LAPORAN PRAKTIKUM

TEKNIK PEMROGRAMAN

CLEAN CODE SONARQUBE TEST

MINGGU KE-10

NAMA: FAUZI ISMAIL

NIM: 241524042

KELAS: D4-1B

PROGRAM STUDI SARJANA TERAPAN

TEKNIK INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2025

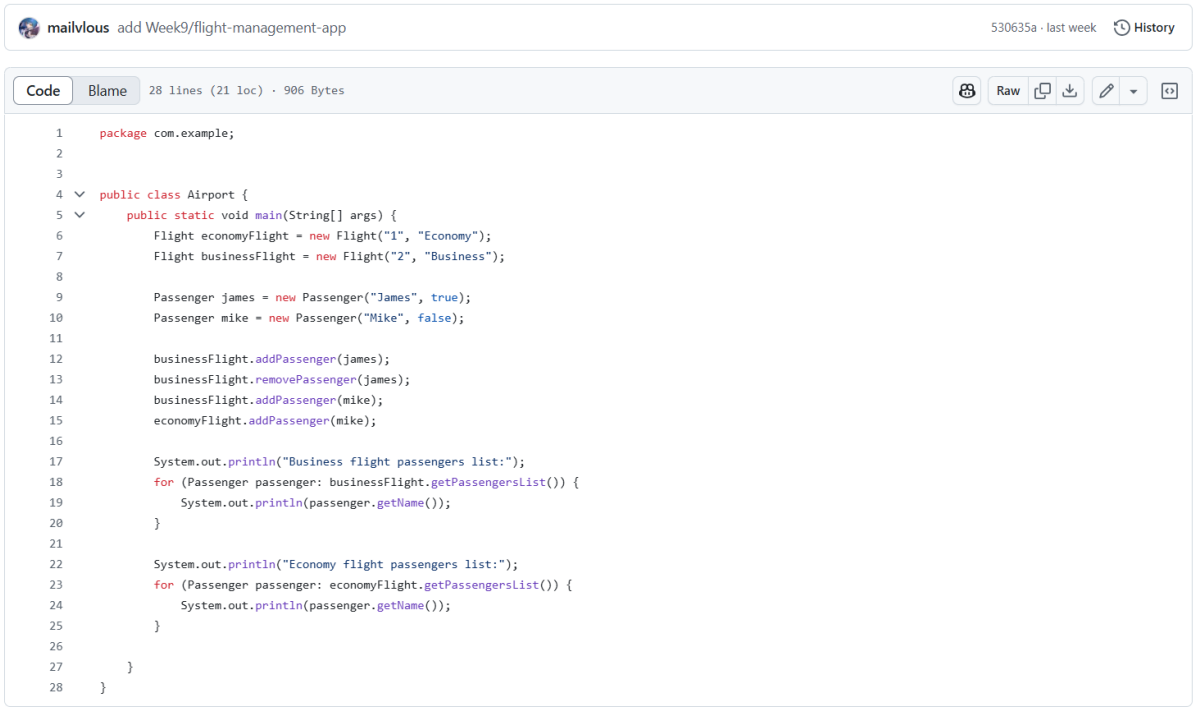
DAFTAR ISI

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

FLIGHT MANAGEMENT APP CLEAN CODE TEST ANALYSIS USING SONARQUBE

1. SEBELUM DI REFACTOR

Airport.java



mailvlous add Week9/flight-management-app 530635a · last week History

Code Blame 28 lines (21 loc) · 906 Bytes Raw

```
1   package com.example;
2
3
4   public class Airport {
5       public static void main(String[] args) {
6           Flight economyFlight = new Flight("1", "Economy");
7           Flight businessFlight = new Flight("2", "Business");
8
9           Passenger james = new Passenger("James", true);
10          Passenger mike = new Passenger("Mike", false);
11
12          businessFlight.addPassenger(james);
13          businessFlight.removePassenger(james);
14          businessFlight.addPassenger(mike);
15          economyFlight.addPassenger(mike);
16
17          System.out.println("Business flight passengers list:");
18          for (Passenger passenger: businessFlight.getPassengersList()) {
19              System.out.println(passenger.getName());
20          }
21
22          System.out.println("Economy flight passengers list:");
23          for (Passenger passenger: economyFlight.getPassengersList()) {
24              System.out.println(passenger.getName());
25          }
26
27      }
28  }
```

