(TM) by Red Hat > AirportTest > EconomyFlightTest > Bu
11  public class AirportTest {
15      class EconomyFlightTest {
36
37          @Test
38          public void testEconomyFlightVipPassenger() {
39              Passenger james = new Passenger("James", true);
40
41              assertEquals("1", economyFlight.getId());
42              assertTrue(economyFlight.addPassenger(james));
43              assertEquals(1, economyFlight.getPassengersList().size());
44              assertEquals("James", economyFlight.getPassengersList().get(0).getName());
45
46              assertFalse(economyFlight.removePassenger(james));
47              assertEquals(1, economyFlight.getPassengersList().size());
48          }
49
50          @DisplayName("Given there is a business flight")
51
52          @Nested
53          class BusinessFlightTest {
54
55              private Flight businessFlight;
56
57              @BeforeEach
58              void setUp() {
59                  businessFlight = new Flight("2", "Business");
60              }
```

```
Week9 > flight-management-app > src > test > java > com > example > AirportTest.java > Language Support for Java(TM) by Red Hat > AirportTest > EconomyFlightTest > Bu
11  public class AirportTest {
15      class EconomyFlightTest {
53          class BusinessFlightTest {
62              @Test
63              public void testBusinessFlightRegularPassenger() {
64                  Passenger mike = new Passenger("Mike", false);
65
66                  assertEquals("2", businessFlight.getId());
67                  assertEquals(false, businessFlight.addPassenger(mike));
68                  assertEquals(0, businessFlight.getPassengersList().size());
69
70                  assertEquals(false, businessFlight.removePassenger(mike));
71                  assertEquals(0, businessFlight.getPassengersList().size());
72              }
73
74              @Test
75              public void testBusinessFlightVipPassenger() {
76                  Passenger james = new Passenger("James", true);
77
78                  assertEquals("2", businessFlight.getId());
79                  assertEquals(true, businessFlight.addPassenger(james));
80                  assertEquals(1, businessFlight.getPassengersList().size());
81
82                  assertEquals(false, businessFlight.removePassenger(james));
83                  assertEquals(1, businessFlight.getPassengersList().size());
84              }
85          }
86
87      }
88  }
```

Dalam file AirportTest.java akan mengecek 4 Test yaitu:

1.

```
24          @Test
25          public void testEconomyFlightRegularPassenger() {
26              Passenger mike = new Passenger("Mike", false);
27
28              assertEquals("1", economyFlight.getId());
29              assertTrue(economyFlight.addPassenger(mike));
30              assertEquals(1, economyFlight.getPassengersList().size());
31              assertEquals("Mike", economyFlight.getPassengersList().get(0).getName());
32
33              assertTrue(economyFlight.removePassenger(mike));
34              assertEquals(0, economyFlight.getPassengersList().size());
35          }
```

Dalam test ini akan mengecek passenger dalam penerbangan regular di kelas economy, dimana tes ini expected:

- Mengecek bahwa object mike dibuat dengan nama “Mike”, dan isVip() false
- Memastikan bahwa ID `economyFlight` adalah `"1"`.
- Mengecek bahwa menambahkan Mike ke penerbangan ekonomi berhasil (bernilai true).
- Memastikan bahwa sekarang jumlah penumpang di flight menjadi **1** setelah Mike ditambahkan.
- Memastikan bahwa penumpang pertama (index 0) di daftar penumpang adalah **"Mike"**.
- Mengecek bahwa **menghapus Mike dari flight berhasil** (bernilai `true`).
- Setelah Mike dihapus, daftar penumpang harus kosong lagi (jumlah = 0).

2.

```
@Test
public void testEconomyFlightVipPassenger() {
    Passenger james = new Passenger("James", true);

    assertEquals("1", economyFlight.getId());
    assertTrue(economyFlight.addPassenger(james));
    assertEquals(1, economyFlight.getPassengersList().size());
    assertEquals("James", economyFlight.getPassengersList().get(0).getName());

    assertFalse(economyFlight.removePassenger(james));
    assertEquals(1, economyFlight.getPassengersList().size());
}
```

Dalam test ini akan mengecek passenger dalam penerbangan vip di kelas economy, dimana tes ini expected:

- Mengecek bahwa object james dibuat dengan nama "James", dan isVip() true.
- Memastikan bahwa ID economyFlight adalah "1".
- Mengecek bahwa menambahkan james ke penerbangan ekonomi berhasil (bernilai true).
- Memastikan bahwa sekarang jumlah penumpang di flight menjadi 1 setelah james ditambahkan.
- Memastikan bahwa penumpang pertama (index 0) di daftar penumpang adalah "James".
- Mengecek bahwa menghapus james dari flight gagal (bernilai false).
- Setelah mencoba menghapus james, daftar penumpang tetap berisi satu orang (jumlah = 1).

3.

```
@Test
public void testBusinessFlightRegularPassenger() {
    Passenger mike = new Passenger("Mike", false);

    assertEquals("2", businessFlight.getId());
    assertEquals(false, businessFlight.addPassenger(mike));
    assertEquals(0, businessFlight.getPassengersList().size());

    assertEquals(false, businessFlight.removePassenger(mike));
    assertEquals(0, businessFlight.getPassengersList().size());
}
```

Dalam test ini akan mengecek passenger dalam penerbangan bisnis namun bukan vip, dimana tes ini:

- Mengecek bahwa object `mike` dibuat dengan nama "Mike" dan `isVip()` bernilai `false`.
- Memastikan bahwa ID `businessFlight` adalah `"2"`.
- Mengecek bahwa menambahkan `mike` ke `businessFlight` gagal (bernilai `false`).
- Memastikan bahwa setelah mencoba menambahkan `mike`, jumlah penumpang di flight tetap `0`.
- Mengecek bahwa menghapus `mike` dari `businessFlight` juga gagal (bernilai `false`).
- Setelah mencoba menghapus `mike`, jumlah penumpang di flight tetap `0`.

4.

```
74      @Test
75      public void testBusinessFlightVipPassenger() {
76          Passenger james = new Passenger("James", true);
77
78          assertEquals("2", businessFlight.getId());
79          assertEquals(true, businessFlight.addPassenger(james));
80          assertEquals(1, businessFlight.getPassengersList().size());
81
82          assertEquals(false, businessFlight.removePassenger(james));
83          assertEquals(1, businessFlight.getPassengersList().size());
84      }
85  }
```

Dalam test ini akan mengecek vip passenger dalam kelas business, dimana tes ini:

- Mengecek bahwa object `james` dibuat dengan nama "James" dan `isVip()` bernilai `true`.
- Memastikan bahwa ID `businessFlight` adalah `"2"`.
- Mengecek bahwa menambahkan `james` ke `businessFlight` berhasil (bernilai `true`).
- Memastikan bahwa setelah `james` ditambahkan, jumlah penumpang di flight menjadi `1`.
- Mengecek bahwa menghapus `james` dari `businessFlight` gagal (bernilai `false`).
- Setelah mencoba menghapus `james`, daftar penumpang tetap berisi satu orang (jumlah = `1`).