Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине «СПП»

за пятый семестр

**Выполнил:**

Студент 3 курса

Группы ПО-5

Брич М.Н.

**Проверил:**

Крощенко А.А.

**Брест 2021**

Вариант 3

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

Ход работы:

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

interface Сотрудник ← class Инженер ← class Руководитель.

Код алгоритма:

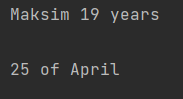
public class Lab5\_1  
{  
  
 public static void main(String[] args)  
 {  
 director d = new director("Maksim", 19, "25 of April");  
 d.show();  
 d.payday();  
 }  
  
}

public class director extends engineer  
{  
 String name;  
 int age;  
 @Override  
 public void show()  
 {  
 System.*out*.println(name+" "+age+" years\n");  
 }  
  
 public director(String \_name, int \_age, String \_salaryDay)  
 {  
 super(\_salaryDay);  
 name = \_name;  
 age = \_age;  
 }  
}

public interface employee  
{  
 public void show();  
 public void payday();  
}

public class engineer implements employee  
{  
 String salaryDay;  
  
 public engineer(String \_salaryDay)  
 {  
 salaryDay = \_salaryDay;  
 }  
  
 public void show()  
 {  
  
 }  
  
 public void payday()  
 {  
 System.*out*.println(salaryDay);  
 }  
}

Результат выполнения программы:



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.

Код алгоритма:

import java.util.\*;  
public class Lab5\_2  
{  
  
 public static void main(String[] args)  
 {  
 Vector<musicInst> insts = new Vector<>();  
 insts.add(new percussion("drums", "Maksim"));  
 insts.add(new strings("guitar","Roma"));  
 insts.add(new wind("trumpet", "Nikita"));  
  
 for (musicInst i: insts)  
 {  
 i.show();  
 }  
 }  
  
}

public class musicInst  
{  
 private String instrument;  
  
 public musicInst(String \_inst)  
 {  
 instrument = \_inst;  
 }  
  
 public String getInst()  
 {  
 return instrument;  
 }  
  
 public void show() {  
 System.*out*.println();  
 }  
}

abstract class musicInst  
{  
 private String inst rument;  
  
 public musicInst(String \_inst)  
 {  
 instrument = \_inst;  
 }  
  
 public String getInst()  
 {  
 return instrument;  
 }  
  
 public abstract void show();  
}

public class percussion extends musicInst  
{  
 String owner;  
  
 public percussion(String \_inst, String \_owner)  
 {  
 super(\_inst);  
 owner = \_owner;  
 }  
  
 public void show()  
 {  
 System.*out*.println(owner+" plays on "+super.getInst());  
 }  
}

public class strings extends musicInst  
{  
 String owner;  
 public strings(String \_inst, String \_owner)  
 {  
 super(\_inst);  
 owner = \_owner;  
 }  
  
 public void show()  
 {  
 System.*out*.println(owner+" plays on "+super.getInst()+" for 2 years");  
 }  
}

public class wind extends musicInst  
{  
 String owner;  
 public wind(String \_inst, String \_owner)  
 {  
 super(\_inst);  
 owner = \_owner;  
 }  
  
 public void show()  
 {  
 System.*out*.println(owner+" plays on "+super.getInst()+" for 3 years");  
 }  
}

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.

3) 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.

Код программы:

Main.java

import entities.Doctor;

import entities.MainDoctor;

import entities.Patient;

import entities.Purpose;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

MainDoctor mainDoctor = new MainDoctor(23, "Доктор", "Айболит");

Doctor doctor = new Doctor(35, "Крокодил", "Гена", 1);

Doctor nurse = new Doctor(37, "Чебу", "рашка", 0);

Patient patient = new Patient(50, "Иван", "Бывалый");

Purpose purpose1 = new Purpose("Анальгин", 1);

purpose1.setCountOfDay(3);

Purpose purpose2 = new Purpose("Массаж спины", 0);

purpose2.setCountOfDay(1);

Purpose purpose3 = new Purpose("Удаление аденоидов", 2);

ArrayList<Purpose> purposes = new ArrayList<>();

purposes.add(purpose1);

purposes.add(purpose2);

purposes.add(purpose3);

mainDoctor.makePurposes(patient, purposes);

patient.print();

for(int i=0; i<5; i++)

System.out.println();

nurse.executePurpose(patient, purpose1);

doctor.executePurpose(patient, purpose3);

nurse.executePurpose(patient, purpose1);

nurse.executePurpose(patient, purpose2);

nurse.executePurpose(patient, purpose1);

mainDoctor.dischargePatient(patient, 1);

patient.print();