Flight.java

mailvlous add Week9/flight-management-app 530635a · last week History

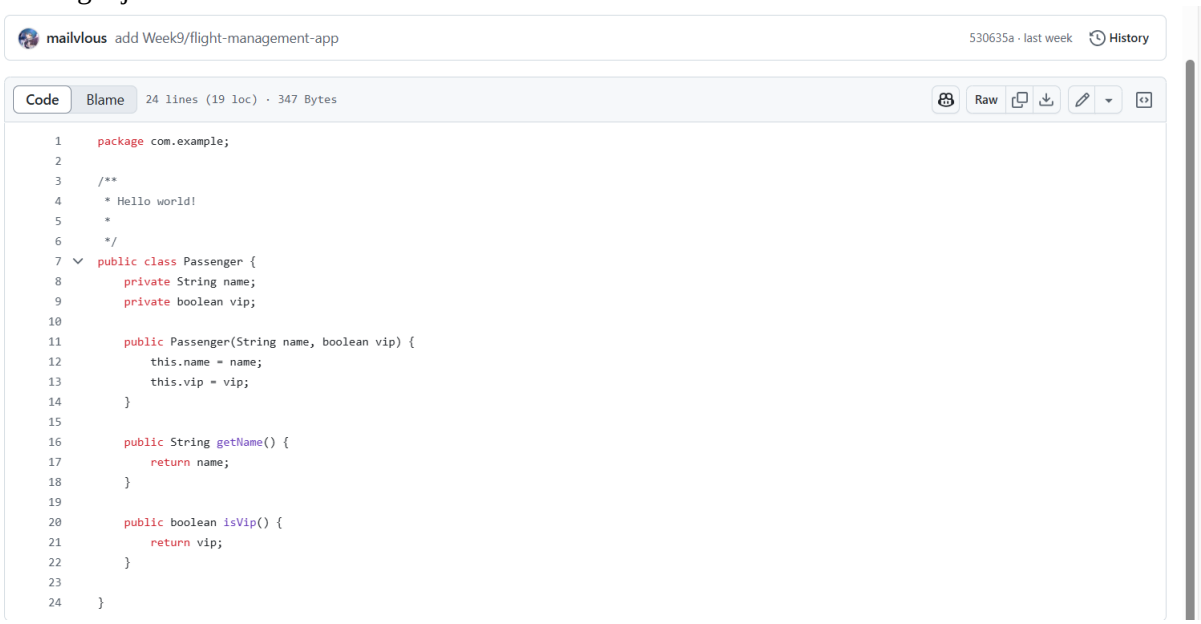
Code Blame 57 lines (48 loc) · 1.45 KB Raw

```java
package com.example;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Flight {

    private String id;
    private List<Passenger> passengers = new ArrayList<>();
    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);
    }
```

Code Blame 57 lines (48 loc) · 1.45 KB Raw

```java
public class Flight {
    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);
        }
    }
```

