

LAPORAN PRAKTIKUM
TEKNIK PEMROGRAMAN

CLEAN CODE SONARQUBE TEST

MINGGU KE-10



POLBAN

NAMA: FAUZI ISMAIL

NIM: 241524042

KELAS: D4-1B

PROGRAM STUDI SARJANA TERAPAN

TEKNIK INFORMATIKA

POLITEKNIK NEGERI BANDUNG

2025

DAFTAR ISI

DAFTAR ISI.....	2
https://github.com/mailvlous/teknikPemrograman/tree/main/Week9/flight-management-app	3
FLIGHT MANAGEMENT APP CLEAN CODE TEST ANALYSIS USING SONARQUBE	3
1. SEBELUM DI REFACTOR.....	3
2. REFACTORING.....	7
3. SETELAH REFACTOR	8

<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

```
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

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

Code Blame 57 lines (48 loc) · 1.45 KB

Raw Download Edit View

```
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}
```

Code Blame 57 lines (48 loc) · 1.45 KB

Raw Download Edit View

```
7 public class Flight {
26    public String getFlightType() {
27        return flightType;
28    }
29
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    }
}
```

Passenger.java

mailvrous add Week9/flight-management-app

530635a · last week History

Code Blame 24 lines (19 loc) · 347 Bytes

Raw Copy Download Edit View

```
1 package com.example;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class Passenger {
8     private String name;
9     private boolean vip;
10
11     public Passenger(String name, boolean vip) {
12         this.name = name;
13         this.vip = vip;
14     }
15
16     public String getName() {
17         return name;
18     }
19
20     public boolean isVip() {
21         return vip;
22     }
23
24 }
```

AirportTest.java

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

mailvoux add Week9/flight-management-app

530635a · last week History

Code Blame 88 lines (66 loc) · 2.99 KB

Raw Copy Download Edit View

```
1 package com.example;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.junit.jupiter.api.Assertions.assertFalse;
5 import static org.junit.jupiter.api.Assertions.assertTrue;
6 import org.junit.jupiter.api.BeforeEach;
7 import org.junit.jupiter.api.DisplayName;
8 import org.junit.jupiter.api.Nested;
9 import org.junit.jupiter.api.Test;
10
11 public class AirportTest {
12
13     @DisplayName("Given there is an economy flight")
14     @Nested
15     class EconomyFlightTest {
16
17         private Flight economyFlight;
18
19         @BeforeEach
20         void setUp() {
21             economyFlight = new Flight("1", "Economy");
22         }
23
24         @Test
25         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         }
36     }
37 }
```

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

↑ Top

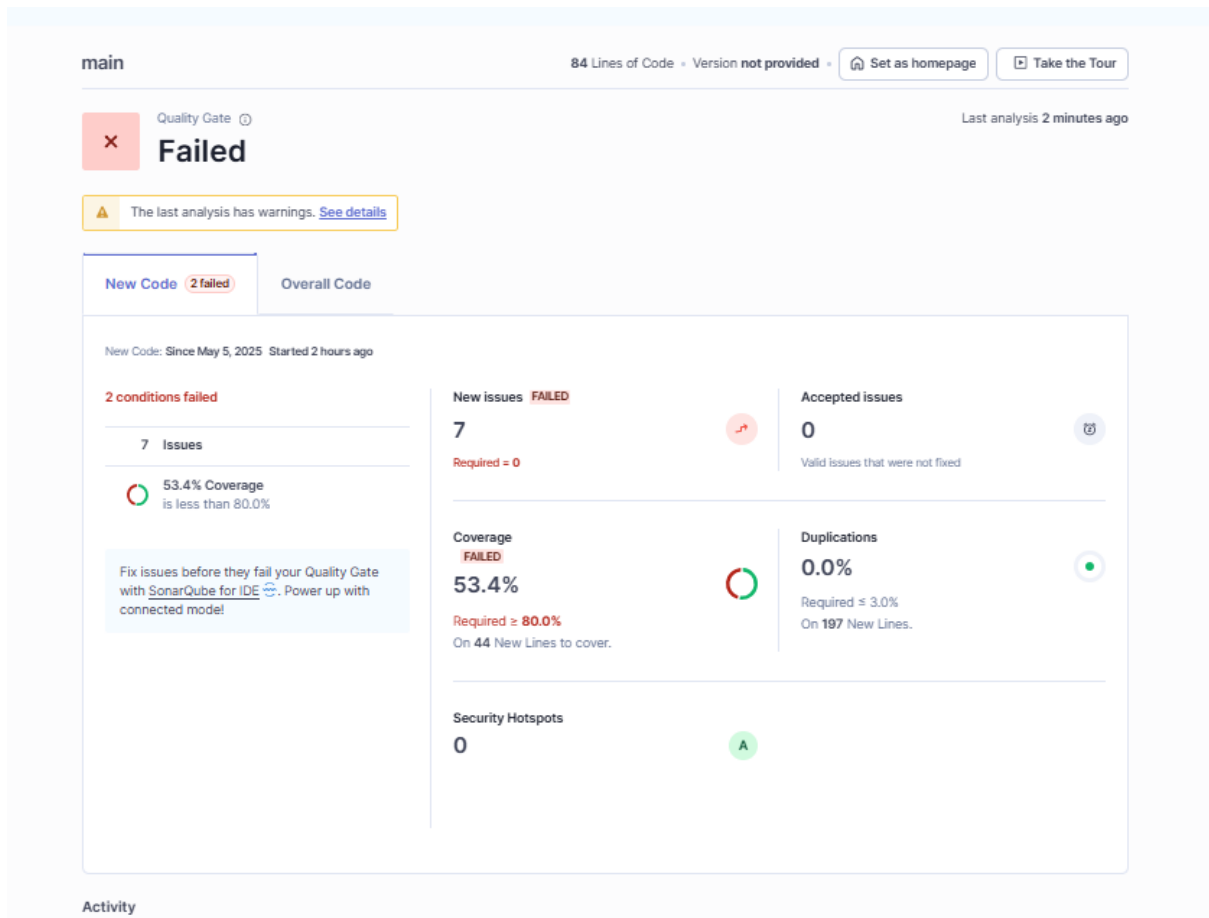
Code Blame 88 lines (66 loc) · 2.99 KB

Raw Copy Download Edit View