}

}

Doctor.java

package entities;

import interfaces.IDoctor;

public class Doctor extends Person implements IDoctor {

private int type; //0 - медсестра, 1 - доктор

public Doctor(int id, String name, String surname, int type) {

super(id, name, surname);

this.type = type;

}

public int getType() {

return type;

}

public void setType(int type) {

this.type = type;

}

public void executePurpose(Patient patient, Purpose purpose){

Purpose donePurpose = purpose.clone();

donePurpose.setDoctor(this);

donePurpose.updateTime();

donePurpose.setDone(true);

patient.addToHistory(donePurpose);

}

}

MainDoctor.java

package entities;

import interfaces.IMainDoctor;

import java.util.ArrayList;

public class MainDoctor extends Doctor implements IMainDoctor {

public MainDoctor(int id, String name, String surname) {

super(id, name, surname, 1);

}

public void makePurposes(Patient patient, ArrayList<Purpose> purposes) {

for (Purpose purpose : purposes) {

Purpose purpose1 = purpose.clone();

purpose1.updateTime();

purpose1.setDoctor(this);

patient.addToHistory(purpose1);

}

}

public void dischargePatient(Patient patient, int reason){

patient.setStatus(reason);

}

}

Patient.java

package entities;

import java.util.ArrayList;

public class Patient extends Person {

private int status; // 0 - лечится, 1 - выписан по окончанию лечения, 2 - выписан из-за нарушения режима, 3 - выписан при иных обстоятельствах

private final ArrayList<Purpose> history;

public Patient(int id, String name, String surname) {

super(id, name, surname);

status=0;

history = new ArrayList<>();

}

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

public void addToHistory(Purpose purpose){

history.add(purpose);

}

public void print(){

System.out.print(super.toString());

System.out.print(", статус: ");

if(status==0)

System.out.println("лечится");

else if(status==1)

System.out.println("выписан по окончанию лечения");

else if(status==2)

System.out.println("выписан из-за нарушения режима");

else if(status==3)

System.out.println("выписан при иных обстоятельствах");

System.out.println("История действий:");

for (Purpose purpose : history) System.out.println(purpose.toString());

}

}

Person.java

package entities;

public class Person {

private int id;

private String name;

private String surname;

public Person(int id, String name, String surname) {

this.id = id;

this.name = name;

this.surname = surname;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String toString(){

return "id: "+id+", name: "+name+", surname: "+surname;

}

}

Purpose.java

package entities;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class Purpose {

private String name;

private int type; //0 - процедура, 1 - лекарство, 2 - операция

private LocalDateTime time;

private int countOfDay;

private Doctor doctor;

private boolean isDone = false;

public Purpose(String name, int type) {

this.name = name;

this.type = type;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getType() {

return type;

}

public void setType(int type) {

this.type = type;

}

public LocalDateTime getTime() {

return time;

}

public void setTime(LocalDateTime time) {

this.time = time;

}

public int getCountOfDay() {

return countOfDay;

}

public void setCountOfDay(int countOfDay) {

this.countOfDay = countOfDay;

}

public Doctor getDoctor() {

return doctor;

}

public void setDoctor(Doctor doctor) {

this.doctor = doctor;

}

public boolean isDone() {

return isDone;

}

public void setDone(boolean done) {

isDone = done;

}

public void updateTime(){

time = LocalDateTime.now();

}

public String toString(){

StringBuilder builder = new StringBuilder();

builder.append(time.format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss")));

builder.append(" пациенту");

if(!isDone)

builder.append(" дано");

else

builder.append(" проведено");

builder.append(" назначение, название: ").append(name).append(", тип: ");

if(type==0)

builder.append("процедура");

else if(type==1)

builder.append("лекарство");

else

builder.append("операция");

builder.append(", доктор: "+doctor.getName()+" "+doctor.getSurname());

if(type!=2 && !isDone)

builder.append(", число в день: "+countOfDay);

return builder.toString();

}

public Purpose clone(){

Purpose purpose = new Purpose(this.name, this.type);

purpose.doctor = doctor;

purpose.time = time;

purpose.countOfDay = countOfDay;

purpose.isDone = isDone;

return purpose;

}

}

IDoctor.java

package interfaces;

import entities.Patient;

import entities.Purpose;

public interface IDoctor {

void executePurpose(Patient patient, Purpose purpose);

}

IMainDoctor.java

package interfaces;

import entities.Patient;

import entities.Purpose;

import java.util.ArrayList;

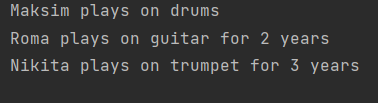
public interface IMainDoctor {

void makePurposes(Patient patient, ArrayList<Purpose> purposes);

void dischargePatient(Patient patient, int reason);

}

Результат выполнения программы:



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