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

Лабораторная работа №4
По дисциплине: «СПП»

Выполнил:
студентка 3 курса
группы ПО-8
Гордейчук М.В.
Проверил:
Крощенко А.А.

2024

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования.

Вариант 7

Задание 1: реализовать указанный класс, включив в него вспомогательный внутренний класс или классы.

Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

Создать класс City (город) с внутренним классом, с помощью объектов которого можно хранить информацию о проспектах, улицах, площадях.

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.

Создать класс Страница, используя классы Строка, Слово.

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.

Система **Автобаза**. **Диспетчер** распределяет заявки на **Рейсы** между **Водителями** и назначает для этого **Автомобиль**. **Водитель** может сделать заявку на ремонт. **Диспетчер** может отстранить **Водителя** от работы. **Водитель** делает отметку о выполнении **Рейса** и состоянии **Автомобиля**.

Ход работы:

Задание 1:

Текст программы:

```
import java.util.ArrayList;

public class Zadanie1 {

   public static void main(String[] args) {

      City city = new City();
      city.AddPlace(1, "Pobediteley", "Gogolya", "Lenina");
      city.AddPlace(2, "Nezavisimosti", "Budennogo", "Svobody");
      city.show();
   }
```

```
}
class City {
    private class Place {
        int num;
                String avenue; //проспект
        String street;//улица
        String square;//площадь

        @Override
        public String toString() {
            return "Place " + num +
                                        "\nAvenu: " + avenue +
                    "\nStreet: " + street +
                    "\nSquare: " + square +"\n";
        }
    }
    ArrayList<Place> places = new ArrayList<Place>();
    public void AddPlace (int n, String AddAvenue, String AddStreet, String AddSquare) {
        Place place = new Place();
        place.num = n;
                place.avenue = AddAvenue;
        place.street = AddStreet;
        place.square = AddSquare;
        places.add(place);
    }
    public void show () {
                System.out.println("City:\n");
        for (Place place : places) {
            System.out.println(place.toString());
        }
    }
}
```

Результат:

Задание 2:

Текст программы:

```
import java.util.Vector;
public class Zadanie2 {
    public static void main(String[] args) {
        Word wordb1 = new Word("I");
        Word wordb2 = new Word("am");
        Word wordb3 = new Word("a");
        Word wordb4 = new Word("student");

        Word wordb5 = new Word("and");
        Word wordb6 = new Word("I");
        Word wordb7 = new Word("love");
        Word wordb8 = new Word("sleeping");

        Str str = new Str();
        str.AddWord(wordb1);
        str.AddWord(wordb2);
        str.AddWord(wordb3);
        str.AddWord(wordb4);

        Str Str2 = new Str();
        Str2.AddWord(wordb5);
        Str2.AddWord(wordb6);
        Str2.AddWord(wordb7);
        Str2.AddWord(wordb8);

        MyPage page = new MyPage();
```

```
        page.AddStr(str);
        page.AddStr(Str2);

        System.out.println(page.toString());
    }
}
class Word {
    private String Word;
    public Word (String c) {
        this.Word = c;
    }
    public String getStr() {
        return Word;
    }
}
class Str {
    private Vector<Word> Str = new Vector<Word>();
    void AddWord (Word word) {
        Str.add(word);
    }
    @Override
    public String toString() {
        StringBuilder str = new StringBuilder();
        for (Word wordb: Str) {
                        str.append(wordb.getStr());
            str.append(' ');
                    }
        return str.toString();
    }
}
class MyPage {
    private  Vector<Str> mypage = new Vector<Str>();
    void AddStr(Str str) {
        mypage.add(str);
    }
         @Override
    public String toString() {
```

```
        StringBuilder page = new StringBuilder();
        for (Str str: mypage) {
            page.append(str.toString());
            page.append('\n');
        }
        return page.toString();
    }

}
```

Результат:

Из слов были составлены две строки, а из строк – страница.

