МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №4

Выполнил:
студент 3 курса
группы ПО-9
Тусюк Т.В.

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.

9) Создать класс **Mobile** с внутренним классом, с помощью объектов которого можно хранить информацию о моделях телефонов и их свойствах.

Выполнение задания:

```
import java.util.HashMap;
import java.util.Map;

public class Mobile {

    private class PhoneModel {
        private String modelName;
        private String brand;
        private int yearReleased;

        public PhoneModel(String modelName, String brand, int yearReleased) {
            this.modelName = modelName;
            this.brand = brand;
            this.yearReleased = yearReleased;
        }

        public String getModelName() {
            return modelName;
        }

        public String getBrand() {
            return brand;
        }

        public int getYearReleased() {
            return yearReleased;
        }
    }

    private Map<String, PhoneModel> phoneModels;

    public Mobile() {
        phoneModels = new HashMap<>();
    }

    public void addPhoneModel(String modelName, String brand, int yearReleased) {
        PhoneModel model = new PhoneModel(modelName, brand, yearReleased);
        phoneModels.put(modelName, model);
    }

    public void getPhoneModelInfo(String modelName) {
        PhoneModel model = phoneModels.get(modelName);
        if (model != null) {
            System.out.println("Model Name: " + model.getModelName());
            System.out.println("Brand: " + model.getBrand());
```

```
            System.out.println("Year Released: " + model.getYearReleased());
        } else {
            System.out.println("Model not found!");
        }
    }

    public void displayAllPhoneModels() {
        System.out.println("All Phone Models:");
        for (PhoneModel model : phoneModels.values()) {
            System.out.println("Model Name: " + model.getModelName() +
                    ", Brand: " + model.getBrand() +
                    ", Year Released: " + model.getYearReleased());
        }
    }

    public static void main(String[] args) {
        Mobile mobile = new Mobile();

        mobile.addPhoneModel("iPhone 13", "Apple", 2021);
        mobile.addPhoneModel("Galaxy S21", "Samsung", 2021);
        mobile.addPhoneModel("Pixel 6", "Google", 2021);

        mobile.getPhoneModelInfo("iPhone 13");

        mobile.displayAllPhoneModels();
    }
}
}
```

Результат:

```
Model Name: iPhone 13
Brand: Apple
Year Released: 2021
All Phone Models:
Model Name: iPhone 13, Brand: Apple, Year Released: 2021
Model Name: Galaxy S21, Brand: Samsung, Year Released: 2021
Model Name: Pixel 6, Brand: Google, Year Released: 2021
```

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.

9) Создать класс Автомобиль, используя класс Колесо.

Выполнение задания:

Wheel.java

```
public class Wheel {
    private double diameter;
    private boolean isPunctured;

    public Wheel(double diameter) {
        this.diameter = diameter;
```

```
        this.isPunctured = false;
    }

    public boolean isPunctured() {
        return isPunctured;
    }

    public void puncture() {
        isPunctured = true;
        System.out.println("The wheel is punctured!");
    }

    public void replaceWheel() {
        isPunctured = false;
        System.out.println("The wheel is replaced.");
    }

    public double getDiameter() {
        return diameter;
    }
}
```

Car.java

```
public class Car {
    private String brand;
    private String model;
    private Wheel[] wheels;

    public Car(String brand, String model, double wheelDiameter) {
        this.brand = brand;
        this.model = model;
        this.wheels = new Wheel[4];
        for (int i = 0; i < wheels.length; i++) {
            wheels[i] = new Wheel(wheelDiameter);
        }
    }

    public boolean isAnyWheelPunctured() {
        for (Wheel wheel : wheels) {
            if (wheel.isPunctured()) {
                return true;
            }
        }
        return false;
    }

    public void replacePuncturedWheel() {
        for (Wheel wheel : wheels) {
            if (wheel.isPunctured()) {
                wheel.replaceWheel();
                return;
            }
        }
        System.out.println("No punctured wheels found.");
    }

    public void displayCarInfo() {
        System.out.println("Car: " + brand + " " + model);
        System.out.println("Wheels:");
        for (int i = 0; i < wheels.length; i++) {
```

```
            System.out.println("Wheel " + (i + 1) + ": Diameter - " +
wheels[i].getDiameter() + " inches, Punctured - " + (wheels[i].isPunctured() ? "Yes"
: "No"));
        }
    }

    public Wheel[] getWheels() {
        return wheels;
    }

    public static void main(String[] args) {
        Car car = new Car("Toyota", "Camry", 16.0);

        car.getWheels()[2].puncture();

        car.displayCarInfo();

        car.replacePuncturedWheel();

        car.displayCarInfo();
    }
}
```

Результат:

```
The wheel is punctured!
Car: Toyota Camry
Wheels:
Wheel 1: Diameter - 16.0 inches, Punctured - No
Wheel 2: Diameter - 16.0 inches, Punctured - No
Wheel 3: Diameter - 16.0 inches, Punctured - Ye
Wheel 4: Diameter - 16.0 inches, Punctured - No
The wheel is replaced.
Car: Toyota Camry
Wheels:
Wheel 1: Diameter - 16.0 inches, Punctured - No
Wheel 2: Diameter - 16.0 inches, Punctured - No
Wheel 3: Diameter - 16.0 inches, Punctured - No
Wheel 4: Diameter - 16.0 inches, Punctured - No
```

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.

