МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №5

По дисциплине «Современные платформы программирования»
Специальность ПО-8

Выполнил:
Липовик И.С.
студент группы ПО-8
Проверил:
ст. преп. кафедры ИИТ,
«__»_____2024 г.

Брест 2024

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

Вариант 15

Задание 1. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

interface Abiturient ← abstract class Student ← class Student Of Faculty.

Код программы

Abiturient.java

```
public interface Abiturient {
    String getFirstName();
    String getLastName();
    void setFirstName(String FirstName);
    void setLastName(String LastName);
    void printInfo();
}
```

Student.java

```
public abstract class Student implements Abiturient {
    protected String firstName;
    protected String lastName;
    Student(String _firstName, String _lastName){
        firstName=_firstName;
        lastName=_lastName;
    }
    @Override
    public String getFirstName(){
        return firstName;
    }
    @Override
    public String getLastName(){
        return lastName;
    }
    @Override
    public void setFirstName(String FirstName){
        firstName=FirstName;
    }
    @Override
    public void setLastName(String LastName) {
        lastName=LastName;
    }

    @Override
    public void printInfo() {
        System.out.println("fisrt name: "+firstName+"\nlast name:
"+lastName);
    }
}
```

StudentOfFacuty.java

```
public class StudentOfFaculty extends Student{
    String faculty;
    StudentOfFaculty(String _firstName, String _lastName, String _faculty) {
        super(_firstName, _lastName);
        faculty=_faculty;
    }
    public String getFaculty(){
        return faculty;
    }
    public void setFaculty(String _faculty){
        faculty=_faculty;
```

```
    }
    @Override
    public void printInfo(){
        System.out.println("faculty: "+faculty+"\nfisrt name:
"+firstName+"\nlast name: "+lastName);
    }
}
```

task1.java

```
public class task1 {
    public static void main(String[] args) {
        StudentOfFaculty student = new StudentOfFaculty("Ivan", "Ivanov",
"Economic");
        student.printInfo();
    }
}
```

Пример

```
D:\СПП\lab5\out\production\lab5>java task1
faculty: Economic
fisrt name: Ivan
last name: Ivanov
```

Задание 2. Создать базовый класс Садовое дерево и производные классы Яблоня, Вишня, Груша и другие. С помощью конструктора автоматически становить номер каждого дерева. Принять решение о пересадке каждого дерева в зависимости от возраста и плодоношения.

Код программы

Tree.java

```
abstract class Tree {
    static int counter=1;
    protected int number;
    protected int age;
    protected boolean fruiting;
    Tree(int _age, boolean _fruiting){
        number=counter++;
        age=_age;
        fruiting=_fruiting;
    }
    public int getNumber() {
        return number;
    }

    public int getAge() {
        return age;
    }

    public boolean isFruiting() {
        return fruiting;
    }
    public void setNumber(int number) {
        this.number = number;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public void setFruiting(boolean fruiting) {
        this.fruiting = fruiting;
    }
```

```
    public abstract void printInfo();
}
```

AppleTree.java

```
public class AppleTree extends Tree{
    AppleTree(int _age, boolean _fruiting){
        super(_age, _fruiting);
    }
    @Override
    public void printInfo(){
        System.out.println("Apple tree\nage: "+age+"\nfruiting: "+fruiting);
        if(age>8 || !fruiting){
            System.out.println("needs a transplant!");
        }
    }
}
```

PearTree.java

```
public class PearTree extends Tree{
    PearTree(int _age, boolean _fruiting){
        super(_age, _fruiting);
    }
    @Override
    public void printInfo(){
        System.out.println("Pear tree\nage: "+age+"\nfruiting: "+fruiting);
        if(age>6 || !fruiting){
            System.out.println("needs a transplant!");
        }
    }
}
```

CherryTree.java

```
public class CherryTree extends Tree {
    CherryTree(int _age, boolean _fruiting){
        super(_age, _fruiting);
    }
    @Override
    public void printInfo(){
        System.out.println("Cherry tree\nage: "+age+"\nfruiting: "+fruiting);
        if(age>10 || !fruiting){
            System.out.println("needs a transplant!");
        }
    }
}
```

Task2.java

```
import java.util.Vector;

public class task2 {
    public static void main(String[] args) {
        Vector<Tree> trees= new Vector<>();
        trees.add(new AppleTree(5, true));
        trees.add(new CherryTree(8, false));
        trees.add(new PearTree(8, true));
        for(Tree tree: trees){
            tree.printInfo();
        }
    }
}
```

Пример

```
D:\СПП\lab5\out\production\lab5>java task2
Apple tree
age: 5
fruiting: true
Cherry tree
age: 8
fruiting: false
needs a transplant!
Pear tree
age: 8
fruiting: true
needs a transplant!
```

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

Person.java

```java
import java.util.Vector;

abstract class Person {
    protected String firstName;
    protected String lastName;

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    @Override
    public String toString(){
        String resultStr=getLastName()+" "+getFirstName();
        return resultStr;
    }
}
```

Applicant.java

```java
import java.util.Vector;

public class Applicant extends Person {
    private Vector<Mark> markList= new Vector<>();

    public Applicant(String firstName, String lastName) {
        super(firstName, lastName);
    }

    public Vector<Mark> getMarkList() {
```

```
        return markList;
    }

    public void setMarkList(Vector<Mark> markList) {
        this.markList = markList;
    }
    public void addMark(Mark mark){
        markList.add(mark);
    }
    @Override
    public String toString(){
        String resultStr=getLastName()+" "+getFirstName();
        return resultStr;
    }
    public void printMarkList(){
        for(Mark mark:markList){
            System.out.println(mark);
        }
    }
}
```

