

LAPORAN PRAKTIKUM
PEMOGRAMAN BERBASIS OBJECT
“Pertemuan ke-11”



Disusun oleh :
Alifah Fisalsabilawati
201511035
2B – D3 Teknik Informatika

Jurusan Teknik Komputer dan Informatika
Program Studi D3 Teknik Informatika
Politeknik Negeri Bandung

20.1 – 20.3 Running class Passenger, Flight, Airport.java

Hasil output

```
Problems @ Javadoc Declaration Console X
<terminated> Airport [Java Application] C:\Program Files\Java
Business flight passengers list:
James
Economy flight passengers list:
Mike
```

Passenger.java

```
2+ * To change this license header, choose License Headers in Pro
6
7- /**
8  *
9  * @author ALIFAH
10 */
11 public class Passenger {
12     private String name;
13     private boolean vip;
14
15-     public Passenger(String name, boolean vip) {
16         this.name = name;
17         this.vip = vip;
18     }
19
20-     public String getName() {
21         return name;
22     }
23
24-     public boolean isVip() {
25         return vip;
26     }
27 }
```

Flight.java

```
20 * To change this license header, choose License Headers in Project Properties
11 import java.util.ArrayList;
14 public class Flight {
15     private String id;
16     private List<Passenger> passengers = new ArrayList<Passenger>();
17     private String flightType;
18
19     public Flight(String id, String flightType) {
20         this.id = id;
21         this.flightType = flightType;
22     }
23
24     public String getId() {
25         return id;
26     }
27     public List<Passenger> getPassengersList() {
28         return Collections.unmodifiableList(passengers);
29     }
30     public String getFlightType() {
31         return flightType;
32     }
33     public boolean addPassenger(Passenger passenger) {
34         switch (flightType) {
35             case "Economy":
36                 return passengers.add(passenger);
37             case "Business":
38                 if (passenger.isVip()) {
39                     return passengers.add(passenger);
40                 }
41                 return false;
42             default:
43                 throw new RuntimeException("Unknown type: " + flightType);
44         }
45     }
46     public boolean removePassenger(Passenger passenger) {
47         switch (flightType) {
48             case "Economy":
49                 if (!passenger.isVip()) {
50                     return passengers.remove(passenger);
51                 }
52                 return false;
53             case "Business":
54                 return false;
55             default:
56                 throw new RuntimeException("Unknown type: " + flightType);
57         }
58     }
59 }
```

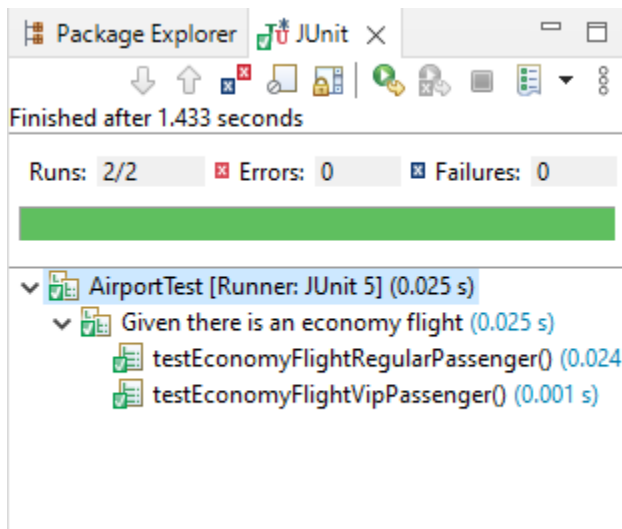
Airport.java

```
2+ * To change this license header, choose License Headers in Project Properties.
6
7- /**
8  *
9  * @author ALIFAH
10 */
11 public class Airport {
12-     public static void main(String[] args) {
13         Flight economyFlight = new Flight("1", "Economy");
14         Flight businessFlight = new Flight("2", "Business");
15         Passenger james = new Passenger("James", true);
16         Passenger mike = new Passenger("Mike", false);
17         businessFlight.addPassenger(james);
18         businessFlight.removePassenger(james);
19         businessFlight.addPassenger(mike);
20         economyFlight.addPassenger(mike);
21
22         System.out.println("Business flight passengers list:");
23         for (Passenger passenger: businessFlight.getPassengersList()) {
24             System.out.println(passenger.getName());
25         }
26
27         System.out.println("Economy flight passengers list:");
28         for (Passenger passenger: economyFlight.getPassengersList()) {
29             System.out.println(passenger.getName());
30         }
31     }
32 }
```

