

Laporan Praktikum

“Praktikum 9 Test-Driven Development JUnit”

Mata Kuliah: Teknik Pemrograman



Disusun oleh:

Nama : Arkan Ramadhan Nugraha

NIM : 241524033

Kelas : 1B-TI4

POLITEKNIK NEGERI BANDUNG

2025

DAFTAR ISI

Laporan Praktikum.....	
DAFTAR ISI.....	i
DAFTAR GAMBAR.....	ii
DAFTAR LAMPIRAN.....	iii
BAB I PENDAHULUAN.....	1
I.1. Latar Belakang.....	1
I.2. Tujuan.....	1
BAB II PEMBAHASAN.....	2
II.1. Kasus 1.....	2
II.1.1. Screenshot Hasil Akhir Program.....	2
II.1.2. Penjelasan.....	6
BAB III HASIL DAN ANALISIS.....	9
III.1. Hasil.....	9
III.2. Lesson Learned.....	9
BAB IV KESIMPULAN DAN SARAN.....	10
IV.1. Kesimpulan.....	10
LAMPIRAN.....	11

DAFTAR GAMBAR

Gambar II.1: Class Passenger.....	2
Gambar II.2: Class Flight.....	3
Gambar II.3: Class Airport (Main).....	3
Gambar II.4: Class BusinessFlight.....	4
Gambar II.5: Class EconomyFlight.....	4
Gambar II.6: Class EconomyFlightTest di Class AirportTest.....	5
Gambar II.7: Class BusinessFlightTest di Class AirportTest.....	5
Gambar II.8: Output “mvn test” dengan hasil Success.....	6

DAFTAR LAMPIRAN

Lampiran 1 - Source Code .zip.....	11
Lampiran 2 - Link Repository GitHub.....	11

BAB I PENDAHULUAN

I.1. Latar Belakang

Test-driven development adalah pemrograman praktis yang menggunakan siklus pengembangan pendek yang berulang di mana persyaratan diubah menjadi kasus uji, dan kemudian program dimodifikasi untuk membuat tes

I.2. Tujuan

Mempelajari penerapan Test-Driven Development (TDD) dalam pengembangan perangkat lunak menggunakan JUnit di Java.

BAB II PEMBAHASAN

II.1. Kasus 1

II.1.1. Screenshot Hasil Akhir Program

```
my-app/src/main/java/com/mycompany/app/Passenger.java
1  package com.mycompany.app;
2  public class Passenger {
3
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 this.name;
14     }
15
16     public boolean isVip() {
17         return this.vip;
18     }
19 }
20
```

Gambar II.1: Class Passenger

```

my-app/src/main/java/com/mycompany/app/Flight.java public abstract class Flight
26 package com.mycompany.app;
25
24 import java.util.ArrayList;
23 import java.util.Collections;
22 import java.util.List;
21
20 public abstract class Flight {
19
18     private String id;
17     protected List<Passenger> passengers = new ArrayList<>();
16
15     public Flight(String id) {
14         this.id = id;
13     }
12
11     public String getId() {
10         return id;
9     }
8
7     public List<Passenger> getPassengersList() {
6         return Collections.unmodifiableList(passengers);
5     }
4
3     public abstract boolean addPassenger(Passenger passenger);
2
1     public abstract boolean removePassenger(Passenger passenger);
27 // return flightType; & Arney1, 1 hour ago
1 // }

```

Gambar II.2: Class Flight

```

my-app/src/main/java/com/mycompany/app/Airport.java public class Airport public static main( )
8 package com.mycompany.app;
7
6 import com.mycompany.app.flights.BusinessFlight;
5 import com.mycompany.app.flights.EconomyFlight;
4
3 public class Airport {
2
1     public static void main(String[] args) {
9         Flight economyFlight = new EconomyFlight("1");
1         Flight businessFlight = new BusinessFlight("2");
2
3         Passenger james = new Passenger("James", true);
4         Passenger mike = new Passenger("Mike", false);
5
6         businessFlight.addPassenger(james);
7         businessFlight.removePassenger(james);
8
9         businessFlight.addPassenger(mike);
10        economyFlight.addPassenger(mike);
11
12        System.out.println("Business flight passengers list: ");
13        for (Passenger passenger : businessFlight.getPassengersList()) {
14            System.out.println(passenger.getName());
15        }
16        System.out.println("Economy flight passengers list: ");
17        for (Passenger passenger : economyFlight.getPassengersList()) {
18            System.out.println(passenger.getName());
19        }
20    }
21 }

```

Gambar II.3: Class Airport (Main)

```

my-app/src/main/java/com/mycompany/app/flights/BusinessFlight.java  Q % I
1 package com.mycompany.app.flights;
2
3 import com.mycompany.app.Flight;
4 import com.mycompany.app.Passenger;
5
6 public class BusinessFlight extends Flight {
7     public BusinessFlight(String id) {
8         super(id);
9     }
10
11     @Override
12     public boolean addPassenger(Passenger passenger) {
13         if (passenger.isVip()) {
14             return passengers.add(passenger);
15         }
16         return false;
17     }
18
19     @Override
20     public boolean removePassenger(Passenger passenger) {
21         return false;
22     }
23 }