Passenger.java

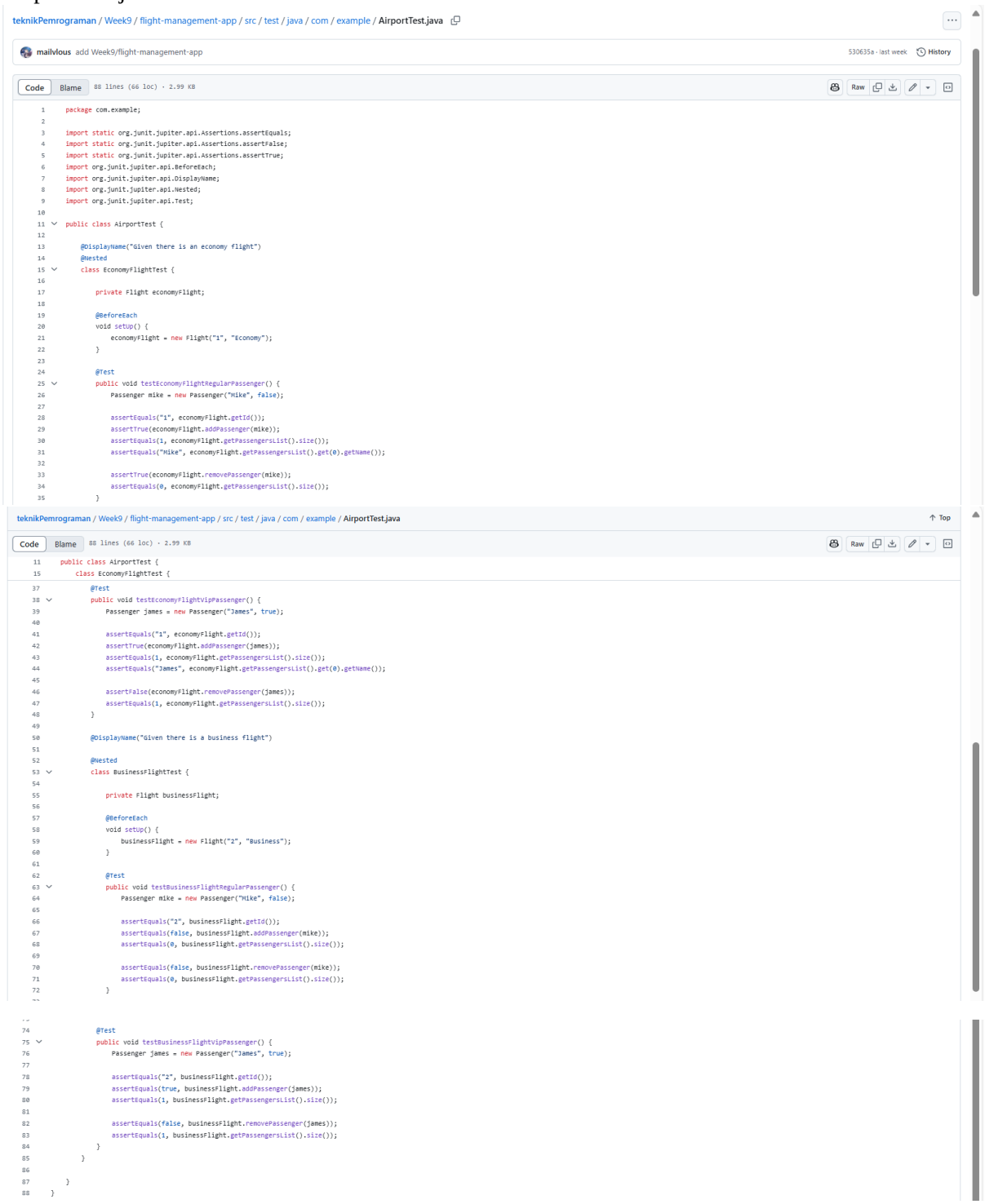


mailvlous add Week9/flight-management-app 530635a · last week History

Code Blame 24 lines (19 loc) · 347 Bytes Raw

```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;
    }

}
```

AirportTest.java



teknikPemrograman / Week9 / flight-management-app / src / test / java / com / example / AirportTest.java

mailvlous add Week9/flight-management-app — 530635a · last week — History

Code | Blame — 88 lines (66 loc) · 2.99 KB — Raw

```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());
        }
```

