Praktikum Pertemuan 10

Disusun untuk memenuhi tugas mata kuliah Pemrograman Berorientasi Objek (Praktek)

Disusun Oleh:
Dwika Ali Ramdhan (231511042)
2B – D3

Jurusan Teknik Komputer dan Informatika
Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung

2024

Contents

1. Listing 20.1 Passanger Class

1.1.Passenger.java

```
public class Passenger {
    private final String name;
    private final boolean vip;

    public Passenger(String name, boolean vip) {
        this.name = name;
        this.vip = vip;
    }

    public String getName() {
        return name;
    }

    public boolean isVip() {
        return vip;
    }
}
```

1.2. Penjelasan :

Kelas ini berfungsi untuk mendefinisikan penumpang pada penerbangan dengan atribut utama name (nama) dan vip (status VIP). Pada kelas ini terdapat konstruktor untuk menginisialisasi nilai nama dan status VIP, serta getter untuk mengakses nilai-nilai tersebut. Status VIP digunakan sebagai pembeda antara penumpang reguler dan penumpang VIP, yang akan mempengaruhi apakah penumpang dapat ditambahkan atau dihapus dari penerbangan tertentu.

2. Listing 20.2 Flight Class

2.1. Flight.java

```
public class Flight {
    private final String id;
    public final List<Passenger> passengers = new ArrayList<Passenger>();
    private final 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;
    }

    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);
        }
    }
}
```

2.2.Penjelasan

Kelas ini memiliki atribut utama seperti id (identifikasi penerbangan), passengers (daftar penumpang), dan flightType (jenis penerbangan). Konstruktor dalam kelas ini menginisialisasi id dan jenis penerbangan, sedangkan daftar penumpang dimulai sebagai daftar kosong. Jenis penerbangan dapat berupa "Economy" atau "Business," yang menentukan aturan khusus terkait penambahan atau penghapusan penumpang: Jika penerbangan bertipe "Economy", semua penumpang dapat ditambahkan. Namun, hanya penumpang reguler yang bisa dihapus. Untuk penerbangan bertipe "Business", hanya penumpang VIP yang dapat ditambahkan, dan tidak ada penumpang yang dapat dihapus.Case 3 : Throwing Exceptions

3. Listing 20.3 Airport Class, including the Main Method

3.1. Airport.java

```
public class Airport {
    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());
        }
    }
}
```

3.2. Penjelasan

Kelas ini berfungsi sebagai klien yang mengelola operasi dasar untuk dua jenis penerbangan, yaitu penerbangan ekonomi dan bisnis. Pada kelas ini terdapat metode main yang digunakan untuk menjalankan kode uji awal tanpa metode pengujian otomatis. Dalam metode main, aplikasi mencoba menambah dan menghapus penumpang sesuai dengan aturan masing-masing jenis penerbangan, serta menampilkan daftar penumpang setelah penambahan dan penghapusan dilakukan.

4. Listing 20.4 Junit 5 Dependencies added to the pom.xml

4.1. pom.xml Dependecies

```
<dependencies>
   <dependency>
         <groupId>org.junit.jupiter</groupId>
         <artifactId>junit-jupiter-api</artifactId>
         <version>5.6.0</version>
         <scope>test</scope>
   </dependency>
   <dependency>
         <groupId>org.junit.jupiter</groupId>
         <artifactId>junit-jupiter-engine</artifactId>
         <version>5.6.0</version>
         <scope>test</scope>
   </dependency>
</dependencies>
```

5. Listing 20.5 Testing the business logic for an economic flight

5.1. AirportTest.java

```
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());
        }

        @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());
        }
    }
```

5.2. Penjelasan

Kelas AirportTest sebagai pengujian unit untuk kelas Flight. Menggunakan JUnit untuk menguji fungsionalitas penambahan dan penghapusan penumpang berdasarkan tipe penerbangan (Ekonomi dan Bisnis). Pada pengetesan regular, dibuat objek passenger dengan nama Mike dan juga bukan vip. Dimana terdapat assert untuk memastikan id dari economy flight adalah 1, lalu jumlah penumpang adalah satu, nama penumpang yang pertama adalah Mike, dan pengetesan apakah remove passenger berhasil, lalu yang terakhir pengecekan size passenger list apakah 0. Sedangkan yang vip tidak beda jauh dengan pengetesan yang regular, yang berbeda adalah di argument pada pemanggilan object, dimana untuk vip-nya adalah true

6. Listing 20.6 Testing the business logic

6.1. AirPortTest.java

```
[…]
@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);

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

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

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

6.2. Penjelasan

Kelas AirportTest sebagai pengujian unit untuk kelas Flight. Menggunakan JUnit untuk menguji fungsionalitas penambahan dan penghapusan penumpang berdasarkan tipe penerbangan (Ekonomi dan Bisnis). Untuk pengetesannya sendiri tidak beda jauh dengan pengetesan pada economy flight, dimana ada pembuatan objek passenger, menambahkan passenger pada list passenger, mengecek size dari list passenger, mengetes remove passenger, dan juga mengecek size dari list passenger.

7. Listing 20.7 Abstract Flight class, the basis of the hierarchy

7.1. Flight.java