Exam.java

```
import java.util.Scanner;
import java.util.Vector;

public class Exam {
    private static String name;
    private Teacher teacher;

    public Exam(String name, Teacher teacher) {
        this.name = name;
        this.teacher = teacher;
    }

    public static String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
    public void giveMarks(Vector<Applicant> applicantList){
        System.out.println("Exam name: "+Exam.getName());
        Scanner scanner=new Scanner(System.in);
        for(Applicant applicant:applicantList){
            System.out.print("Set mark for applicant
"+applicant.getLastName()+" "+applicant.getFirstName()+": ");
            int mark = scanner.nextInt();
            applicant.addMark(new Mark(mark, this));
        }
    }
    @Override
    public String toString(){
        String resultStr="Exam name: "+getName()+"\nteacher: "+teacher;
        return resultStr;
    }
}
```

Faculty.java

```
import java.util.Vector;

public class Faculty {
    private String name;
    private Vector<Applicant> applicantList= new Vector<>();
    private Vector<Applicant> acceptedApplicantsList= new Vector<>();
```

```
    private Vector<Exam> examList= new Vector<>();

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
    public void registerApplicant(Applicant applicant){
        applicantList.add(applicant);
    }
    public void addExam(Exam exam){
        examList.add(exam);
    }
    public void passExams(){
        for(Exam exam:examList){
            exam.giveMarks(applicantList);
        }
        for(Applicant applicant:applicantList){
            float GPA=0;
            for(Mark mark:applicant.getMarkList()){
                GPA+= mark.getMark();
            }
            GPA/=applicant.getMarkList().size();
            if(GPA>=6){
                acceptedApplicantsList.add(applicant);
            }
        }
    }
    public void printInfo(){
        System.out.println("Faculty name: "+getName()+"\nexam
list:\n"+examList);
    }
    public void printAcceptedApplicantsList(){
        System.out.println("Accepted applicants:");
        for(Applicant applicant:acceptedApplicantsList){
            System.out.println(applicant);
        }
    }
public void printApplicantList(){
    System.out.println("Applicants:");
    for(Applicant applicant:applicantList){
        System.out.println(applicant);
    }
}

}
```

Mark.java

```
public class Mark {
    private int mark;
    private Exam exam;

    public Mark(int mark, Exam exam) {
        this.mark = mark;
        this.exam = exam;
    }

    public int getMark() {
```

```
        return mark;
    }

    public void setMark(int mark) {
        this.mark = mark;
    }

    public Exam getExam() {
        return exam;
    }

    public void setExam(Exam exam) {
        this.exam = exam;
    }
    @Override
    public String toString(){
        String resultStr=getExam()+"\nmark: "+getMark();
        return  resultStr;
    }
}
```

Teacher.java

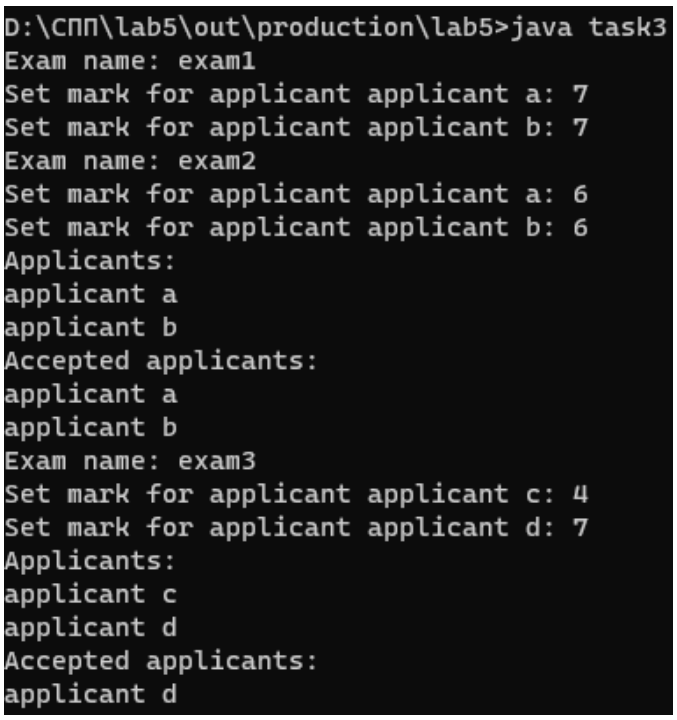
```
public class Teacher extends Person {

    public Teacher(String firstName, String lastName) {
        super(firstName, lastName);
    }
}
```

Task3.java

```
public class task3 {
    public static void main(String[] args){
        Faculty faculty1 = new Faculty("Faculty a");
        Faculty faculty2 = new Faculty("Faculty b");
        Teacher teacher1 = new Teacher("a", "teacher");
        Teacher teacher2 = new Teacher("b", "teacher");
        Teacher teacher3 = new Teacher("c", "teacher");
        Applicant applicant1 = new Applicant("a", "applicant");
        Applicant applicant2 = new Applicant("b", "applicant");
        Applicant applicant3 = new Applicant("c", "applicant");
        Applicant applicant4 = new Applicant("d", "applicant");
        faculty1.addExam(new Exam("exam1", teacher1));
        faculty1.addExam(new Exam("exam2", teacher2));
        faculty2.addExam(new Exam("exam3", teacher3));
        faculty1.registerApplicant(applicant1);
        faculty1.registerApplicant(applicant2);
        faculty2.registerApplicant(applicant3);
        faculty2.registerApplicant(applicant4);
        faculty1.passExams();
        faculty1.printApplicantList();
        faculty1.printAcceptedApplicantsList();
        faculty2.passExams();
        faculty2.printApplicantList();
        faculty2.printAcceptedApplicantsList();
    }
}
```

Пример:



Вывод: научились создавать и использовать классы в программах на языке программирования Java.