Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «Современные платформы программирования»

Выполнил:
Студент 3 курса
Группы ПО-8
Шлыков А.Л.
Проверил:
Крощенко А.А.

Брест 2024

Цель работы:

приобрести практические навыки в области объектно-ориентированного проектирования

Задание 1

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

11) interface Устройство Печати ← class Принтер ← class Лазерный Принтер.

```
interface PrintingDevice {
    void print();
}

class Printer implements PrintingDevice {
    public void print() {
        System.out.println("Printing from printer...");
    }
}

class LaserPrinter extends Printer {
    @Override
    public void print() {
        System.out.println("Printing from laser printer...");
    }
}
public class task1{
    public static void main(String[] args) {
        Printer printer = new Printer();
        printer.print();

        LaserPrinter laserPrinter = new LaserPrinter();
        laserPrinter.print();
    }
}
```

```
[Running] cd "/Users/harweast/Documents/3kurs/sem2/SPP/laba5/5/src/" && javac task1.java && java task1
Printing from printer...
Printing from laser printer...

[Done] exited with code=0 in 0.315 seconds
```

Задание 2

В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

3) Создать суперкласс Музыкальный инструмент и классы Ударный, Струнный, Духовой. Создать массив объектов Оркестр. Осуществить вывод состава оркестра.

```
abstract class MusicalInstrument {
    String name;
```

```
    public MusicalInstrument(String name) {
        this.name = name;
    }

    abstract void play();
}

class Percussion extends MusicalInstrument {
    public Percussion(String name) {
        super(name);
    }

    @Override
    void play() {
        System.out.println("Играет ударный инструмент: " + name);
    }
}

class Stringed extends MusicalInstrument {
    public Stringed(String name) {
        super(name);
    }

    @Override
    void play() {
        System.out.println("Играет струнный инструмент: " + name);
    }
}

class Wind extends MusicalInstrument {
    public Wind(String name) {
        super(name);
    }

    @Override
    void play() {
        System.out.println("Играет духовой инструмент: " + name);
    }
}

class Orchestra {
    MusicalInstrument[] orchestra;

    public Orchestra(MusicalInstrument[] orchestra) {
        this.orchestra = orchestra;
    }

    void play() {
        for (MusicalInstrument instrument : orchestra) {
            instrument.play();
        }
    }
}

public class task2{
    public static void main(String[] args) {
        Percussion drum = new Percussion("Барабан");
        Stringed violin = new Stringed("Скрипка");
        Wind flute = new Wind("Флейта");
```

```
        MusicalInstrument[] instruments = {drum, violin, flute};

        Orchestra orchestra = new Orchestra(instruments);
        orchestra.play();
    }
}
```

```
[Running] cd "/Users/harweast/Documents/3kurs/sem2/SPP/laba5/5/src/" && javac task2.java && java task2
Играет ударный инструмент: Барабан
Играет струнный инструмент: Скрипка
Играет духовой инструмент: Флейта

[Done] exited with code=0 in 0.432 seconds
```

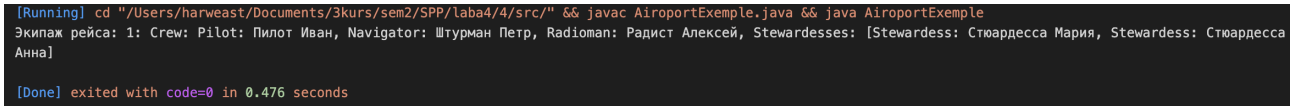
Задание 3

В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

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

abstract class Staff {
    String name;

    public Staff(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return this.getClass().getSimpleName() + ": " + this.name;
    }
}

class Crew {
    Pilot pilot;
    Navigator navigator;
    Radioman radioman;
    List<Stewardess> stewardesses;

    public Crew(Pilot pilot, Navigator navigator, Radioman radioman, List<Stewardess>
stewardesses) {
        this.pilot = pilot;
        this.navigator = navigator;
        this.radioman = radioman;
        this.stewardesses = stewardesses;
    }

    @Override
    public String toString() {
        return "Crew: " + pilot + ", " + navigator + ", " + radioman + ", Stewardesses: " +
stewardesses;
    }
}

class Plane {
    int capacity;
    int flightRange;
```

```
    public Plane(int capacity, int flightRange) {
        this.capacity = capacity;
        this.flightRange = flightRange;
    }

    @Override
    public String toString() {
        return "Plane: Capacity - " + capacity + ", Flight Range - " + flightRange;
    }
}

class Flight {
    Crew crew;
    Plane plane;
    Airport departureAirport;
    Airport destinationAirport;

    public Flight(Crew crew, Plane plane, Airport departureAirport, Airport destinationAirport) {
        this.crew = crew;
        this.plane = plane;
        this.departureAirport = departureAirport;
        this.destinationAirport = destinationAirport;
    }

    @Override
    public String toString() {
        return "Flight: " + crew + ", " + plane + ", Departure: " + departureAirport + ", Destination: " +
destinationAirport;
    }
}

class Airport {
    String name;

    public Airport(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Airport: " + name;
    }
}
class Administrator {
    void formCrew(Flight flight, Pilot pilot, Navigator navigator, Radioman radioman,
List<Stewardess> stewardesses) {
        Crew crew = new Crew(pilot, navigator, radioman, stewardesses);
        flight.crew = crew;
    }
}

public class AiroportExemple {
    public static void main(String[] args) {
        // Создаем персонал
        Pilot pilot = new Pilot("Пилот Иван");
        Navigator navigator = new Navigator("Штурман Петр");
        Radioman radioman = new Radioman("Радист Алексей");
        Stewardess stewardess1 = new Stewardess("Стюардесса Мария");
        Stewardess stewardess2 = new Stewardess("Стюардесса Анна");
```

```
        // Создаем список стюардесс
        List<Stewardess> stewardesses = new ArrayList<>();
        stewardesses.add(stewardess1);
        stewardesses.add(stewardess2);

        // Создаем аэропорты
        Airport departureAirport = new Airport("Москва");
        Airport destinationAirport = new Airport("Санкт-Петербург");

        // Создаем самолет
        Plane plane = new Plane(180, 2000);

        // Создаем рейс без экипажа
        Flight flight = new Flight(null, plane, departureAirport, destinationAirport);

        // Создаем администратора и формируем экипаж
        Administrator admin = new Administrator();
        admin.formCrew(flight, pilot, navigator, radioman, stewardesses);

        // Теперь у нас есть рейс с экипажем
        System.out.println("Экипаж рейса: " + flight.crew );
    }
}
```



Вывод: научился создавать и использовать классы в программах на языке программирования Java