```
public abstract class Flight {
    private String id;
    List<Passenger> passengers = new ArrayList<Passenger>();

    public Flight(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }

    public List<Passenger> getPassengersList() {
        return Collections.unmodifiableList(passengers);
    }

    public abstract boolean addPassenger(Passenger passenger);

    public abstract boolean removePassenger(Passenger passenger);
}
```

7.2. Penjelasan

Kelas Flight dideklarasikan sebagai kelas abstrak, menjadikannya dasar hierarki untuk tipe-tipe penerbangan yang akan diturunkan. Ini berarti Flight tidak dapat diinstansiasi secara langsung, tetapi memberikan struktur umum bagi kelas turunannya. Dimana di sini terdapat beberapa perubahan seperti pada method addPassanger dan juga removePassenger yang menjadi sebuah abstract method dan juga atribut FlightType yang sudah tidak ada.

8. Listing 20.8 EconomyFlight class, extending the abstract

8.1. EconomyFlight.java

```
public class EconomyFlight extends Flight {
    public EconomyFlight(String id) {
        super(id);
    }

    @Override
    public boolean addPassenger(Passenger passenger) {
        return passengers.add(passenger);
    }

    @Override
    public boolean removePassenger(Passenger passenger) {
        if (!passenger.isVip()) {
            return passengers.remove(passenger);
        }
        return false;
    }
}
```

8.2. Penjelasan

Di atas adalah sub class dari Flight yang mana merepresentasikan flight type economy yang di dalamnya mengimplementasikan/me-override method abstract di class Flight. Konstruktor EconomyFlight(String id) Konstruktor ini memanggil konstruktor kelas induk Flight dengan parameter id, yang mewakili ID penerbangan. Implementasi Metode addPassenger Metode ini mengesampingkan (@Override) metode abstrak addPassenger dari Flight untuk menambahkan penumpang ke daftar passengers. Implementasi Metode removePassenger Metode ini juga mengesampingkan metode abstrak removePassenger dari Flight. Pada metode ini, pengecekan dilakukan untuk memastikan bahwa penumpang yang dihapus bukanlah penumpang VIP. Jika

penumpang bukan VIP, maka ia dapat dihapus dari daftar penumpang

9. Listing 20.9 BusinessFlight class, extending the abstract

9.1. BusinessFlight.java

```
public class BusinessFlight extends Flight {
    public BusinessFlight(String id) {
        super(id);
    }

    @Override
    public boolean addPassenger(Passenger passenger) {
        if (passenger.isVip()) {
            return passengers.add(passenger);
        }
        return false;
    }

    @Override
    public boolean removePassenger(Passenger passenger) {
        return false;
    }
}
```

9.2. Penjelasan

Di atas merupakan class Business Flight yang merepresentasikan flight type business dan juga mewariskan method abstract add and remove passenger. Di mana yang agak berbeda disini adalah pada add passenger terdapat pengecekan apakah passenger tersebut merupakan vip atau bukan. Konstruktor BusinessFlight(String id) Konstruktor ini memanggil konstruktor kelas induk Flight dengan parameter id, yang merepresentasikan ID penerbangan bisnis. Implementasi Metode addPassenger Metode ini mengesampingkan (@Override) metode abstrak addPassenger dari Flight untuk menambahkan penumpang ke daftar passengers. Pada metode ini, hanya penumpang VIP yang bisa ditambahkan ke penerbangan bisnis, yang diperiksa melalui kondisi if (passenger.isVip()). Jika penumpang adalah VIP, maka ia ditambahkan ke daftar passengers, dan metode ini mengembalikan true. Jika bukan VIP, maka penumpang tidak ditambahkan, dan metode ini mengembalikan false. Implementasi Metode removePassenger Metode ini juga mengesampingkan metode abstrak removePassenger dari Flight. Dalam penerbangan bisnis, tidak ada penumpang yang dapat dihapus dari daftar passengers, sehingga metode ini selalu mengembalikan false, baik untuk penumpang biasa maupun VIP.

10. Listing 20.10 Refactoring propagation into the AirportTest class

10.1. AirportTest

```
public class AirportTest {
    @DisplayName("Given there is an economy flight")
    @Nested
    class EconomyFlightTest {
        private Flight economyFlight;

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

        @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());
        }

        @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());
        }
    }

    //20.6
    @DisplayName("Given there is a business flight")
    @Nested
    class BusinessFlightTest {
        private Flight businessFlight;

        @BeforeEach
        void setUp() {
            businessFlight = new BusinessFlight("2");
        }

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

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

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

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

10.2. Penjelasan

Kode di atas adalah kelas AirportTest, yang menguji dua jenis penerbangan, yaitu EconomyFlight dan BusinessFlight, menggunakan kelas bersarang untuk pengujian terpisah. Pada kelas EconomyFlightTest, diuji apakah penumpang biasa dapat ditambahkan dan dihapus dari daftar penumpang, sementara penumpang VIP hanya dapat ditambahkan tetapi tidak dapat dihapus. Di sisi lain, dalam BusinessFlightTest, penumpang biasa tidak dapat ditambahkan atau dihapus dari penerbangan bisnis, sedangkan penumpang VIP dapat ditambahkan tetapi tidak dapat dihapus. Kode ini melakukan pengujian dengan instansiasi EconomyFlight dan BusinessFlight untuk memeriksa perilaku yang berbeda dalam penanganan penumpang biasa dan VIP.

LAMPIRAN

LINK GITHUB: https://github.com/DAliRIJTK/PBO_Dwika_231511042.git