Задание 3:

Текст программы:

```
import java.util.HashMap;
import java.util.ArrayList;

public class Autobasa {
    public static void main(String[] args) {
        Race race1 = new Race(Race.RaceName.Race1);
        Race race2 = new Race(Race.RaceName.Race2);

        Car carstate1 = new Car(Car.States.GoodWork);
        Car carstate2 = new Car(Car.States.CarState);
            Car carstate3 = new Car(Car.States.Crash);

        dispatcher dispatcherCriteriy1 = new dispatcher("Vladimir", "Gordiy", carstate1);
        dispatcher dispatcherCriteriy2 = new dispatcher("Vladimir", "Gordiy", carstate2);
            dispatcher dispatcherCriteriy3 = new dispatcher("Vladimir", "Gordiy", carstate3);

        //для race1
        Driver driv1 = new Driver("Semen", "Sova");
        Driver driv2 = new Driver("Saveliy", "Gromov");

        driv1.Register(race1);
```

```
        driv2.Register(race1);

        driv1.PassCar(dispatcherCriteriy1.Rate(8), dispatcherCriteriy1.getState());
        driv1.PassCar(dispatcherCriteriy2.Rate(6), dispatcherCriteriy2.getState());
          driv1.PassCar(dispatcherCriteriy3.Rate(0), dispatcherCriteriy3.getState());

        driv2.PassCar(dispatcherCriteriy1.Rate(7), dispatcherCriteriy1.getState());
        driv2.PassCar(dispatcherCriteriy2.Rate(9), dispatcherCriteriy2.getState());
          driv2.PassCar(dispatcherCriteriy3.Rate(4), dispatcherCriteriy3.getState());

        race1.showRegisteredDrivers();
        race1.RaceOK();
        race1.showRecivedDrivers();
          race1.showSuspendedDrivers();

        //для race2
        Driver driv3 = new Driver("Alexandr", "Ivanov");
        Driver driv4 = new Driver("Petr", "Zarya");

        driv3.Register(race2);
        driv4.Register(race2);

        driv3.PassCar(dispatcherCriteriy1.Rate(4), dispatcherCriteriy1.getState());
        driv3.PassCar(dispatcherCriteriy2.Rate(5), dispatcherCriteriy2.getState());
          driv3.PassCar(dispatcherCriteriy3.Rate(0), dispatcherCriteriy3.getState());

        driv4.PassCar(dispatcherCriteriy1.Rate(7), dispatcherCriteriy1.getState());
        driv4.PassCar(dispatcherCriteriy2.Rate(9), dispatcherCriteriy2.getState());
          driv4.PassCar(dispatcherCriteriy3.Rate(0), dispatcherCriteriy3.getState());

        race2.showRegisteredDrivers();
        race2.RaceOK();
        race2.showRecivedDrivers();
          race2.showSuspendedDrivers();
    }
}
```