```
11 public class AirportTest {
15     class EconomyFlightTest {
16
17         @Test
18         void testEconomyFlightVipPassenger() {
19             Passenger james = new Passenger("James", true);
20
21             assertEquals("1", economyFlight.getId());
22             assertTrue(economyFlight.addPassenger(james));
23             assertEquals(1, economyFlight.getPassengersList().size());
24             assertEquals("James", economyFlight.getPassengersList().get(0).getName());
25
26             assertFalse(economyFlight.removePassenger(james));
27             assertEquals(1, economyFlight.getPassengersList().size());
28         }
29
30         @DisplayName("Given there is a business flight")
31         @Nested
32         class BusinessFlightTest {
33
34             private Flight businessFlight;
35
36             @BeforeEach
37             void setUp() {
38                 businessFlight = new Flight("2", "Business");
39             }
40
41             @Test
42             void testBusinessFlightRegularPassenger() {
43                 Passenger mike = new Passenger("Mike", false);
44
45                 assertEquals("2", businessFlight.getId());
46                 assertEquals(false, businessFlight.addPassenger(mike));
47                 assertEquals(0, businessFlight.getPassengersList().size());
48
49                 assertEquals(false, businessFlight.removePassenger(mike));
50                 assertEquals(0, businessFlight.getPassengersList().size());
51             }
52
53             @Test
54             void testBusinessFlightVipPassenger() {
55                 Passenger james = new Passenger("James", true);
56
57                 assertEquals("2", businessFlight.getId());
58                 assertEquals(true, businessFlight.addPassenger(james));
59                 assertEquals(1, businessFlight.getPassengersList().size());
60
61                 assertEquals(false, businessFlight.removePassenger(james));
62                 assertEquals(1, businessFlight.getPassengersList().size());
63             }
64         }
65     }
66 }
```

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

```
Week9 > flight-management-app > src > main > java > com > example > J Airport.java > ...  
1 package com.example;  
2 import java.util.logging.Logger;  
3  
4 public class Airport {  
5     private static final Logger logger = Logger.getLogger(Airport.class.getName());  
6  
7     public static void main(String[] args) {  
8         runExample();  
9     }  
10  
11     public static void runExample() {  
12         Flight economyFlight = new Flight("1", "Economy");  
13         Flight businessFlight = new Flight("2", "Business");  
14  
15         Passenger james = new Passenger("James", true);  
16         Passenger mike = new Passenger("Mike", false);  
17  
18         businessFlight.addPassenger(james);  
19         businessFlight.removePassenger(james);  
20         businessFlight.addPassenger(mike);  
21         economyFlight.addPassenger(mike);  
22  
23         logger.info("Economy flight passengers list:");  
24         for (Passenger passenger : economyFlight.getPassengersList()) {  
25             logger.info(passenger.getName());  
26         }  
27     }  
28  
29 }
```


Flight.java

```
Week9 > flight-management-app > src > main > java > com > example > J 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    }
29 }
```

