Министерство образования Республики Беларусь

Учреждение образования «Брестский государственный технический университет»

Кафедра ИИТ

Отчёт по лабораторной работе № 5

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 абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

interface Сотрудник ← class Инженер ← class Руководитель.

Код:

```
public interface Employee {
    void interacted(String type, int cost);
    void Interaction(Employee employee, int cost);
    void out();
}
```

```
public class Engineer implements Employee{
    int age = 0;
    String name = null;
    int money = 0;
    int mood = 0;
    String type="engineer";
    @Override
    public void interacted(String type, int cost){
        if(type.equals("director")){
            this.money+=cost/4;
            this.mood-=cost/4;
        }
        else{
            this.money+=cost*3/4;
            this.mood+=cost/2;
        }
    }
    @Override
    public void Interaction(Employee employee, int cost){
        employee.interacted(type,cost);
        this.money+=cost/2;
        this.mood+=cost/4;
    }
    public void out(){
        System.out.println("age: " + age);
        System.out.println("name: " +name);
        System.out.println("money: " +money);
        System.out.println("mood: " +mood);
        System.out.println();
    }
    Engineer(int age,String name,int money,int mood){
        this.mood=mood;
        this.age=age;
        this.money=money;
        this.name=name;
    }
}
```

```java
public class Director implements Employee{
    int age = 0;
    String name = null;
    int money = 0;
    int mood = 0;
    String type="director";
    Director(int age,String name,int money,int mood){
        this.mood=mood;
        this.age=age;
        this.money=money;
        this.name=name;
    }
    @Override
    public void interacted(String type, int cost){
        if(type.equals("director")){
            this.money+=cost/4;
            this.mood-=cost/4;
        }
        else{
            this.money+=cost*3/4;
            this.mood+=cost/2;
        }
    }
    @Override
    public void Interaction(Employee employee, int cost){
        employee.interacted(type,cost);
        this.money+=cost*3/4;
        this.mood+=cost/2;
    }
    public void out(){
        System.out.println("age: " + age);
        System.out.println("name: " +name);
        System.out.println("money: " +money);
        System.out.println("mood: " +mood);
        System.out.println();
    }
}
```

```java
public class Ex_1 {
    //когда рабочие трудятся вместе, они получают большую долю
    //и соответственно поовышают настроение, а если приходит начальник
    //то он забирает больше денег и отнимает настроение трудящегося
    Ex_1() {
        Employee Factory[] = new Employee[3];
        //Director dir=new Director(19,"Milosh",300,41);
        Factory[0]=new Director(19,"Milosh",300,41);;
        Factory[1]=new Engineer(44,"Ricardo",100,41);
        Factory[2]=new Engineer(33,"Mykola",150,41);
        for (int i = 0; i < 3; i++) {
            Factory[i].out();
        }
        Factory[0].Interaction(Factory[1],50);
        Factory[0].Interaction(Factory[2],50);
        out(Factory);
        Factory[1].Interaction(Factory[2],50);
        out(Factory);
    }
    void out(Employee Factory[]){
        for (int i = 0; i < 3; i++) {
            Factory[i].out();
        }
        System.out.println();
```

```
        System.out.println();
        System.out.println();

    }
}
```

Работа:

```
age: 19
name: Milosh
money: 300
mood: 41

age: 44
name: Ricardo
money: 100
mood: 41

age: 33
name: Mykola
money: 150
mood: 41

age: 19
name: Milosh
money: 374
mood: 91

age: 44
name: Ricardo
money: 112
mood: 29
```

Задание 2

В следующих заданиях требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

Создать суперкласс Музыкальный инструмент и классы Ударный, Струнный, Духовой. Создать массив объектов Оркестр. Осуществить вывод состава оркестра.

Код:

```
abstract public class Instrument {
    protected int cost;
    protected int year;
    protected String brand;
    protected String type;
    Instrument(int cost,int year, String brand) throws Exception {
        this.cost=cost;
        this.year=year;
        this.brand=brand;
        type=null;
    }
    public void out(){
        System.out.println("cost: " + cost);
        System.out.println("year: " +year);
        System.out.println("brand: " +brand);
        if(type!=null)
            System.out.println("type: " +type);
        System.out.println();
        System.out.println();
    }
}
```

```
public class Wind_Instrument extends Instrument{
    Wind_Instrument(int cost, int year, String brand) throws Exception {
        super(cost, year, brand);
        this.type = new String("Wind");
    }
    public void dzyn(){
        new PlayMp3("./musicbossorg_Shitty_flute_-
_Harry_Potter_Theme_64368916.mp3");
    }
}
```

```
public class String_Instrument extends Instrument{
    String_Instrument(int cost, int year, String brand) throws Exception {
        super(cost, year, brand);
        this.type = new String("String");
    }
    public void trun(){
        new PlayMp3("./jg-032316-sfx-acoustic-guitar-a-minor-chord.mp3");
    }
}
```

```
public class Percussion_Instrument extends Instrument{
    Percussion_Instrument(int cost,int year, String brand) throws Exception {
        super( cost, year, brand);
        this.type=new String("Percussion");
    }
    public void boom(){
        new PlayMp3("./barabannaya-drob-otlichnaya-shutka.mp3");
    }
}
```

Работа:

```
cost: 200
year: 2000
brand: Boomba
type: Percussion


cost: 3200
year: 2020
brand: Strings
type: String


cost: 2050
year: 2003
brand: O, i am sorry
type: Wind
```

Задание 3 В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Здесь я добавил предка врачам и больным. Этим предком стал человек.

Код:

```
abstract public class Mensch {
    protected String name;
    protected int age;
    Mensch(String name,int age){
        this.name=name;
        this.age=age;
    }
}
```

```
public class Patient extends Mensch{
    short in_type=0,out_type=1,wait_type=2;
    Hospital hospital;
    int id;
    int type;
    public Doctor doctor;
    String note=null;
    Patient(String name, Hospital hospital, Doctor doctor,int age){
        super(name,age);
        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;

    }
    public String toString(){
        return "PATIENT\n" +
                "id: "+this.id + '\n' +
                "name: "+this.name + "\n\n";
    }
}



public class Doctor extends Mensch {
    Hospital hospital;
    int id;
    Doctor(String name, Hospital hospital,int age){
        super(name,age);
        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);
            }
        }

    }
    public String toString(){
        return "DOCTOR\n" +
                "id: "+this.id + '\n' +
                "name: "+this.name + "\n\n";
    }
}




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

    void Set_ukol(Patient patient){
        System.out.println(patient.toString());
            System.out.println("УКОЛОТ");
    }
    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);
            }
        }

    }
```

```
    public String toString(){
        return "NURSE\n" +
        "id: "+this.id + '\n' +
        "name: "+this.name + "\n\n";
    }
}
```

Вывод: я попрактиковался в объектно-ориентированном проектировании.