LAPORAN PRAKTIKUM
PEMOGRAMAN BERBASIS OBJECT
“Pertemuan ke-11”

Disusun oleh :

Alifah Fisalsabilawati

201511035

2B – D3 Teknik Informatika

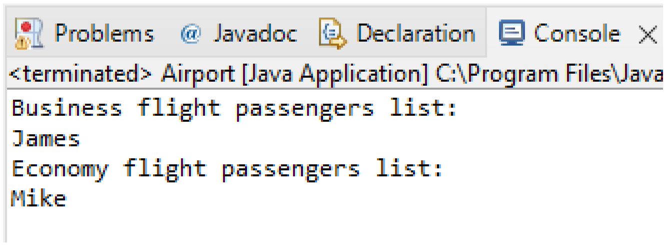
Jurusan Teknik Komputer dan Informatika

Program Studi D3 Teknik Informatika

Politeknik Negeri Bandung

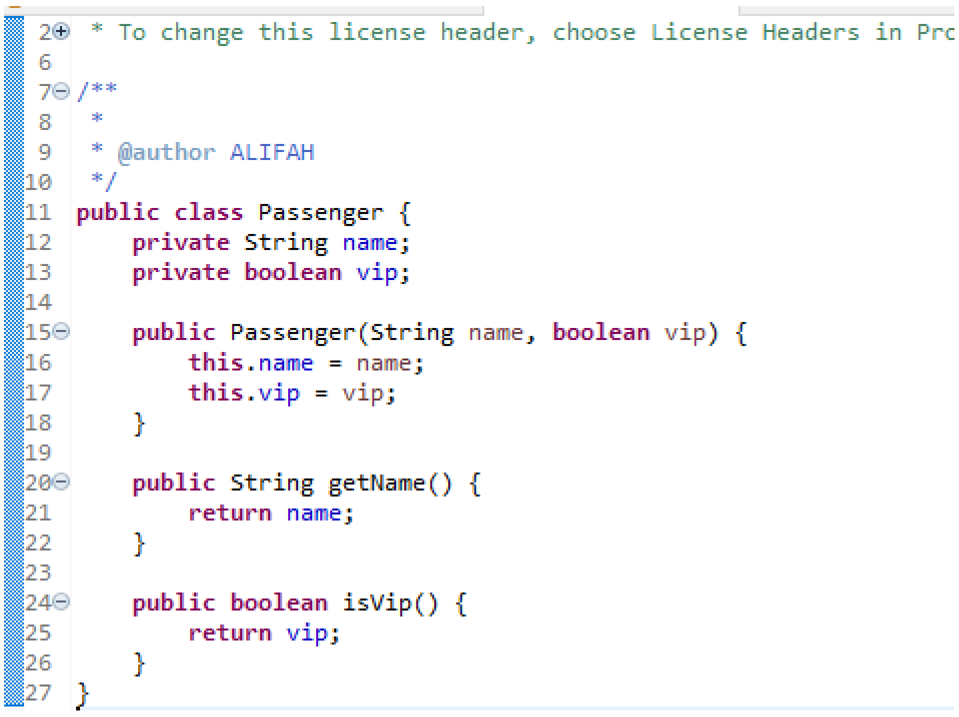
20.1 – 20.3 Running class Passenger, Flight, Airport.java