```
Week9 > flight-management-app > src > main > java > com > example > J Flight.java > Java > Flight > removePassenger( Passenger passenger)
7 public class Flight {
8
9     public boolean addPassenger(Passenger passenger) {
10         switch (flightType) {
11             case "Economy":
12                 return passengers.add(passenger);
13             case "Business":
14                 if (passenger.isVip()) {
15                     return passengers.add(passenger);
16                 }
17                 return false;
18             default:
19                 throw new Flight.UnknownFlightTypeException(flightType);
20         }
21     }
22
23     public boolean removePassenger(Passenger passenger) {
24         switch (flightType) {
25             case "Economy":
26                 if (!passenger.isVip()) {
27                     return passengers.remove(passenger);
28                 }
29                 return false;
30             case "Business":
31                 return false;
32             default:
33                 throw new Flight.UnknownFlightTypeException(flightType);
34         }
35     }
36
37     // Inner class: exception khusus
38     private static class UnknownFlightTypeException extends RuntimeException {
39         public UnknownFlightTypeException(String flightType) {
40             super("Unknown flight type: " + flightType);
41         }
42     }
43 }
44 }
```

Passenger.java

```
Week9 > flight-management-app > src > main > java > com > example > J Passenger.java > ...
1 package com.example;
2
3 public class Passenger {
4     private String name;
5     private boolean vip;
6
7     public Passenger(String name, boolean vip) {
8         this.name = name;
9         this.vip = vip;
10    }
11
12    public String getName() {
13        return name;
14    }
15
16    public boolean isVip() {
17        return vip;
18    }
19 }
20 }
```

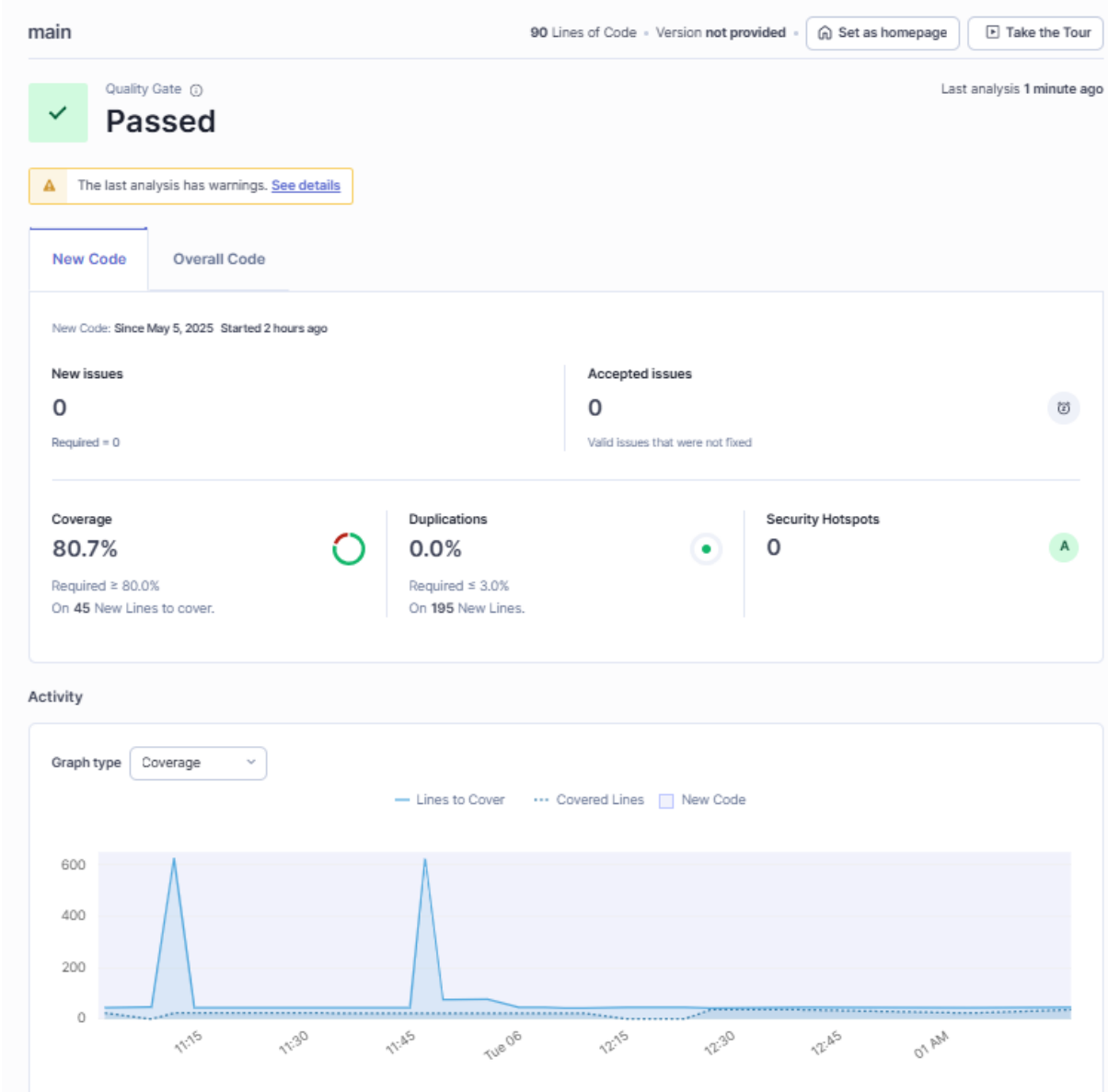
AirportTest.java