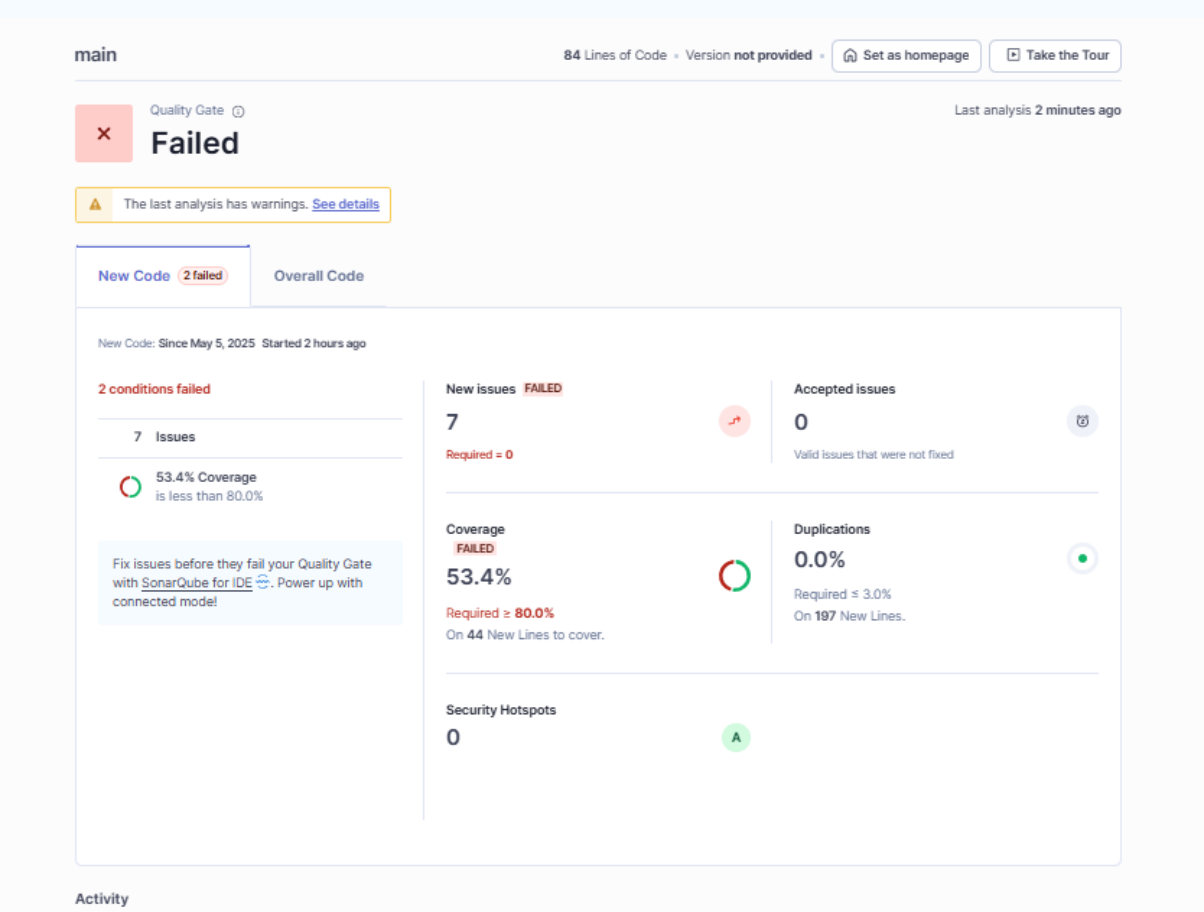
teknikPemrograman / Week9 / flight-management-app / src / test / java / com / example / AirportTest.java — Top

Code | Blame — 88 lines (66 loc) · 2.99 KB — Raw

```java
public class AirportTest {
    class EconomyFlightTest {
        @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());
        }

        @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("2", businessFlight.getId());
                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("2", businessFlight.getId());
                assertEquals(true, businessFlight.addPassenger(james));
                assertEquals(1, businessFlight.getPassengersList().size());

                assertEquals(false, businessFlight.removePassenger(james));
                assertEquals(1, businessFlight.getPassengersList().size());
            }
        }

    }
}
```

2. REFACTORING

SEBELUM DI REFACTOR DI SONARQUBE ANALYSIS



Yang saya lakukan adalah:

1. Mengganti `System.out.println()` dengan Logger

Menggunakan:

```
private static final Logger logger = Logger.getLogger(Airport.class.getName());
```

Sehingga:

```
logger.info("Pesan log...");
```

2. Menghindari RuntimeException generic

Menggunakan:

