**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра «Электронные вычислительные машины»

**Отчет о лабораторной работе № 4**

по теме

«Наследование и полиморфизм»

**Выполнили:**

Студенты гр. 045

Бригады №10

Анохин В.А.

Вашкулатов Н.А.

**Проверил:**

Асс. Бастрычкин А.С.

Цель работы:изучение принципов ООП, приобретение навыков использования наследования и полиморфизма в Java-программах.

Ход работы

Задание №1

Создать объект класса Пианино, используя классы Клавиша, Педаль. Методы: настроить, играть на пианино, нажимать клавишу.

Код программы:

Класс Piano

import java.util.ArrayList;  
  
public class Piano {  
 private ArrayList<Key> keys;  
  
 Piano(int numButtons){  
 keys = new ArrayList<Key>();  
 for (int i=0; i<numButtons; ++i){  
 keys.add(new Button(i+1));  
 }  
 keys.add(new Pedal(PedalType.LEFT));  
 keys.add(new Pedal(PedalType.RIGHT));  
 }  
  
 public void play(int[] melody){  
 try {  
 for(int i : melody){  
 keys.get(i).press();  
 }  
 }  
 catch (IndexOutOfBoundsException e){  
 System.err.println("Клавиша не найдена\n"+e.getMessage());  
 }  
 }  
  
 public void config(){  
 for(Key k : keys){  
 k.configure();  
 }  
 }  
}

Интерфейс Key

public interface Key {  
 void configure();  
 void press();  
}

Перечисление PedalType

public enum PedalType {  
 LEFT,  
 RIGHT,  
 MIDDLE;  
  
 @Override  
 public String toString() {  
 return switch (this) {  
 case LEFT -> "Левая";  
 case RIGHT -> "Правая";  
 case MIDDLE -> "Средняя";  
 };  
 }  
}

Класс Pedal

public class Pedal implements Key{  
 PedalType type;  
  
 Pedal(PedalType pedalType){  
 type = pedalType;  
 }  
 @Override  
 public void press(){  
 System.out.println(type+" педаль нажата ");  
 }  
 @Override  
 public void configure(){  
 System.out.println(type+" педаль настроена");  
 }  
}

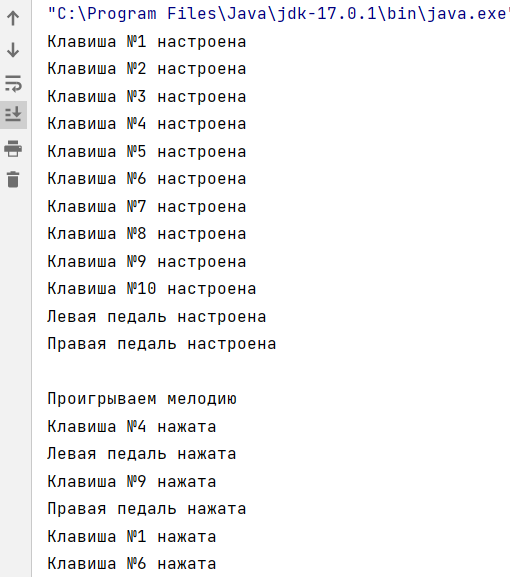
Класс Button

public class Button implements Key{  
 int number;  
  
 Button(int number){  
 this.number = number;  
 }  
 @Override  
 public void press(){  
 System.out.println("Клавиша №"+number+" нажата");  
 }  
 @Override  
 public void configure(){  
 System.out.println("Клавиша №"+number+" настроена");  
 }  
}

Основная программа

public class App {  
 public static void main(String[] args) {  
 Piano piano = new Piano(10);  
 piano.config();  
 System.out.println();  
 System.out.println("Проигрываем мелодию");  
 int[] melody = {3, 10, 8, 11, 0, 5};  
 piano.play(melody);  
 }  
}

Окно вывода:



Задание №2

**Мобильная связь**. Определить иерархию тарифов мобильной компании. Создать список тарифов компании. Подсчитать общую численность клиентов. Провести сортировку тарифов на основе размера абонентской платы. Найти тариф в компании, соответствующий заданному диапазону параметров.

Код программы:

Класс MobileTariff

public abstract class MobileTariff {  
 private int users;  
 private String name;  
 private int costInRubles;  
  
 public MobileTariff(int users, String name, int costInRubles) {  
 this.users = users;  
 this.name = name;  
 this.costInRubles = costInRubles;  
 }  
  
 public int getUsers() {  
 return users;  
 }  
  
 public void setUsers(int users) {  
 this.users = users;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getCostInRubles() {  
 return costInRubles;  
 }  
  
 public void setCostInRubles(int costInRubles) {  
 this.costInRubles = costInRubles;  
 }  
  
 public void addUsers(int count){  
 users =+ users;  
 };  
}

Класс TariffCostComparator

import java.util.Comparator;  
  
public class TariffCostComparator implements Comparator<MobileTariff> {  
  
 @Override  
 public int compare(MobileTariff o1, MobileTariff o2) {  
 if (o1.getCostInRubles() - o2.getCostInRubles() < 0){  
 return -1;  
 }  
 else return 1;  
 }  
}

