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:

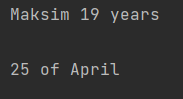
public class Lab5\_1  
{  
  
 public static void main(String[] args)  
 {  
 director d = new director("Maksim", 19, "25 of April");  
 d.show();  
 d.payday();  
 }  
  
}

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();  
 }  
 }  
  
}

musicInst.java(Задание 2):

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

musicInst.java(Задание 3):

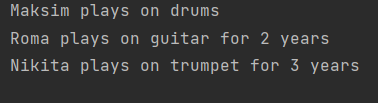
abstract class musicInst  
{  
 private String inst rument;  
  
 public musicInst(String \_inst)  
 {  
 instrument = \_inst;  
 }  
  
 public String getInst()  
 {  
 return instrument;  
 }  
  
 public abstract void show();  
}

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");  
 }  
}

Результат выполнения программы:



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