Hasil output



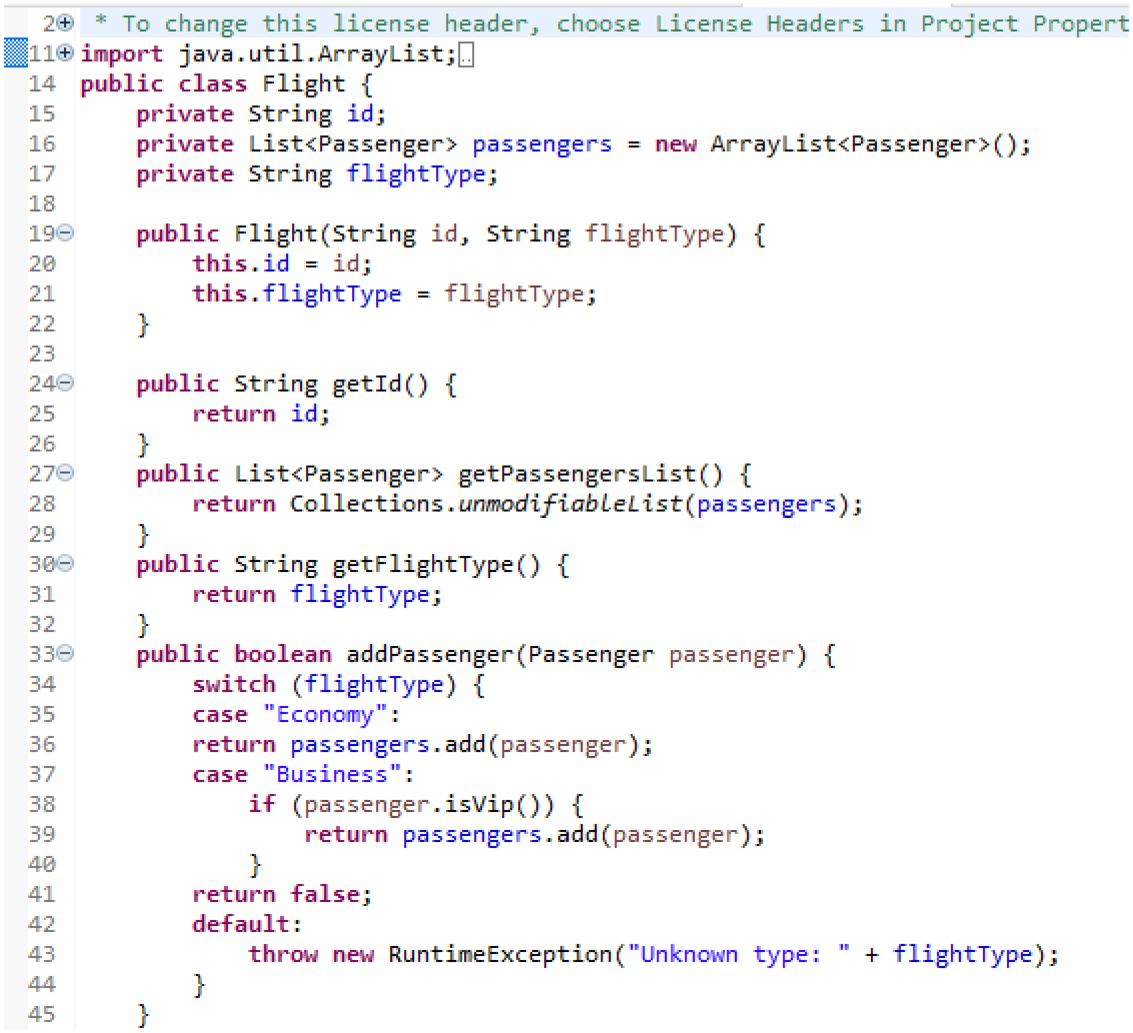
```
<terminated> Airport [Java Application] C:\Program Files\Java
Business flight passengers list:
James
Economy flight passengers list:
Mike
```

Passenger.java

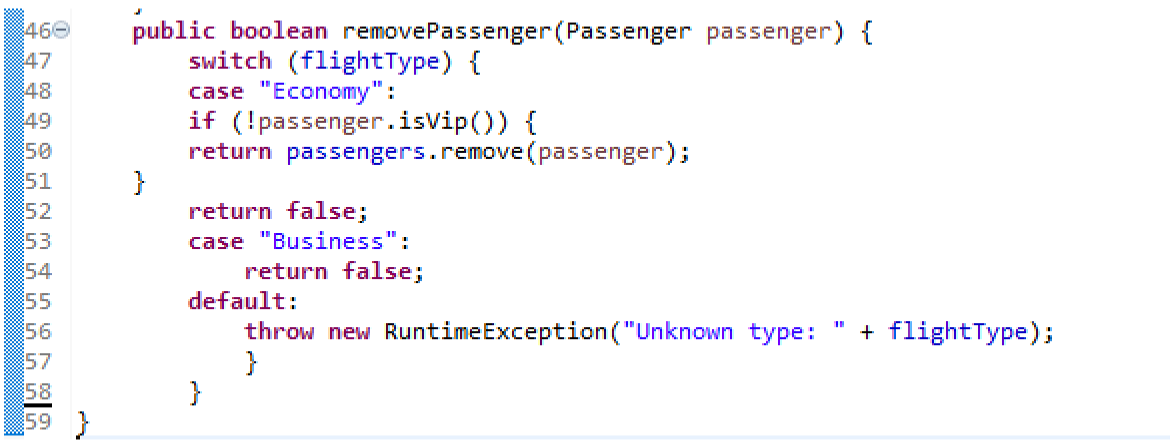


```java
 * To change this license header, choose License Headers in Pro

/**
 *
 * @author ALIFAH
 */
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;
    }
}
```