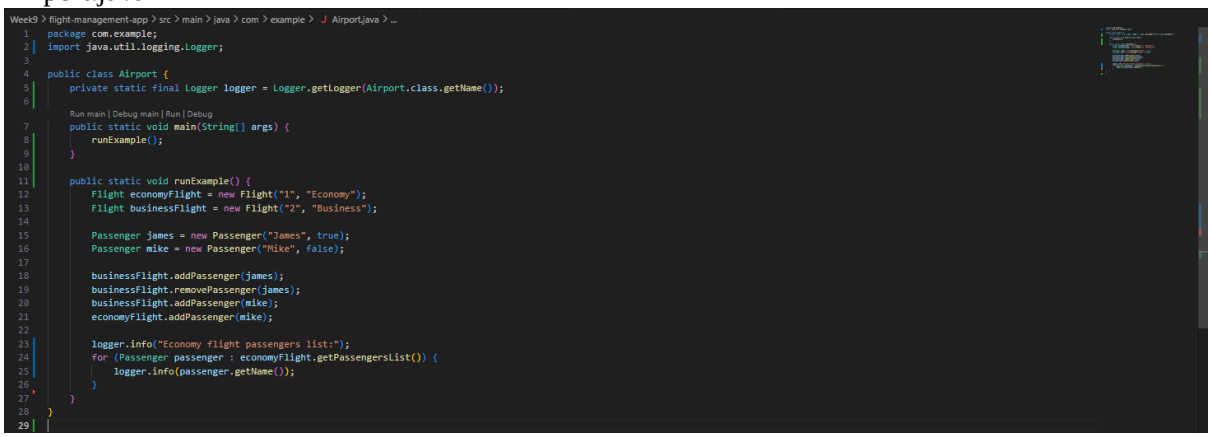
```
private static class UnknownFlightTypeException extends RuntimeException {
    public UnknownFlightTypeException(String flightType) {
        super("Unknown flight type: " + flightType);
    }
}
```

3. Memisahkan kode bisnis dan kode demo

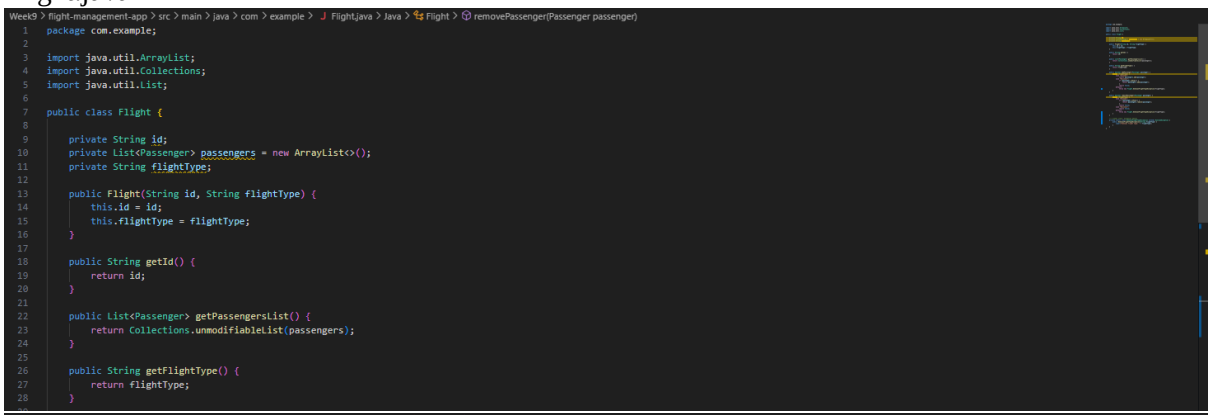
```
public static void main(String[] args) {
    runExample();
}
```