```

Gambar II.4: Class *BusinessFlight*

```

my-app/src/main/java/com/mycompany/app/flights/EconomyFlight.java  Q % I
1 package com.mycompany.app.flights;
2
3 import com.mycompany.app.Flight;
4 import com.mycompany.app.Passenger;
5
6 public class EconomyFlight extends Flight {
7     public EconomyFlight(String id) {
8         super(id);
9     }
10
11     @Override
12     public boolean addPassenger(Passenger passenger) {
13         return passengers.add(passenger);
14     }
15
16     @Override
17     public boolean removePassenger(Passenger passenger) {
18         if (!passenger.isVip()) {
19             return passengers.remove(passenger);
20         }
21         return false;
22     }
23 }

```

Gambar II.5: Class *EconomyFlight*


```

/app/src/test/java/com/mycompany/app/AirportTest.java public class AirportTest { class EconomyFlightTest
15 import static org.junit.jupiter.api.Assertions.assertTrue;
16
17 import com.mycompany.app.flights.BusinessFlight;
18 import com.mycompany.app.flights.EconomyFlight;
19 import org.junit.jupiter.api.BeforeEach;
20 // import org.junit.jupiter.api.BeforeEach;
21 import org.junit.jupiter.api.DisplayName;
22 import org.junit.jupiter.api.Nested;
23 import org.junit.jupiter.api.Test;
24
25 /**
26  * Unit test for simple App.
27  */
28
29 public class AirportTest {
30
31     @DisplayName("Given there is an economy flight") // Arneyi, 1 hour ago
32     @Nested
33     class EconomyFlightTest {
34
35         private Flight economyFlight;
36
37         @BeforeEach
38         void setUp() {
39             economyFlight = new EconomyFlight("1");
40         }
41
42         @Test
43         public void testEconomyFlightRegularPassengers() {
44             Passenger mike = new Passenger("Mike", false);
45
46             assertEquals("1", economyFlight.getId());
47             assertEquals(true, economyFlight.addPassenger(mike));
48             assertEquals(1, economyFlight.getPassengersList().size());
49             assertEquals(
50                 "Mike",
51                 economyFlight.getPassengersList().get(0).getName()
52             );
53             assertEquals(true, economyFlight.removePassenger(mike));
54             assertEquals(0, economyFlight.getPassengersList().size());
55         }
56
57         @Test
58         public void testEconomyFlightVipPassenger() {
59             Passenger james = new Passenger("James", true);
60             assertEquals("1", economyFlight.getId());
61             assertEquals(true, economyFlight.addPassenger(james));
62             assertEquals(1, economyFlight.getPassengersList().size());
63             assertEquals(
64                 "James",
65                 economyFlight.getPassengersList().get(0).getName()
66             );
67             assertEquals(false, economyFlight.removePassenger(james));
68             assertEquals(1, economyFlight.getPassengersList().size());
69         }
70     }
71 }

```

Gambar II.6: Class *EconomyFlightTest* di Class *AirportTest*

```

40
41     @DisplayName("Given there is a business flight")
42     @Nested
43     class BusinessFlightTest {
44
45         private Flight businessFlight;
46
47         @BeforeEach
48         void setUp() {
49             businessFlight = new BusinessFlight("2");
50         }
51
52         @Test
53         public void testBusinessFlightRegularPassengers() {
54             Passenger mike = new Passenger("Mike", false);
55
56             // assertEquals("2", businessFlight.getId());
57             assertEquals(false, businessFlight.addPassenger(mike));
58             assertEquals(0, businessFlight.getPassengersList().size());
59             assertEquals(false, businessFlight.removePassenger(mike));
60             assertEquals(0, businessFlight.getPassengersList().size());
61         }
62
63         @Test
64         public void testBusinessFlightVipPassenger() {
65             Passenger james = new Passenger("James", true);
66             assertEquals(true, businessFlight.addPassenger(james));
67             assertEquals(1, businessFlight.getPassengersList().size());
68             assertEquals(false, businessFlight.removePassenger(james));
69             assertEquals(1, businessFlight.getPassengersList().size());
70         }
71     }
72
73     /**
74      * Rigorous Test :-)
75      */
76     @Test
77     public void shouldAnswerWithTrue() {
78         assertTrue(true);
79     }
80 }