Flight.java

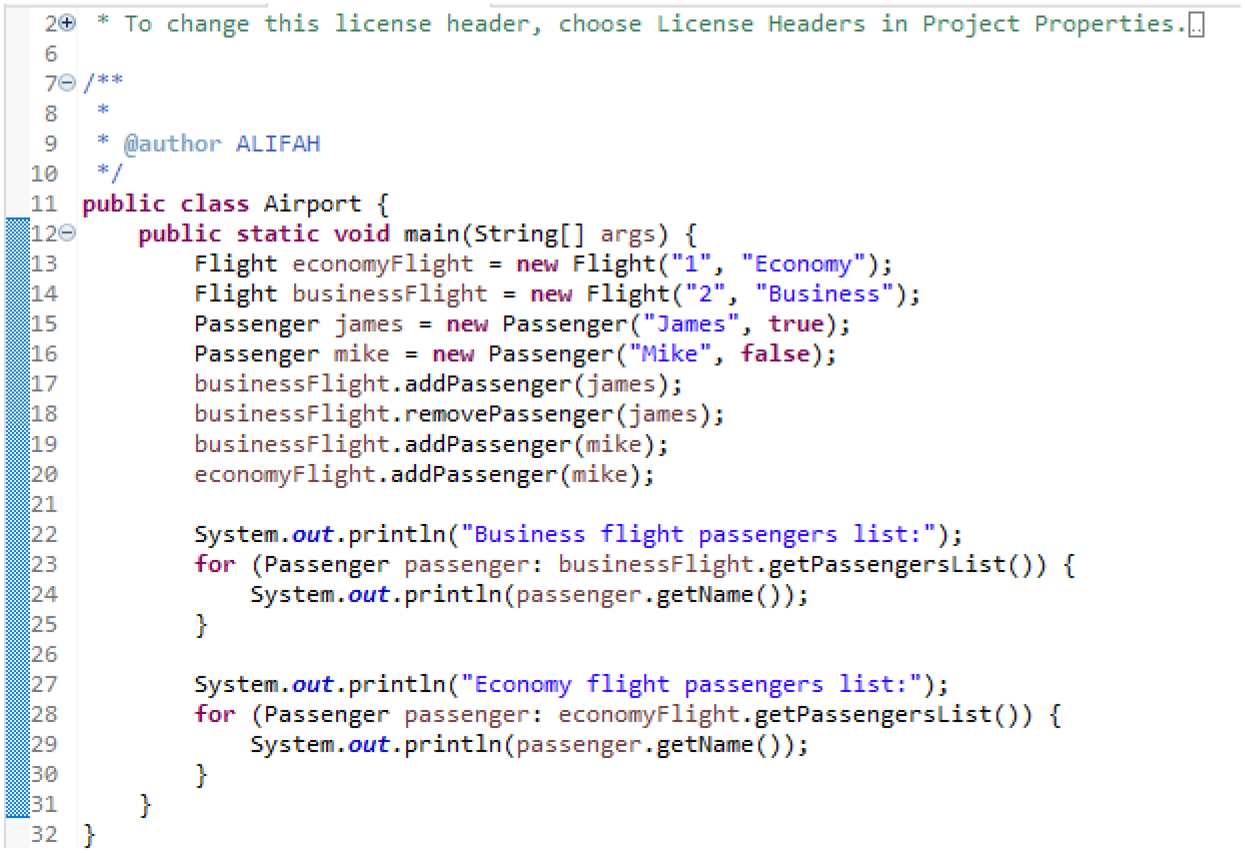


```java
 * To change this license header, choose License Headers in Project Propert
import java.util.ArrayList;
public class Flight {
    private String id;
    private List<Passenger> passengers = new ArrayList<Passenger>();
    private String flightType;

    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;
    }
    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);
        }
    }
```