```
Week9 > flight-management-app > src > test > java > com > example > J AirportTest.java > Language Support for Java(TM) by Red Hat > AirportTest > testBusinessFlightWithRegularPassenger()
1 package com.example;
2
3 import java.util.logging.Logger;
4
5 import static org.junit.jupiter.api.Assertions.assertDoesNotThrow;
6 import static org.junit.jupiter.api.Assertions.assertEquals;
7 import static org.junit.jupiter.api.Assertions.assertFalse;
8 import static org.junit.jupiter.api.Assertions.assertTrue;
9 import org.junit.jupiter.api.BeforeEach;
10 import org.junit.jupiter.api.Test;
11
12 public class AirportTest {
13
14     private static final Logger logger = Logger.getLogger(AirportTest.class.getName());
15
16     private Flight economyFlight;
17     private Flight businessFlight;
18     private Passenger mike;
19     private Passenger james;
20
21     @Test
22     public void testRunExampleDoesNotThrow() {
23         assertDoesNotThrow(() -> Airport.runExample());
24     }
25
26     @BeforeEach
27     void setUp() {
28         economyFlight = new Flight("1", "Economy");
29         businessFlight = new Flight("2", "Business");
30         mike = new Passenger("Mike", false); // regular
31         james = new Passenger("James", true); // VIP
32     }
33 }
```

```
Week9 > flight-management-app > src > test > java > com > example > J AirportTest.java > Language Support for Java(TM) by Red Hat > AirportTest > testBusinessFlightWithRegularPassenger()
33 public class AirportTest {
34
35     @Test
36     public void testEconomyFlightWithRegularPassenger() {
37         logger.info("Test: Regular passenger on Economy flight");
38
39         assertTrue(economyFlight.addPassenger(mike));
40         assertEquals(1, economyFlight.getPassengersList().size());
41         assertEquals("Mike", economyFlight.getPassengersList().get(0).getName());
42
43         assertTrue(economyFlight.removePassenger(mike));
44         assertEquals(0, economyFlight.getPassengersList().size());
45     }
46
47     @Test
48     public void testEconomyFlightWithVipPassenger() {
49         logger.info("Test: VIP passenger on Economy flight");
50
51         assertTrue(economyFlight.addPassenger(james));
52         assertEquals(1, economyFlight.getPassengersList().size());
53         assertEquals("James", economyFlight.getPassengersList().get(0).getName());
54
55         assertFalse(economyFlight.removePassenger(james));
56         assertEquals(1, economyFlight.getPassengersList().size());
57     }
58
59     @Test
60     public void testBusinessFlightWithRegularPassenger() {
61         logger.info("Test: Regular passenger on Business flight");
62
63         assertFalse(businessFlight.addPassenger(mike));
64         assertEquals(0, businessFlight.getPassengersList().size());
65
66         assertFalse(businessFlight.removePassenger(mike));
67         assertEquals(0, businessFlight.getPassengersList().size());
68     }
69 }
```

```
Week9 > flight-management-app > src > test > java > com > example > J AirportTest.java > Language Support for Java(TM) by Red Hat > AirportTest > testBusinessFlightWithRegularPassenger()
68 public class AirportTest {
69
70     @Test
71     public void testBusinessFlightWithVipPassenger() {
72         logger.info("Test: VIP passenger on Business flight");
73
74         assertTrue(businessFlight.addPassenger(james));
75         assertEquals(1, businessFlight.getPassengersList().size());
76         assertEquals("James", businessFlight.getPassengersList().get(0).getName());
77
78         assertFalse(businessFlight.removePassenger(james));
79         assertEquals(1, businessFlight.getPassengersList().size());
80     }
81 }
```

Di Sonarqube setelah refactoring:



Quality Gate ⓘ

Passed

Last analysis 2 minutes ago

The last analysis has warnings. [See details](#)

New Code

Overall Code

Security 0 Open issues	Reliability 0 Open issues	Maintainability 0 Open issues
Accepted issues 0 <small>Valid issues that were not fixed</small>	Coverage 80.7% <small>On 45 lines to cover.</small>	Duplications 0.0% <small>On 114 lines.</small>
Security Hotspots 0		

Activity