```

Gambar II.7: Class *BusinessFlightTest* di Class *AirportTest*

```

[arney1@arney-82kt my-app]$ mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.mycompany.app:my-app >-----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ my-app ---
[INFO] skip non existing resourceDirectory /home/arney1/Documents/School/Spring 2025/Programming Techniques/P/10/my-app/src/main/resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ my-app ---
[INFO] skip non existing resourceDirectory /home/arney1/Documents/School/Spring 2025/Programming Techniques/P/10/my-app/src/test/resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date.
[INFO]
[INFO] --- surefire:3.3.0:test (default-test) @ my-app ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.mycompany.app.AirportTest
[INFO] Running com.mycompany.app.AirportTest$BusinessFlightTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.010 s -- in com.mycompany.app.AirportTest$BusinessFlightTest
[INFO] Running com.mycompany.app.AirportTest$EconomyFlightTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.008 s -- in com.mycompany.app.AirportTest$EconomyFlightTest
[INFO] Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.068 s -- in com.mycompany.app.AirportTest
[INFO] Running com.mycompany.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 s -- in com.mycompany.app.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 1.118 s
[INFO] Finished at: 2025-04-28T14:10:14+07:00
[INFO]

```

Gambar II.8: Output “mvn test” dengan hasil Success

II.1.2. Penjelasan

1. Deskripsi Kasus

Sistem manajemen penerbangan, dapat mengelola penumpang dalam dua jenis penerbangan, dan mengatur aturan siapa saja yang boleh naik/turun berdasarkan tipe penumpang.

2. Penjelasan AirportTest

1. Testing EconomyFlight

1. Membuat grup tes khusus untuk EconomyFlight.

1. `@Nested` digunakan untuk membuat pengelompokan yang lebih rapi.
2. `@DisplayName` hanya untuk memberi label saat test dijalankan, supaya output di laporan lebih deskriptif.

2. `@BeforeEach`

```

void setUp() {
    economyFlight = new EconomyFlight("1");
}

```

1. Ini akan selalu dijalankan sebelum setiap tes dalam kelas `EconomyFlightTest`.
 2. Membuat instance baru `economyFlight` setiap kali supaya tes tidak saling memengaruhi.
 3. Test 1: Regular Passenger di `EconomyFlight`
 1. Membuat penumpang biasa (`false` berarti dia bukan VIP).
 2. Menguji:
 1. `addPassenger(mike)` berhasil (`true`).
 2. Penumpang masuk ke daftar (`passengersList` bertambah 1).
 3. Nama penumpang betul ("Mike").
 4. `removePassenger(mike)` berhasil (`true`), lalu daftar jadi kosong lagi.
 3. Jadi, Penumpang biasa boleh naik `EconomyFlight` dan boleh turun.
 4. Test 2: VIP Passenger di `EconomyFlight`
 1. Membuat penumpang VIP (`true` berarti VIP).
 2. Menguji:
 1. `addPassenger(james)` berhasil (`true`).
 2. Penumpang tercatat di daftar.
 3. Tapi tidak boleh `removePassenger(james)` (hasil `false`).
 4. Penumpang tetap ada di daftar.
 3. Jadi, VIP boleh naik `EconomyFlight`, tapi tidak bisa turun.
2. Testing `BusinessFlight`
1. `@BeforeEach`
 1. Membuat objek `BusinessFlight` baru sebelum setiap tes
 2. Test 1: Regular Passenger di `BusinessFlight`
 1. Penumpang biasa tidak boleh naik ke `BusinessFlight` (`addPassenger false`).
 2. Tidak boleh menghapus (karena tidak pernah berhasil masuk).
 3. `passengersList` tetap 0.
 3. Test 2: VIP Passenger di `BusinessFlight`

1. VIP boleh naik BusinessFlight.
2. Tapi, tidak bisa turun (`removePassenger false`).
3. Tetap ada di daftar.

BAB III HASIL DAN ANALISIS

III.1. Hasil

Unit test yang dibuat berhasil memvalidasi bahwa penanganan penumpang biasa dan VIP pada EconomyFlight dan BusinessFlight sesuai dengan aturan yang ditentukan, dengan seluruh skenario pengujian berjalan sukses tanpa error.

III.2. Lesson Learned

1. Menulis tes terlebih dahulu (sebelum membuat fitur) membantu:
 1. Menentukan spesifikasi dan aturan program lebih jelas sejak awal.
 2. Menghindari kesalahan logika karena semuanya diuji dari awal.
2. Jika ada perubahan di kode, kita langsung tahu jika ada yang rusak karena test akan gagal.
3. Dengan TDD, design class menjadi lebih rapi, modular, dan fokus pada kebutuhan.

BAB IV KESIMPULAN DAN SARAN

IV.1. Kesimpulan

Penerapan Test-Driven Development (TDD) membantu memastikan aplikasi dikembangkan dengan lebih terarah, mengurangi risiko kesalahan, dan mempercepat deteksi bug, sehingga menghasilkan kode yang lebih stabil dan mudah dirawat.

LAMPIRAN

Lampiran 1 - Source Code .zip

Telampir di Google Classroom

Lampiran 2 - Link Repository GitHub

<https://github.com/Arney1/Test-Driven-Development-Java>