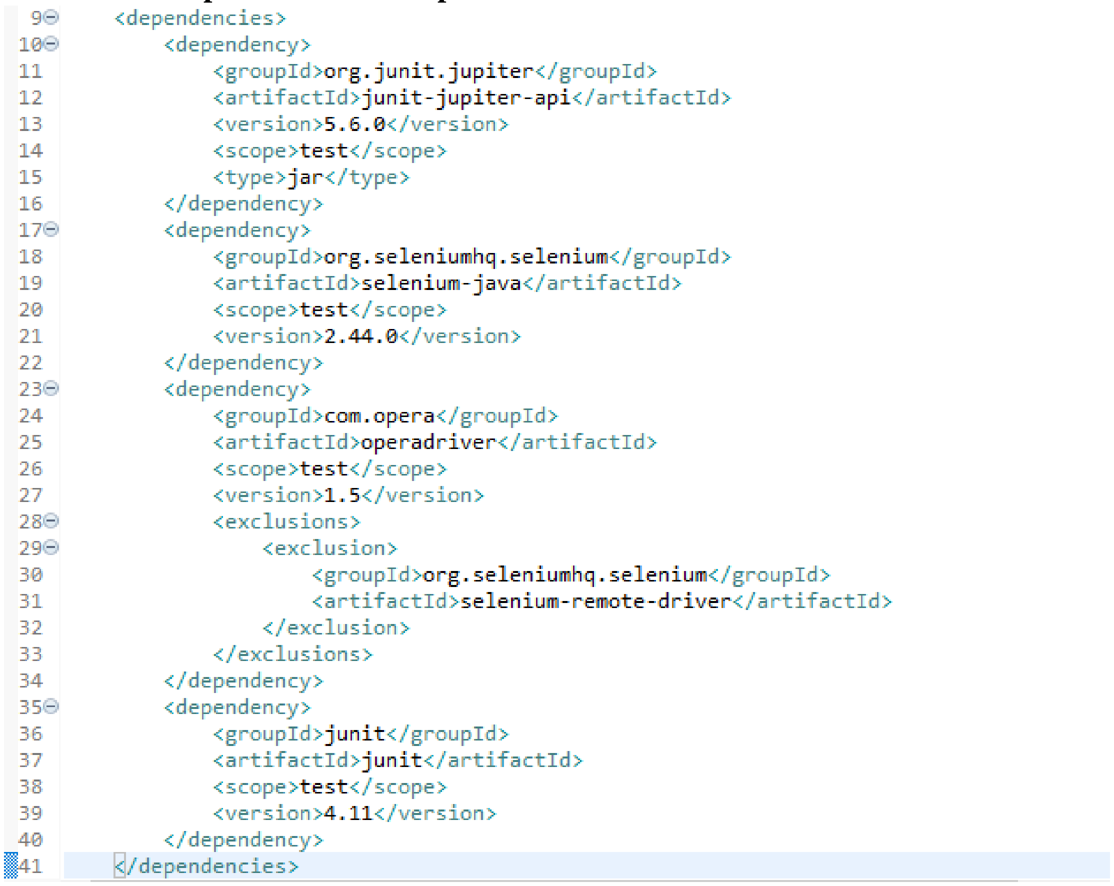
```java
    public boolean removePassenger(Passenger passenger) {
        switch (flightType) {
        case "Economy":
        if (!passenger.isVip()) {
        return passengers.remove(passenger);
    }
        return false;
        case "Business":
            return false;
        default:
            throw new RuntimeException("Unknown type: " + flightType);
            }
        }
}
```