3. SETELAH REFACTOR

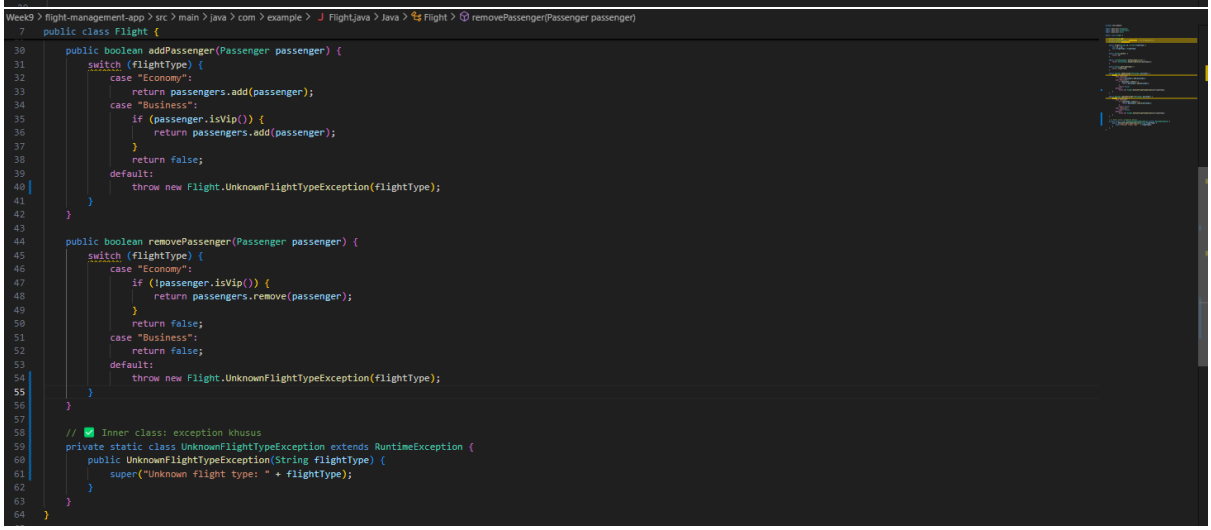
Airport.java



Flight.java



```java
package com.example;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class Flight {

    private String id;
    private List<Passenger> passengers = new ArrayList<>();
    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;
    }
```

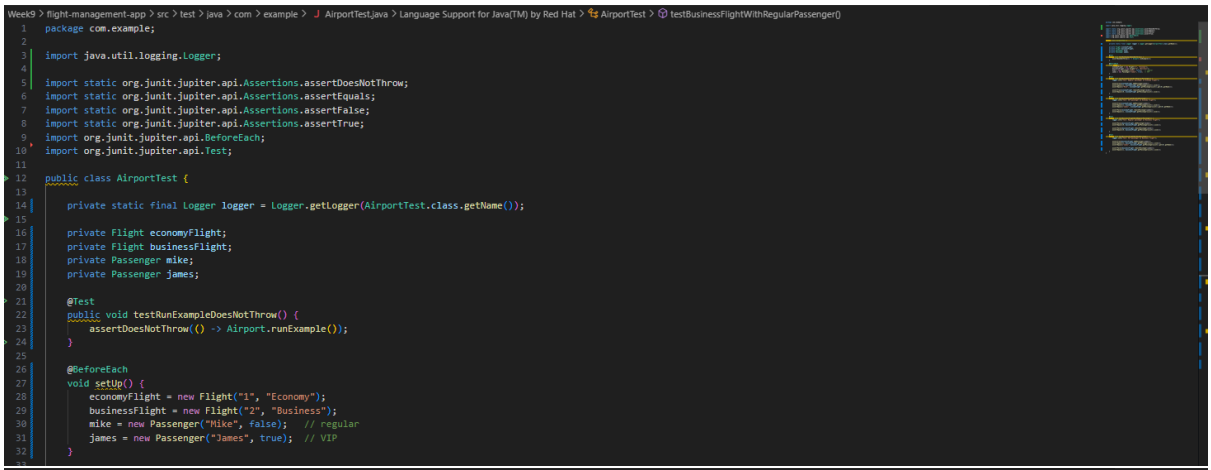


```java
public class Flight {

    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 Flight.UnknownFlightTypeException(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 Flight.UnknownFlightTypeException(flightType);
        }
    }

    // ✅ Inner class: exception khusus
    private static class UnknownFlightTypeException extends RuntimeException {
        public UnknownFlightTypeException(String flightType) {
            super("Unknown flight type: " + flightType);
        }
    }
}
```