20.4 Junit 5 dependencies add to pom.xml

```
9  <dependencies>
10    <dependency>
11      <groupId>org.junit.jupiter</groupId>
12      <artifactId>junit-jupiter-api</artifactId>
13      <version>5.6.0</version>
14      <scope>test</scope>
15      <type>jar</type>
16    </dependency>
17    <dependency>
18      <groupId>org.seleniumhq.selenium</groupId>
19      <artifactId>selenium-java</artifactId>
20      <scope>test</scope>
21      <version>2.44.0</version>
22    </dependency>
23    <dependency>
24      <groupId>com.opera</groupId>
25      <artifactId>operadriver</artifactId>
26      <scope>test</scope>
27      <version>1.5</version>
28      <exclusions>
29        <exclusion>
30          <groupId>org.seleniumhq.selenium</groupId>
31          <artifactId>selenium-remote-driver</artifactId>
32        </exclusion>
33      </exclusions>
34    </dependency>
35    <dependency>
36      <groupId>junit</groupId>
37      <artifactId>junit</artifactId>
38      <scope>test</scope>
39      <version>4.11</version>
40    </dependency>
41  </dependencies>
```

20.5 testing the business logic for an economic flight



The screenshot shows an IDE window titled "JUnit" with a "Package Explorer" on the left. The main area displays the test results for "AirportTest [Runner: JUnit 5] (0.025 s)". The test suite "AirportTest" is expanded, showing a single test "Given there is an economy flight (0.025 s)". This test contains two sub-tests: "testEconomyFlightRegularPassenger() (0.024 s)" and "testEconomyFlightVipPassenger() (0.001 s)". Both sub-tests are marked as passed with green checkmarks. A green progress bar at the top of the results area indicates that all tests passed. The status bar at the bottom shows "Runs: 2/2", "Errors: 0", and "Failures: 0".

Package Explorer JUnit x

Finished after 1.433 seconds

Runs: 2/2 Errors: 0 Failures: 0

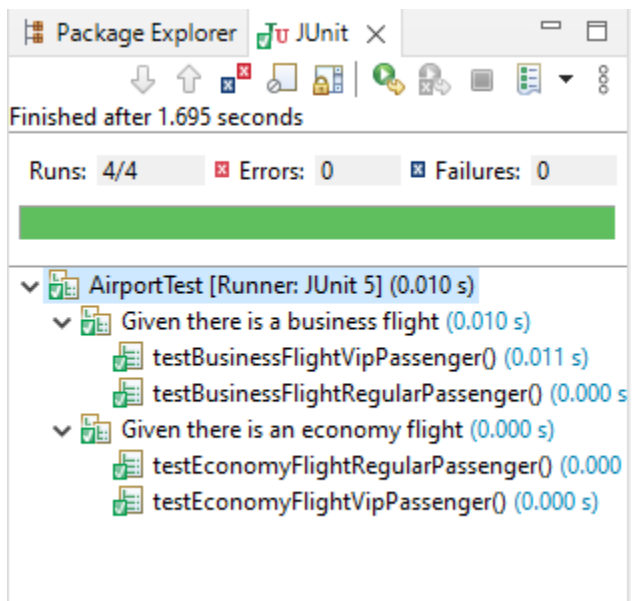
▼ AirportTest [Runner: JUnit 5] (0.025 s)

▼ Given there is an economy flight (0.025 s)

testEconomyFlightRegularPassenger() (0.024 s)

testEconomyFlightVipPassenger() (0.001 s)

20.6 testing the business logic for an business flight



20.7 Refactoring class flight

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.List;
4
5 public abstract class Flight {
6     private String id;
7     List<Passenger> passengers = new ArrayList<Passenger>();
8     public Flight(String id) {
9         this.id = id;
10    }
11
12    public String getId() {
13        return id;
14    }
15
16    public List<Passenger> getPassengersList() {
17        return Collections.unmodifiableList(passengers);
18    }
19
20    public abstract boolean addPassenger(Passenger passenger);
21    public abstract boolean removePassenger(Passenger passenger);
22 }
```

20.8 – 20.9 membuat economyFlight dan BusinessFlight

economyFlight.java

```
1 public class economyFlight extends Flight {
2     public economyFlight(String id) {
3         super(id);
4     }
5
6     @Override
7     public boolean addPassenger(Passenger passenger) {
8         return passengers.add(passenger);
9     }
10    @Override
11    public boolean removePassenger(Passenger passenger) {
12        if (!passenger.isVip()) {
13            return passengers.remove(passenger);
14        }
15        return false;
16    }
17 }
```

BusinessFlight.java

```
1 public class BusinessFlight extends Flight {
2     public BusinessFlight(String id) {
3         super(id);
4     }
5
6     @Override
7     public boolean addPassenger(Passenger passenger) {
8         if (passenger.isVip()) {
9             return passengers.add(passenger);
10        }
11        return false;
12    }
13
14    @Override
15    public boolean removePassenger(Passenger passenger) {
16        return false;
17    }
18 }
```

20.10 Refactoring class AirportTest

```
20 public class AirportTest {
21
22     @DisplayName("Given there is an economy flight")
23
24     @Nested
25     class EconomyFlightTest {
26         private Flight economyFlight;
27
28         @BeforeEach
29         void setUp() {
30             economyFlight = new economyFlight("1");
31         }
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61     @DisplayName("Given there is a business flight")
62
63     @Nested
64     class BusinessFlightTest {
65         private Flight businessFlight;
66
67         @BeforeEach
68         void setUp() {
69             businessFlight = new BusinessFlight("2");
70         }
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```