Airport.java



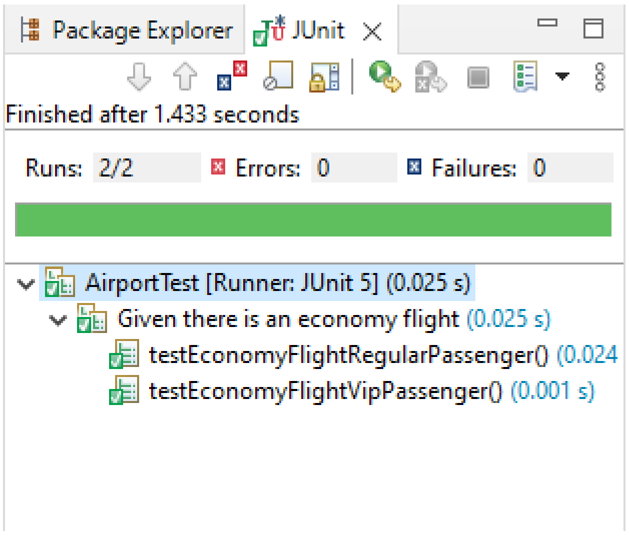
```
 * To change this license header, choose License Headers in Project Properties.

/**
 *
 * @author ALIFAH
 */
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());
        }
    }
}
```

20.4 Junit 5 depedencies add to pom.xml



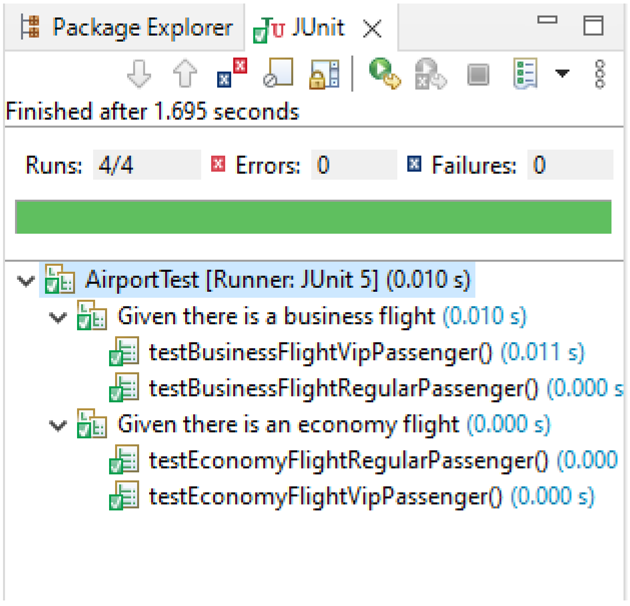
```
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.6.0</version>
            <scope>test</scope>
            <type>jar</type>
        </dependency>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <scope>test</scope>
            <version>2.44.0</version>
        </dependency>
        <dependency>
            <groupId>com.opera</groupId>
            <artifactId>operadriver</artifactId>
            <scope>test</scope>
            <version>1.5</version>
            <exclusions>
                <exclusion>
                    <groupId>org.seleniumhq.selenium</groupId>
                    <artifactId>selenium-remote-driver</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <scope>test</scope>
            <version>4.11</version>
        </dependency>
    </dependencies>
```

20.5 testing the business logic for an economic flight



```
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
        testEconomyFlightVipPassenger() (0.001 s)
```