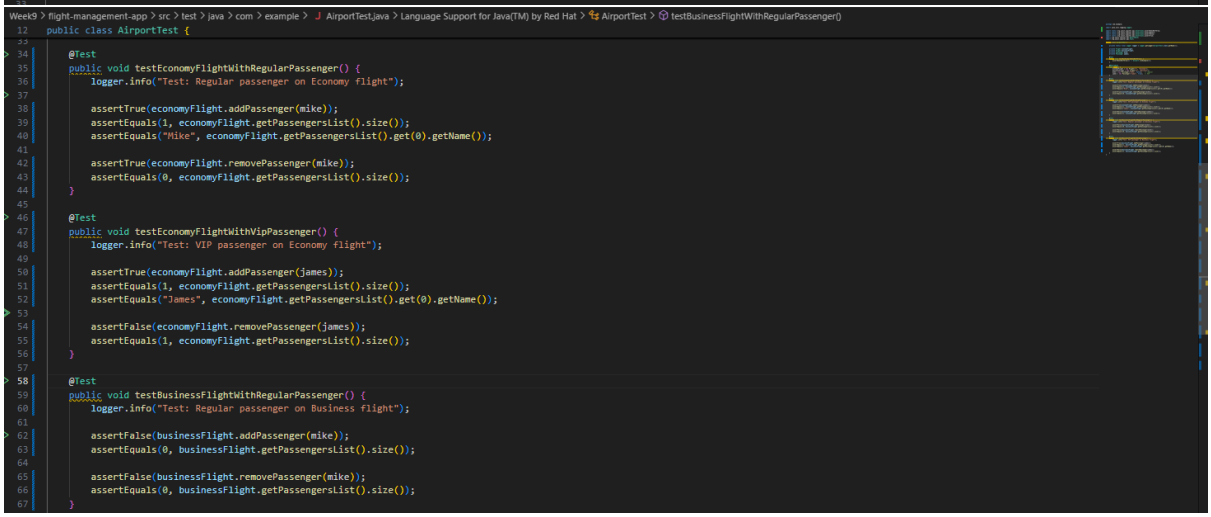
Passenger.java

```java
package com.example;

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;
    }
}
```

