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

Лабораторная работа №4

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 указанный класс, включив в него вспомогательный внутренний класс или классы.
Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.
2) Создать класс Payment (покупка) с внутренним классом, с помощью объектов которого можно
сформировать покупку из нескольких товаров.

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

public class Payment {
    private int orderId;
    private List<Item> items;

    public Payment(int orderId) {
        this.orderId = orderId;
        this.items = new ArrayList<>();
    }

    public void addItem(String name, double price, int quantity) {
        items.add(new Item(name, price, quantity));
    }

    public double calculateTotalAmount() {
        double total = 0;
        for (Item item : items) {
            total += item.getPrice() * item.getQuantity();
        }
        return total;
    }

    public void displayItems() {
        for (Item item : items) {
            System.out.println("Item: " + item.getName() + ", Price: " + item.getPrice() + ", Quantity: " +
item.getQuantity());
        }
    }

    private class Item {
        private String name;
        private double price;
        private int quantity;

        public Item(String name, double price, int quantity) {
            this.name = name;
            this.price = price;
            this.quantity = quantity;
        }

        public String getName() {
            return name;
```

```
        }

        public double getPrice() {
            return price;
        }

        public int getQuantity() {
            return quantity;
        }
    }

    public static void main(String[] args) {
        Payment payment = new Payment(12345);
        payment.addItem("Product 1", 10.99, 2);
        payment.addItem("Product 2", 15.49, 1);

        payment.displayItems();
        System.out.println("Total: $" + payment.calculateTotalAmount());
    }
}
```

```
[Running] cd "/Users/harweast/Documents/3kurs/sem2/SPP/laba4/4/src/" && javac Payment.java && java Payment
Item: Product 1, Price: 10.99, Quantity: 2
Item: Product 2, Price: 15.49, Quantity: 1
Total: $37.47
```

Задание 2

Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут (локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Продемонстрировать использование разработанных классов.

11) Создать класс Звездная система, используя классы Планета, Звезда.

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

public class StarSystem {
    private String name;
    private Star star;
    private List<Planet> planets;

    public StarSystem(String name, Star star) {
        this.name = name;
        this.star = star;
        this.planets = new ArrayList<>();
    }

    public void addPlanet(Planet planet) {
        planets.add(planet);
    }

    public void printStarSystemInfo() {
        System.out.println("StarSystem: " + name);
        star.printStarInfo();
        System.out.println("Planets in the system:");
        for (Planet planet : planets) {
```

```
            planet.printPlanetInfo();
        }
    }

    public static void main(String[] args) {
        Star sun = new Star("Sun", "Yellow", 5.0);
        StarSystem solarSystem = new StarSystem("Solar System", sun);

        Planet mercury = new Planet("Mercury", 0.33);
        Planet venus = new Planet("Venus", 4.87);
        Planet earth = new Planet("Earth", 5.97);
        solarSystem.addPlanet(mercury);
        solarSystem.addPlanet(venus);
        solarSystem.addPlanet(earth);

        solarSystem.printStarSystemInfo();
    }
}

class Star {
    private String name;
    private String type;
    private double mass;

    public Star(String name, String type, double mass) {
        this.name = name;
        this.type = type;
        this.mass = mass;
    }

    public void printStarInfo() {
        System.out.println("Star: " + name + ", Type: " + type + ", Mass: " + mass + " Solar Masses");
    }
}

class Planet {
    private String name;
    private double mass;

    public Planet(String name, double mass) {
        this.name = name;
        this.mass = mass;
    }

    public void printPlanetInfo() {
        System.out.println("Planet: " + name + ", Mass: " + mass + " Earth Masses");
    }
}
```

```
[Running] cd "/Users/harweast/Documents/3kurs/sem2/SPP/laba4/4/src/" && javac StarSystem.java && java StarSystem
StarSystem: Solar System
Star: Sun, Type: Yellow, Mass: 5.0 Solar Masses
Planets in the system:
Planet: Mercury, Mass: 0.33 Earth Masses
Planet: Venus, Mass: 4.87 Earth Masses
Planet: Earth, Mass: 5.97 Earth Masses

[Done] exited with code=0 in 0.398 seconds
```

Задание 3

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.
11) Система Аэрофлот. Администратор формирует летную Бригаду (пилоты, штурман, радист, стюардессы) на Рейс. Каждый Рейс выполняется Самолетом с определенной вместимостью и дальностью полета. Рейс может быть отменен из-за погодных условий в Аэропорту отлета или назначения. Аэропорт назначения может быть изменен в полете из-за технических неисправностей, о которых сообщил командир.

```
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) {

        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");


        List<Stewardess> stewardesses = new ArrayList<>();
        stewardesses.add(stewardess1);
        stewardesses.add(stewardess2);


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

        Plane plane = new Plane(180, 2000);
```

```
        Flight flight = new Flight(null, plane, departureAirport, destinationAirport);

        Administrator admin = new Administrator();
        admin.formCrew(flight, pilot, navigator, radioman, stewardesses);

        System.out.println("Экипаж рейса: " + flight.crew );
    }
}
```

```
[Running] cd "/Users/harweast/Documents/3kurs/sem2/SPP/laba4/4/src/" && javac AiroportExemple.java && java AiroportExemple
Экипаж рейса: 1: Crew: Pilot: Пилот Иван, Navigator: Штурман Петр, Radioman: Радист Алексей, Stewardesses: [Stewardess: Стюардесса Мария, Stewardess: Стюардесса
Анна]

[Done] exited with code=0 in 0.476 seconds
```

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