20.6 testing the business logic for an business flight



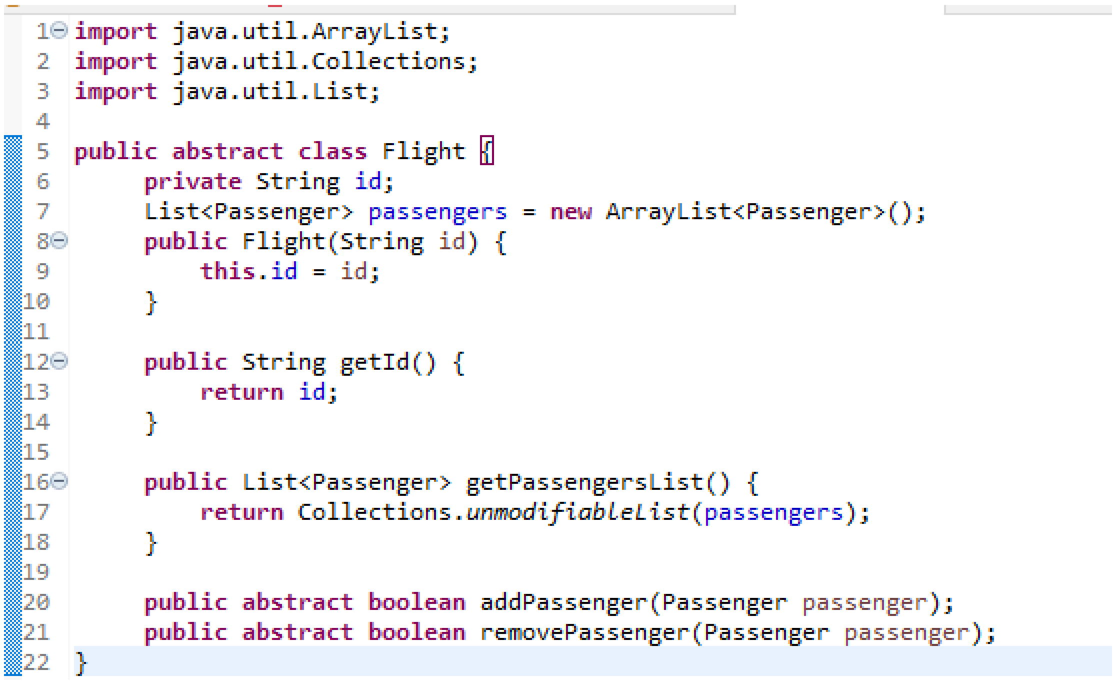
Package Explorer JUnit

Finished after 1.695 seconds

Runs: 4/4 Errors: 0 Failures: 0

```
AirportTest [Runner: JUnit 5] (0.010 s)
    Given there is a business flight (0.010 s)
        testBusinessFlightVipPassenger() (0.011 s)
        testBusinessFlightRegularPassenger() (0.000 s
    Given there is an economy flight (0.000 s)
        testEconomyFlightRegularPassenger() (0.000
        testEconomyFlightVipPassenger() (0.000 s)
```