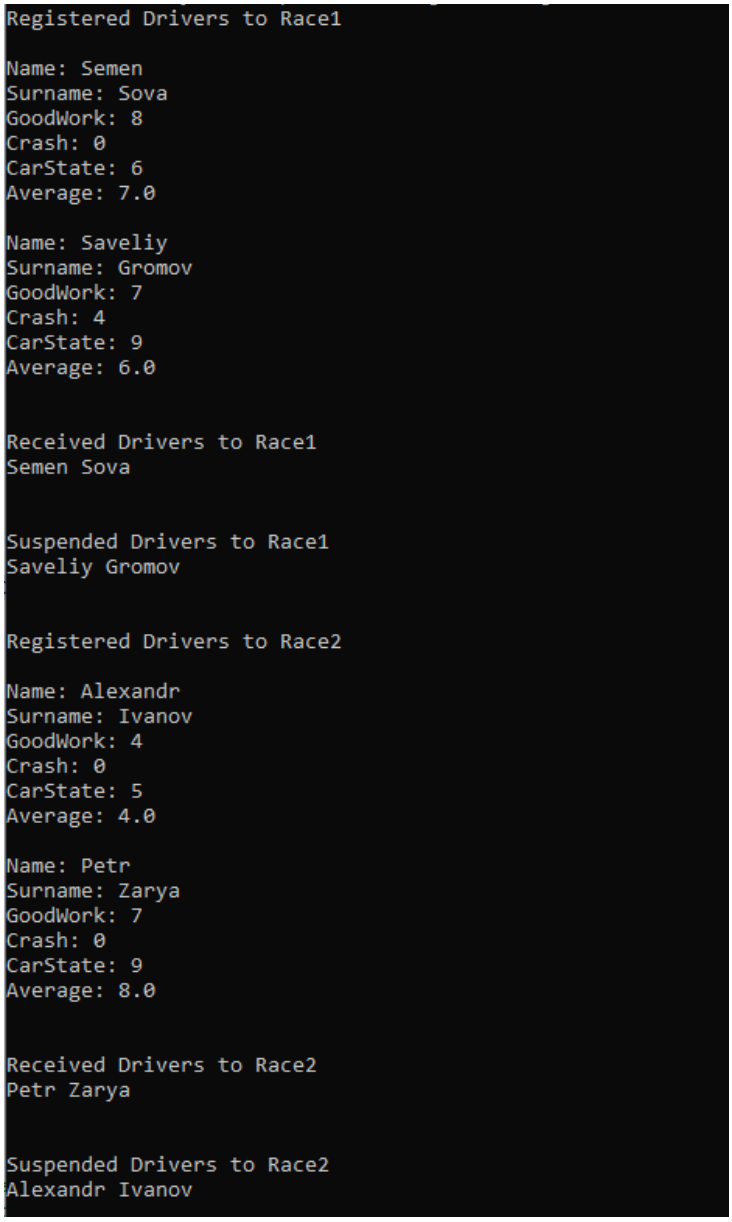
```
class Driver {
    private String name;
    private String surname;

    private Race race;
    private HashMap<Car.States, Mark> results = new HashMap<Car.States, Mark>();


    public Driver (String Dname, String Dsurname) {
        name = Dname;
        surname = Dsurname;
    }
    public void Register (Race rac) {
        rac.AddDriverToRace(this);
        race = rac;
    }

    public void PassCar(Mark _mark, Car.States _state) {
        results.put(_state, _mark);
    }


    public int getResults(Car.States _state) {
        return this.results.get(_state).getMark();
    }

    public double GetAverage () {
        return  (this.getResults(Car.States.GoodWork) + this.getResults(Car.States.CarState) -
this.getResults(Car.States.Crash))/2;
    }

    @Override
    public String toString() {
        return "\nName: " + name + '\n' +
                "Surname: " + surname + '\n' +
                "GoodWork: " + results.get(Car.States.GoodWork).getMark() + '\n' +
                                "Crash: " + results.get(Car.States.Crash).getMark() + '\n' +
```

```
            "CarState: " + results.get(Car.States.CarState).getMark() + '\n' +
            "Average: " + GetAverage();
    }

    public String toSmallString() {
        return name + ' ' + surname;
    }
}

class Race {
    public enum RaceName {
        Race1,
        Race2
    }
    private RaceName name;
    private ArrayList<Driver> RegisteredDrivers = new ArrayList<Driver>(); //получившие заявку на рейс
    private ArrayList<Driver> ReceivedDrivers = new ArrayList<Driver>();//кто хорошо отработал    private ArrayList<Driver> SuspendedDrivers = new ArrayList<Driver>(); //отстранённые

    public Race (RaceName Rname) {
        name = Rname;
    }

    public void AddDriverToRace (Driver driv) {
        RegisteredDrivers.add(driv);
    }

    public void RaceOK () {
        for (Driver driver:RegisteredDrivers) {
            if (driver.GetAverage() > 6) ReceivedDrivers.add(driver);
                        else SuspendedDrivers.add(driver);
        }

    }
```

```
    public void showRegisteredDrivers() {
        System.out.println("Registered Drivers to " + name);
        for (Driver driv: RegisteredDrivers)
            System.out.println(driv.toString());
        System.out.println('\n');
    }

    public void showRecivedDrivers () {
        System.out.println("Received Drivers to " + name);
        for (Driver driv: ReceivedDrivers)
            System.out.println(driv.toSmallString());
        System.out.println('\n');
    }

        public void showSuspendedDrivers () {
        System.out.println("Suspended Drivers to " + name);
        for (Driver driv: SuspendedDrivers)
            System.out.println(driv.toSmallString());
        System.out.println('\n');
    }

}

class dispatcher {
    private String name;
    private String surname;
    private Car.States state;
    public dispatcher (String _name, String _surname, Car _car) {
        name = _name;
        surname = _surname;
        state = _car.getState();
    }
    public Mark Rate(int _mark) {
        Mark mark = new Mark();
        mark.setMark(_mark);
        return  mark;
    }
```

```
    public Car.States getState() {
        return state;
    }
}

class Car {

    public enum States {
        GoodWork, CarState, Crash
    }
    private States state;
    public Car (States _state) {
        state = _state;
    }

    public States getState() {
        return state;
    }
}

class Mark {
    private int mark;

    public int getMark() {
        return mark;
    }

    public void setMark(int _mark) {
        mark = _mark;
    }
}
```

Результат:



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