Министерство образования Республики Беларусь

Учреждение образования «Брестский государственный технический университет»

Кафедра ИИТ

Отчёт по лабораторной работе № 4

По дисциплине «Современные платформы программирования»

Выполнил:
студент 3-го курса
группы ПО-9(2)
Николайчик Н.С.
Проверил:
Крощенко А. А.

Брест, 2024

Вариант 3

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.

Создать класс Account (счет) с внутренним классом, с помощью объектов которого можно хранить информацию обо всех операциях со счетом (снятие, платежи, поступления).

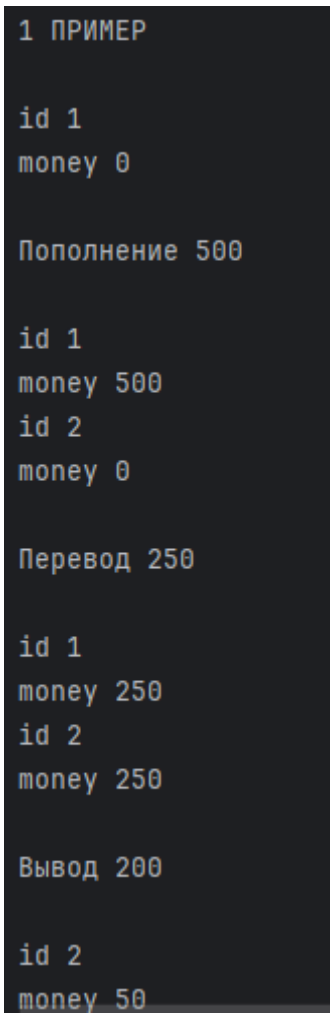
```
public class Account {
    final static short in_type=0,out_type=1,to_type=2;
    int id;
    int money;
    ArrayList<History> history;

    Account(){
        this.history=new ArrayList<History>();
    }

    public int getId() {
        return id;
    }

    public void refresh(){
        System.out.println("id " + this.id);
        System.out.println("money " +this.money);

    }
    public void output_history(){
        for(int i=0;i<this.history.size();i++){

System.out.println("////////////////////////////////////////////////////////");
            System.out.println("История пользователя "+ this.id);

System.out.println("////////////////////////////////////////////////////////");
            if(this.history.get(i).type == in_type)
System.out.println("Поступление средств");
            if(this.history.get(i).type == out_type)
System.out.println("Вывод средств");
            if(this.history.get(i).type == to_type)
System.out.println("Перечисление средств");
            System.out.println("дата "+this.history.get(i).date.toString());
            System.out.println("величина "+this.history.get(i).size);
            System.out.println("от "+this.history.get(i).from_id);
            System.out.println("к "+this.history.get(i).to_id);
            System.out.println();
            System.out.println();
        }
    }
}
```

```
public class History {
    final static short in_type=0,out_type=1,to_type=2;
    public Date date;
    public short type;
    public int to_id,from_id;
    public long size;
    History(int size,short type,int from_id,int to_id){
        this.date = new Date();
        this.type = type;
        this.size = size;
        this.to_id = to_id;
        this.from_id = from_id;
    }
}
```

```
public class Manager {
    static ArrayList<Account> stored_accounts = null;
    static void addAccount(Account account){
        account.id=0;
        if(stored_accounts==null){
            stored_accounts=new ArrayList<Account>();
        }
        stored_accounts.add(account);
        boolean flag=true;
        for (int i=1;account.id==0;i++){
            flag=true;
            for(int j=0;j<stored_accounts.size();j++){
                if(stored_accounts.get(j).id == i)
                {
                    flag=false;
                }
            }
            if(flag) {
                account.id = i;
            }
        }
        account.money=0;
    }

    static public void Operation_in(Account myAccount, int size){
        if(stored_accounts==null){
            stored_accounts=new ArrayList<Account>();
        }
        myAccount.money+=size;
        myAccount.history.add(new
History(size,Account.in_type,myAccount.id,myAccount.id));
    }
    static public boolean Operation_out(Account myAccount,int size){
        if(stored_accounts==null){
            stored_accounts=new ArrayList<Account>();
        }
        if(myAccount.money>size) {
            myAccount.money -= size;
            myAccount.history.add(new History(size, Account.out_type,
myAccount.id, myAccount.id));
            return true;
        }
        else {
            System.out.println("Нет денюх");
            return false;
```

```
        }
    }
    static public boolean Operation_to(Account myAccount,int size, int
to_id){
        if(stored_accounts==null){
            stored_accounts=new ArrayList<Account>();
        }
        Account temp=null;
        for(int i=0; i<stored_accounts.size();i++){
            if(stored_accounts.get(i).id==to_id)
                temp = stored_accounts.get(i);
        }
        if(temp==null){
            System.out.println("Нет такого пользователя");
            return false;
        }
        if(myAccount.money>size) {
            myAccount.money -= size;
            temp.money += size;
            myAccount.history.add(new History(size, Account.to_type,
myAccount.id, temp.id));
            temp.history.add(new History(size, Account.in_type, myAccount.id,
temp.id));
            return true;
        }
        else {
            System.out.println("Нет денюх");
            return false;
        }
    }
}
```

Работа:



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 класс Абзац.

```
public class Paragraph {
    String storage;
    Paragraph(String storage){
        this.storage= storage;
```

```
    }
    public void out(){
        System.out.println(this.storage);
        System.out.println();
    }
}
```

```
public class Page {
    ArrayList<Paragraph> list;
    public void out(){
        for(int i=0;i<this.list.size();i++){
            this.list.get(i).out();
        }
    }
    Page(){
        this.list = new ArrayList<Paragraph>();
        this.list.add(new Paragraph("afqasgtregghfew dfgtrw rtew re trewy
trew\n erytrwhtrdrew"));
        this.list.add(new Paragraph("afqtrewy trew\n erytrwhtrdrew"));
        this.list.add(new Paragraph("afqasgtrевнпраовопапавоав trew\n
еруваporвadrew"));
    }
    Paragraph getParagraph(int index){
        return this.list.get(index);
    }
    void delParagraph(int index){
        this.list.remove(index);
    }
    void addParagraph(int index, Paragraph para){
        this.list.add(index,para);
    }
    void addParagraphEnd(int index, Paragraph para){
        this.list.add(para);
    }
}
```

Работа:

```
2 ПРИМЕР

afqasgtregghfew dfgtrw rtew re trewy trew
 erytrwhtrdrew

afqtrewy trew
 erytrwhtrdrew

afqasgtrевнпраовопапавоав trew
 ерувапорвadrew


Удаление и добавление параграфа


АВОВА

afqasgtregghfew dfgtrw rtew re trewy trew
 erytrwhtrdrew

afqasgtrевнпраовопапавоав trew
 ерувапорвadrew
```

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

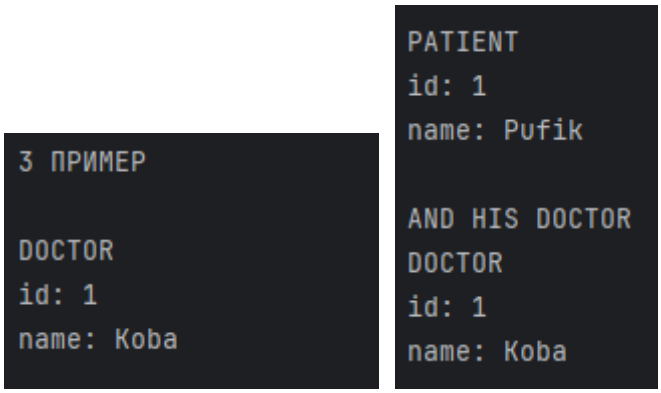
```
import java.util.ArrayList;

public class Hospital {
    public ArrayList<Doctor> stored_doctors = null;
    public ArrayList<Patient> stored_patients = null;
    Hospital(){
        this.stored_doctors.add(new Doctor("Koba",this));
        this.stored_doctors.add(new Doctor("Zloba",this));
        this.stored_doctors.add(new Doctor("Moba",this));
        this.stored_doctors.add(new Doctor("Zaznoba",this));
        this.stored_patients.add(new Patient("Pufik",this,
                stored_doctors.get(0)));
        this.stored_patients.add(new Patient("Aufik",this,
                stored_doctors.get(1)));
        this.stored_patients.add(new Patient("Gulfik",this,
                stored_doctors.get(2)));
        this.stored_patients.add(new Patient("Rufik",this,
                stored_doctors.get(3)));
    }
    public void out_doc(){
        for (int i = 0; i < this.stored_doctors.size(); i++) {
            this.stored_doctors.get(i).out();
        }
    }
    public void out_pat(){
        for (int i = 0; i < this.stored_patients.size(); i++) {
            this.stored_patients.get(i).out();
        }
    }
}
```

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

```
public class Doctor {
    Hospital hospital;
    String name;
    int id;
    Doctor(String name, Hospital hospital){
        this.name=name;
        this.hospital=hospital;
        this.id=0;
        hospital.stored_doctors.add(this);
        if(hospital.stored_doctors==null){
            hospital.stored_doctors= new ArrayList<Doctor>();
        }
        boolean flag=true;
        for (int i=1;this.id==0;i++){
            flag=true;
            for(int j=0;j<hospital.stored_doctors.size();j++){
                if(hospital.stored_doctors.get(j).id == i)
                {
                    flag=false;
                }
            }
            if(flag) {
                this.id = i;
            }
        }
    }

    void Set_note(String note, Patient patient){
        if(this.id== patient.doctor.id) {
            patient.note = note;
            patient.type = patient.wait_type;
            System.out.println("Назначено");
        }
        else System.out.println("Невозможно назначить");
    }
    void Cure(Patient patient){
        for(int j=0;j<patient.hospital.stored_patients.size();j++){
            if(patient.hospital.stored_patients.get(j).id == patient.id)
            {
                patient.hospital.stored_patients.remove(j);
            }
        }

    }
    void out(){
        System.out.println("DOCTOR");
        System.out.println("id: "+this.id);
        System.out.println("name: "+this.name);
        System.out.println();
    }
}
```

```
public class Patient {
    short in_type=0,out_type=1,wait_type=2;
    Hospital hospital;
    public String name;
    int id;
    int type;
    public Doctor doctor;
    String note=null;
    Patient(String name, Hospital hospital, Doctor doctor){
        this.name=name;
        this.hospital=hospital;
        this.doctor=doctor;
        this.id=0;
        hospital.stored_patients.add(this);
        if(hospital.stored_patients==null){
            hospital.stored_patients= new ArrayList<Patient>();
        }
        boolean flag=true;
        for (int i=1;this.id==0;i++){
            flag=true;
            for(int j=0;j<hospital.stored_patients.size();j++){
                if(hospital.stored_patients.get(j).id == i)
                {
                    flag=false;
                }
            }
            if(flag) {
                this.id = i;
            }
        }
        this.type=in_type;

    }
    void out(){
        System.out.println("PATIENT");
        System.out.println("id: "+this.id);
        System.out.println("name: "+this.name);
        System.out.println();
        System.out.println("AND HIS DOCTOR");
        this.doctor.out();
    }
}
```

Работа:



Вывод: я попрактиковался в объектно-ориентированном проектировании.