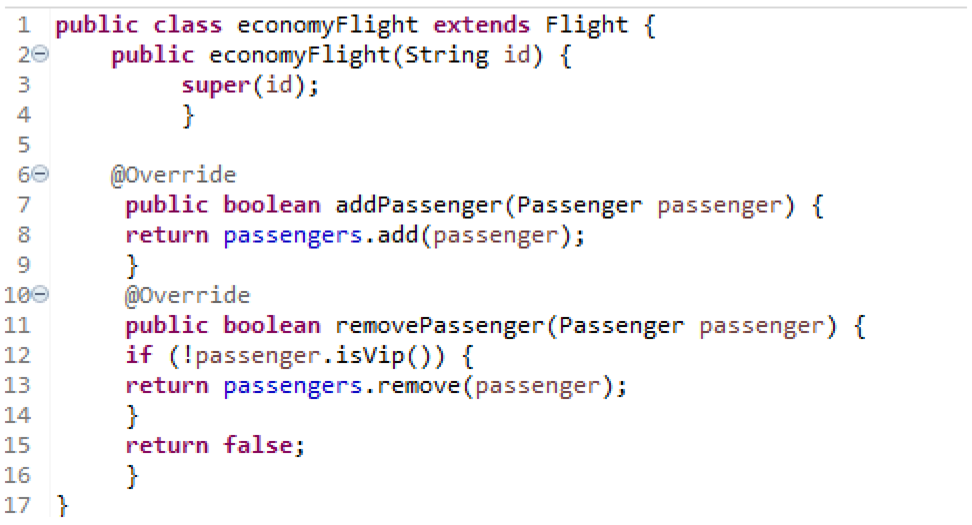
20.7 Refactoring class flight



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

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);
}
```

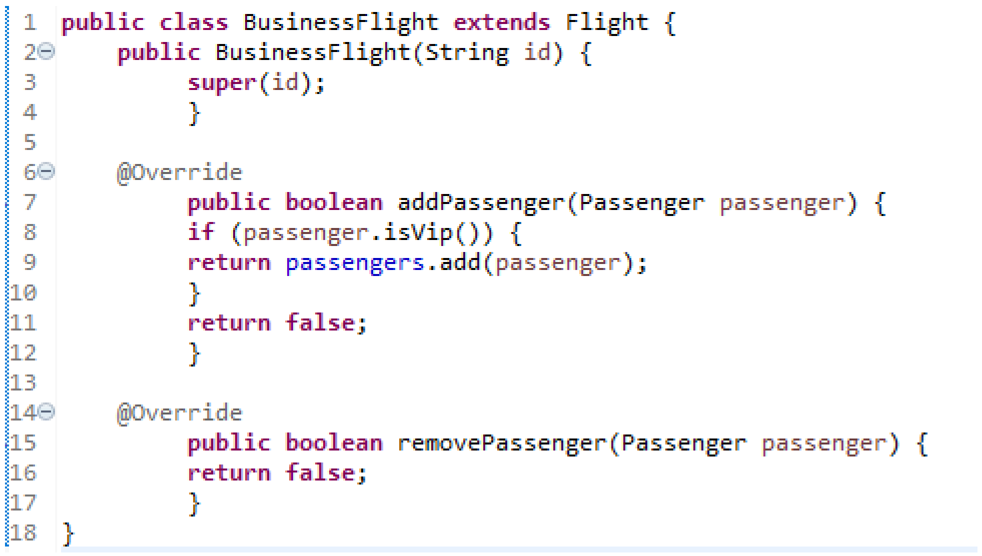
20.8 – 20.9 membuat economyFlight dan BusinessFlight

economyFlight.java



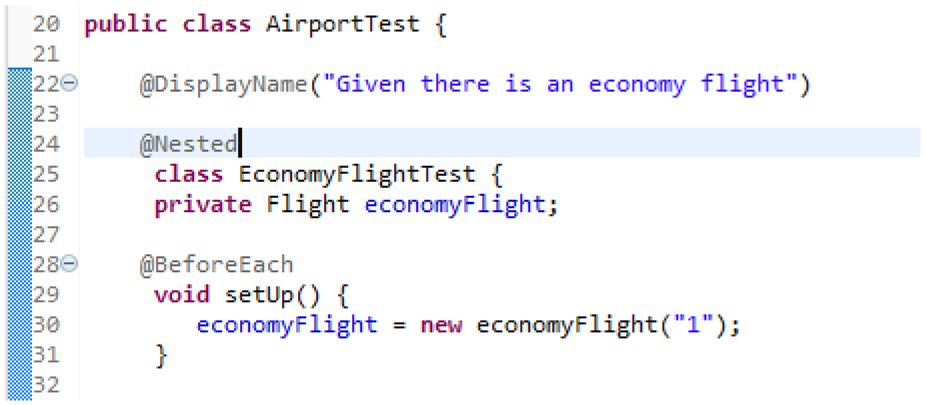
```java
public class economyFlight extends Flight {
    public economyFlight(String id) {
         super(id);
         }

    @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;
     }
}
```

BusinessFlight.java

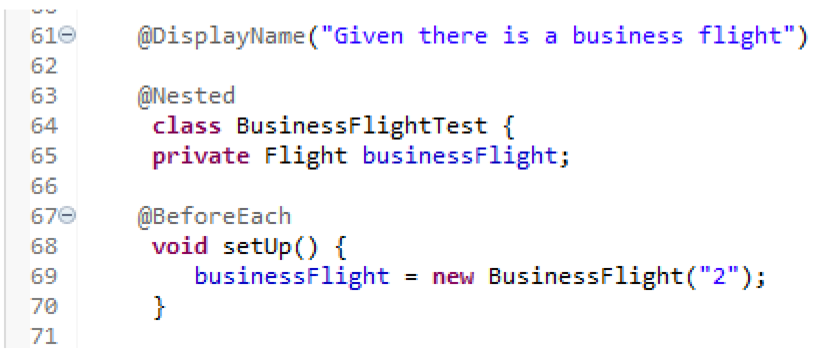


```java
public class BusinessFlight extends Flight {
    public BusinessFlight(String id) {
         super(id);
         }

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

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

20.10 Refactoring class AirportTest



```java
public class AirportTest {

    @DisplayName("Given there is an economy flight")

    @Nested
     class EconomyFlightTest {
     private Flight economyFlight;

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

```



```java
    @DisplayName("Given there is a business flight")

    @Nested
     class BusinessFlightTest {
     private Flight businessFlight;

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

```