AirportTest.java



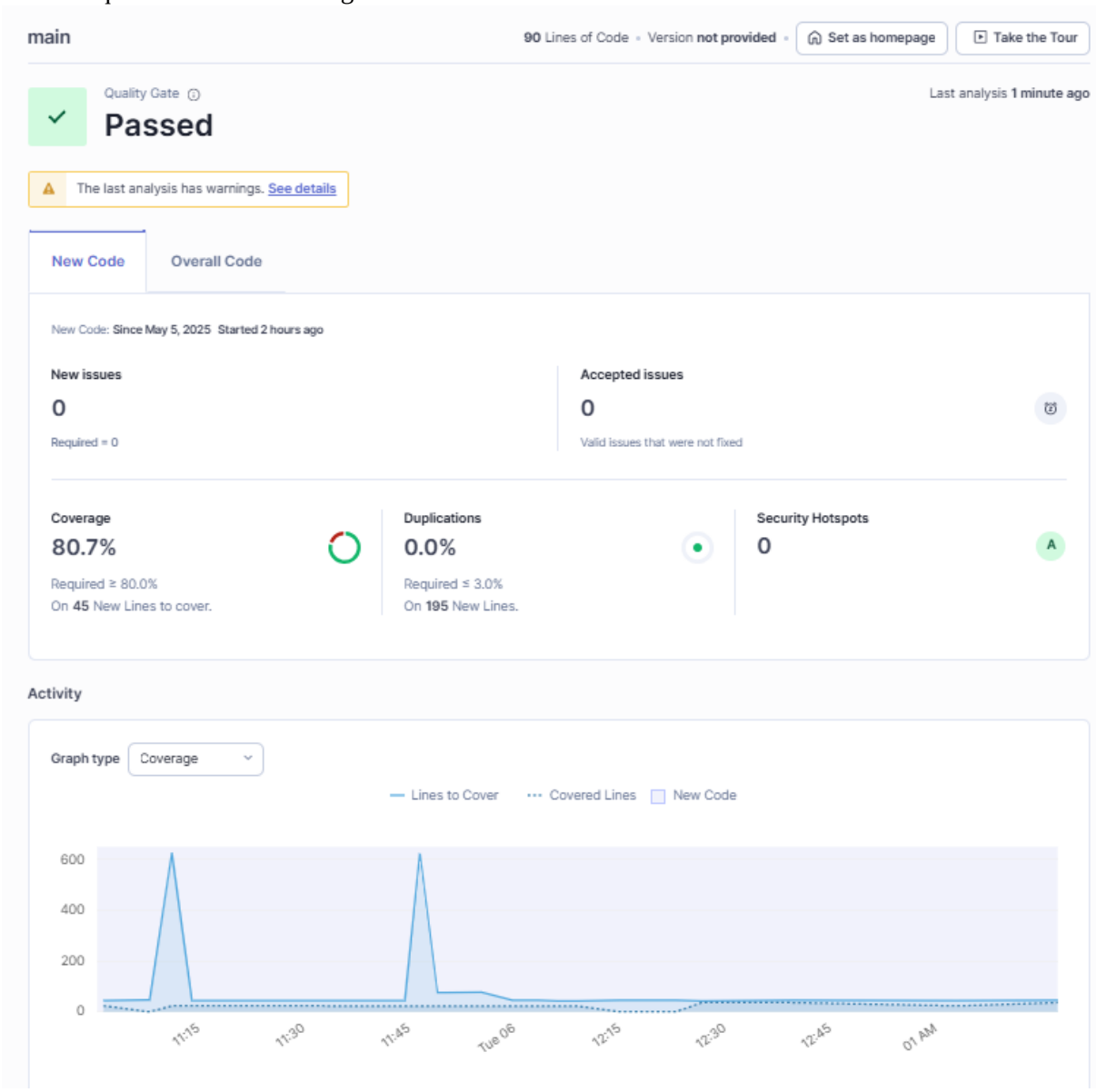
```java
package com.example;

import java.util.logging.Logger;

import static org.junit.jupiter.api.Assertions.assertDoesNotThrow;
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.Test;

public class AirportTest {

    private static final Logger logger = Logger.getLogger(AirportTest.class.getName());

    private Flight economyFlight;
    private Flight businessFlight;
    private Passenger mike;
    private Passenger james;

    @Test
    public void testRunExampleDoesNotThrow() {
        assertDoesNotThrow(() -> Airport.runExample());
    }

    @BeforeEach
    void setUp() {
        economyFlight = new Flight("1", "Economy");
        businessFlight = new Flight("2", "Business");
        mike = new Passenger("Mike", false);   // regular
        james = new Passenger("James", true);  // VIP
    }
```



```java
public class AirportTest {

    @Test
    public void testEconomyFlightWithRegularPassenger() {
        logger.info("Test: Regular passenger on Economy flight");

        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());
    }

    @Test
    public void testEconomyFlightWithVipPassenger() {
        logger.info("Test: VIP passenger on Economy flight");

        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());
    }

    @Test
    public void testBusinessFlightWithRegularPassenger() {
        logger.info("Test: Regular passenger on Business flight");

        assertFalse(businessFlight.addPassenger(mike));
        assertEquals(0, businessFlight.getPassengersList().size());

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

```java
public class AirportTest {

    @Test
    public void testBusinessFlightWithVipPassenger() {
        logger.info("Test: VIP passenger on Business flight");

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

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

Di Sonarqube setelah refactoring:



main

90 Lines of Code • Version **not provided** • Set as homepage | Take the Tour

Quality Gate

Passed

Last analysis **2 minutes ago**

The last analysis has warnings. See details

New Code | Overall Code

Security
0 Open issues — A

Reliability
0 Open issues — A

Maintainability
0 Open issues — A

Accepted issues
0
Valid issues that were not fixed

Coverage
80.7%
On **45** lines to cover.

Duplications
0.0%
On **114** lines.

Security Hotspots
0 — A

Activity

Graph type: Coverage