9) Система Железнодорожная касса. Пассажир делает Заявку на станцию назначения, время и дату поездки. Система регистрирует Заявку и осуществляет поиск подходящего Поезда.Пассажир делает выбор Поезда и получает Счет на

оплату. Администратор вводит номера Поездов, промежуточные и конечные станции, цены.

Выполнение задания:

```
import java.util.*;

class Passenger {
    private String name;
    private String passportNumber;

    public Passenger(String name, String passportNumber) {
        this.name = name;
        this.passportNumber = passportNumber;
    }
    @Override
    public String toString() {
        return "Пассажир: " + name + ", Паспорт: " + passportNumber;
    }

    public void createRequest(String destinationStation, Date dateTime) {

        System.out.println("Заявка создана на станцию: " + destinationStation + " на
" + dateTime);
    }

    public void chooseTrain(ArrayList<Train> trains) {

        Train chosenTrain = trains.get(0);
        System.out.println("Выбран поезд: " + chosenTrain.getTrainNumber());
    }

    public Invoice getInvoice(double price) {

        System.out.println("Счет выставлен на сумму: " + price);
        return new Invoice(price);
    }
}

class Train {
    private String trainNumber;
    private ArrayList<Station> stations;
    private HashMap<Station, Double> prices;

    public Train(String trainNumber) {
        this.trainNumber = trainNumber;
        this.stations = new ArrayList<>();
        this.prices = new HashMap<>();
    }

    public void addStation(Station station) {

        stations.add(station);
    }

    public void setPrice(Station station, double price) {

        prices.put(station, price);
    }

    public String getTrainNumber() {
```

```
        return trainNumber;
    }

    public double getPriceForStation(Station station) {

        return prices.get(station);
    }
}

class Station {
    private String name;
    private Date arrivalTime;
    private Date departureTime;

    public Station(String name, Date arrivalTime, Date departureTime) {
        this.name = name;
        this.arrivalTime = arrivalTime;
        this.departureTime = departureTime;
    }

    @Override
    public String toString() {
        return name;
    }
}

class Ticket {
    private Passenger passenger;
    private Train train;
    private Station destinationStation;
    private Date dateTime;
    private double price;

    public Ticket(Passenger passenger, Train train, Station destinationStation, Date
dateTime, double price) {
        this.passenger = passenger;
        this.train = train;
        this.destinationStation = destinationStation;
        this.dateTime = dateTime;
        this.price = price;
    }

    public void printTicket() {
        System.out.println("Детали билета:");
        System.out.println("Пассажир: " + passenger);
        System.out.println("Поезд: " + train.getTrainNumber());
        System.out.println("Станция назначения: " + destinationStation);
        System.out.println("Дата и время: " + dateTime);
        System.out.println("Стоимость: " + price);
    }
}

class Invoice {
    private double amount;

    public Invoice(double amount) {
        this.amount = amount;
    }

    public void printInvoice() {
```

```
        System.out.println("Счет на сумму: " + amount);
    }
}

class Administrator {
    public void addTrain(Train train) {

        System.out.println("Поезд добавлен: " + train.getTrainNumber());
    }

    public void setPrice(Train train, Station station, double price) {
        train.setPrice(station, price);
        System.out.println("Установлена цену для поезда " + train.getTrainNumber() +
" на станции " + station + ": " + price);
    }
}
 class Main {
    public static void main(String[] args) {
        Passenger passenger = new Passenger("Энакин Скайуокер", "ABC12345");
        Train train = new Train("Экспресс");
        Station station = new Station("ЗвездаСмерти", new Date(), new Date());

        train.addStation(station);

        Administrator admin = new Administrator();
        admin.setPrice(train, station, 50.0);

        passenger.createRequest("ЗвездаСмерти", new Date());

        ArrayList<Train> trains = new ArrayList<>();
        trains.add(train);

        passenger.chooseTrain(trains);

        double price = train.getPriceForStation(station);
        Invoice invoice = passenger.getInvoice(price);
        invoice.printInvoice();
        Ticket ticket = new Ticket(passenger, train, station, new Date(), price);
        ticket.printTicket();
    }
}
```

Результат:

```
Установлена цену для поезда Экспресс на станции ЗвездаСмерти: 50.0
Заявка создана на станцию: ЗвездаСмерти на Tue May 07 22:09:23 MSK 2024
Выбран поезд: Экспресс
Счет выставлен на сумму: 50.0
Счет на сумму: 50.0
Детали билета:
Пассажир: Пассажир: Энакин Скайуокер, Паспорт: ABC12345
Поезд: Экспресс
Станция назначения: ЗвездаСмерти
Дата и время: Tue May 07 22:09:23 MSK 2024
Стоимость: 50.0
```

Вывод: приобрел практические навыки в области объектно-ориентированного проектирования.