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

Лабораторная работа №4

По дисциплине: «Современные платформы программирования»

Выполнила:
Студентка 3 курса
Группы ПО-6
Юсковец М.А.
Проверил:
Монтик Н.С.

Брест, 2022

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

Ход работы:

Вариант 25

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.

Текст программы:

Main.java

```
import pack.City;

public class Main {
    public static void main(String[] args) {
        City c = new City();
        c.addData("av", "st", "sq");
        c.show();
    }
}
```

City.java

```
package pack;

import java.util.ArrayList;

public class City {
    private class Data {
        public String avenue;
        public String street;
        public String square;

        Data(String avenue, String street, String square){
            this.avenue = avenue;
            this.street = street;
            this.square = square;
        }

        public String dataToString() {
            return "Avenue: "+this.avenue
                    + "\nStreet: "+this.street
                    + "\nSquare: "+this.square
                    + "\n-------------------------";
        }
    }
    ArrayList<Data> dataList = new ArrayList<>();
```

```
    public void addData(String avenue, String street, String square) {
        Data d = new Data(avenue, street, square);
        dataList.add(d);
    }

    public void show() {
        for(Data d : dataList) {
            System.out.println(d.dataToString());
        }
    }
}
```

Результат программы:

```
Avenue: av
Street: st
Square: sq
------------------------

Process finished with exit code 0
```

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 Строка, используя классы Слово, Символ.

Текст программы:

```
import java.util.Vector;

public class Main {
    public static void main(String[] args) {
        Symbol symb1 = new Symbol('H');
        Symbol symb2 = new Symbol('e');
        Symbol symb3 = new Symbol('l');
        Symbol symb4 = new Symbol('l');
        Symbol symb5 = new Symbol('o');

        Symbol symb6 = new Symbol('t');
        Symbol symb7 = new Symbol('o');

        Symbol symb8 = new Symbol('m');
        Symbol symb9 = new Symbol('e');

        Word w1 = new Word();
        Word w2 = new Word();
        Word w3 = new Word();
```

```
        w1.addSymbol(symb1);
        w1.addSymbol(symb2);
        w1.addSymbol(symb3);
        w1.addSymbol(symb4);
        w1.addSymbol(symb5);

        w2.addSymbol(symb6);
        w2.addSymbol(symb7);

        w3.addSymbol(symb8);
        w3.addSymbol(symb9);

        MyString str = new MyString();
        str.addWord(w1);
        str.addWord(w2);
        str.addWord(w3);

        System.out.println(str.toString());
    }
}

class Symbol {
    private char s;

    public Symbol(char s) {
        this.s = s;
    }

    public char getSymbol() {
        return s;
    }

}

class Word {
    private Vector<Symbol> word = new Vector<>();

    public void addSymbol(Symbol c) {
        word.add(c);
    }

    @Override
    public String toString() {
        StringBuilder str = new StringBuilder();
        for (Symbol symb: word)
            str.append(symb.getSymbol());
        return str.toString();

    }
}


class MyString {
    private Vector<Word> string = new Vector<>();

    public void addWord(Word w) {
        string.add(w);
    }

    @Override
    public String toString() {
        StringBuilder str = new StringBuilder();
        for (Word w : string) {
            str.append(w.toString());
```

```
            str.append(' ');
        }
        return str.toString();
    }
}
```

Результат программы:

```
Hello to me


Process finished with exit code 0
```

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 Больница. Пациенту назначается лечащий Врач. Врач может сделать назначение Пациенту (процедуры, лекарства, операции). Медсестра или другой Врач выполняют назначение. Пациент может быть выписан из Больницы по окончании лечения, при нарушении режима или иных обстоятельствах.

Текст программы:

Main.java

```
package pack;

public class Main {
    public static void main(String[] args) {
        Hospital hosp = new Hospital("Hospital №1");
        Doctor doc1 = new Doctor("Name1", "Surname1", "Surgeon");
        Nurse nurse1 = new Nurse("Name2", "Surname2");
        Patient patient1 = new Patient("Name3", "Surname3", doc1);

        hosp.addDoctor(doc1);
        hosp.showListDoctors();

        doc1.addAppointments("medicines", patient1, nurse1);
        System.out.println(patient1.getAppointments());
        System.out.println(patient1.getDischarge());
        System.out.println(nurse1.getAppointmentsDone());

        nurse1.setAppointmentsDone(patient1);
        System.out.println(nurse1.getAppointmentsDone());
        System.out.println(patient1.getDischarge());
    }
}
```

Hospital.java

```
package pack;
import java.util.ArrayList;

public class Hospital {
    private String name;
    private ArrayList<Doctor> doctors = new ArrayList<>();

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

    public String getName() {
        return this.name;
    }

    public void showListDoctors() {
        System.out.println("Doctors:");
        for (Doctor doc : doctors) {
            System.out.println("Name: "+doc.getName()
                               +"\nSurname: "+doc.getSurname()
                               +"\nSpeciality: "+doc.getSpeciality()
                               +"\n----------------------");
        }
    }

    public void addDoctor(Doctor d) {
        doctors.add(d);
    }
}
```

Doctor.java

```
package pack;

public class Doctor {
    private String name;
    private String surname;
    private String speciality;

    public Doctor(String name, String surname, String speciality) {
        this.name = name;
        this.surname = surname;
        this.speciality = speciality;
    }

    public String getName() {
        return this.name;
    }

    public String getSurname() {
        return this.surname;
    }

    public String getSpeciality() {
        return this.speciality;
    }

    public void addAppointments(String appointments, Patient p, Nurse n) {
```

```
        p.setAppointments(appointments);
        p.setDischarge(false);
        n.setAppointments(false);
    }
}
```

Nurse.java

```
package pack;

public class Nurse {
    private String name;
    private String surname;
    private Boolean appointmentsDone;

    public Nurse(String name, String surname) {
        this.name = name;
        this.surname = surname;
    }

    public String getName() {
        return this.name;
    }

    public String getSurname() {
        return this.surname;
    }

    public Boolean getAppointmentsDone() {
        return this.appointmentsDone;
    }

    public void setAppointments(Boolean res) {
        this.appointmentsDone = res;
    }

    public void setAppointmentsDone(Patient p) {
        this.appointmentsDone = true;
        p.setDischarge(true);
    }
}
```

Patient.java

```
package pack;

public class Patient {
    private String name;
    private String surname;
    private Doctor doc;
    private String appointments;
    private Boolean discharge;

    public Patient(String name, String surname, Doctor doc) {
        this.name = name;
        this.surname = surname;
        this.doc = doc;
    }

    public String getName() {
```

```
        return this.name+this.surname;
    }

    public Doctor getDoctor() {
        return this.doc;
    }

    public String getAppointments() {
        return this.appointments;
    }

    public void setAppointments(String appointments) {
        this.appointments = appointments;
    }

    public Boolean getDischarge() {
        return this.discharge;
    }

    public void setDischarge(Boolean d) {
        this.discharge = d;
    }
}
```

Результат программы:

```
Doctors:
Name: Name1
Surname: Surname1
Speciality: Surgeon
--------------------
medicines
false
false
true
true

Process finished with exit code 0
```

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