Класс TabletTariff

public class TabletTariff extends MobileTariff{  
 private double mBOfInternet;  
  
 public TabletTariff(int users, String name, int costInRubles, double mBOfInternet) {  
 super(users, name, costInRubles);  
 this.mBOfInternet = mBOfInternet;  
 }  
  
 public double getmBOfInternet() {  
 return mBOfInternet;  
 }  
  
 public void setmBOfInternet(double mBOfInternet) {  
 this.mBOfInternet = mBOfInternet;  
 }  
  
 @Override  
 public String toString() {  
 return "TabletTariff{" +  
 "users " + getUsers() + '\'' +  
 "name='" + getName() + '\'' +  
 ", costInRubles=" + getCostInRubles() +  
 ", mBOfInternet=" + mBOfInternet +  
 '}';  
 }  
}

Класс PhoneTariff

public class PhoneTariff extends MobileTariff{  
 private int minutesToTalk;  
  
 public int getMinutesToTalk() {  
 return minutesToTalk;  
 }  
  
 public void setMinutesToTalk(int minutesToTalk) {  
 this.minutesToTalk = minutesToTalk;  
 }  
  
 public PhoneTariff(int users, String name, int costInRubles, int minutesToTalk) {  
 super(users, name, costInRubles);  
 this.minutesToTalk = minutesToTalk;  
 }  
  
 @Override  
 public String toString() {  
 return "PhoneTariff{" +  
 "users= " + getUsers() + '\'' +  
 "name='" + getName() + '\'' +  
 ", costInRubles=" + getCostInRubles() +  
 ", minutesToTalk=" + minutesToTalk +  
 '}';  
 }  
}

Класс InternetPhoneTariff

public class InternetPhoneTariff extends PhoneTariff {  
 private double mBOfInternet;  
  
 public InternetPhoneTariff(int users, String name, int costInRubles, int minutesToTalk, double mBOfInternet) {  
 super(users, name, costInRubles, minutesToTalk);  
 this.mBOfInternet = mBOfInternet;  
 }  
  
 public double getmBOfInternet() {  
 return mBOfInternet;  
 }  
  
 public void setmBOfInternet(double mBOfInternet) {  
 this.mBOfInternet = mBOfInternet;  
 }  
  
 @Override  
 public String toString() {  
 return "InternetPhoneTariff{" +  
 "users= " + getUsers() + '\'' +  
 "minutesToTalk= " + getMinutesToTalk() + '\'' +  
 "mBOfInternet=" + mBOfInternet +  
 ", name='" + getName() + '\'' +  
 ", costInRubles=" + getCostInRubles() +  
 '}';  
 }  
}

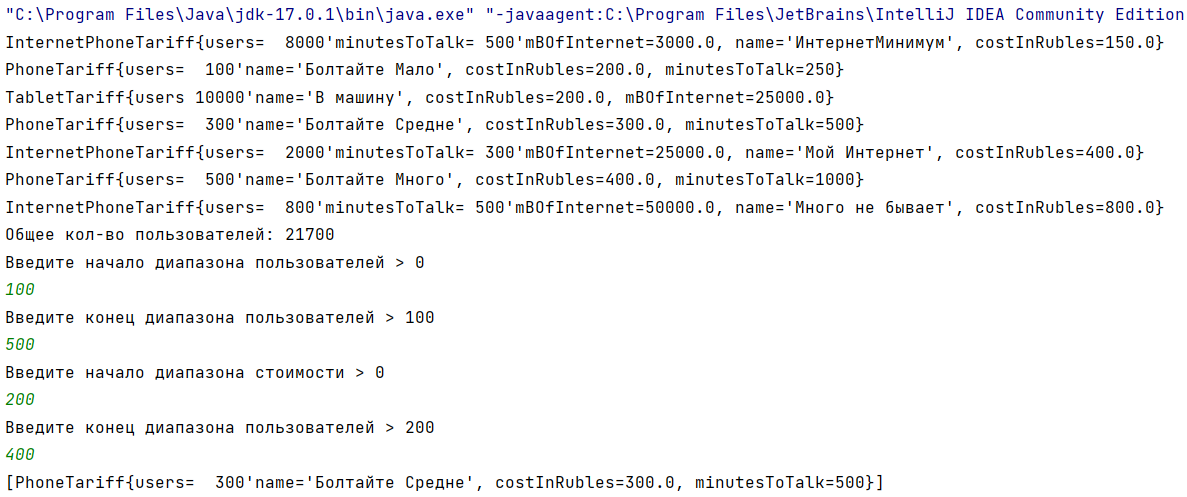
Класс MobileTariffAction

public class MobileTariffAction {  
 public static ArrayList<MobileTariff> filterByUsersAndCost(ArrayList<MobileTariff> tariffs, int startUserRange, int endUserRange, int startCostRange, int endCostRange){  
 ArrayList<MobileTariff> result = new ArrayList<>();  
 for (MobileTariff tariff: tariffs) {  
 if((tariff.getUsers() > startUserRange) && (tariff.getUsers() < endUserRange) && (tariff.getCostInRubles() > startCostRange) && (tariff.getCostInRubles() < endCostRange)){  
 result.add(tariff);  
 }  
 }  
 return result;  
 }  
}

Основная программа

public class App {  
 public static int getSummaryUsers(ArrayList<MobileTariff> tariffs) {  
 int sum = 0;  
 for (MobileTariff tariff : tariffs) {  
 sum += tariff.getUsers();  
 }  
 return sum;  
 }  
  
 public static void main(String[] args) {  
 ArrayList<MobileTariff> tariffs = new ArrayList<MobileTariff>(){  
 {  
 add(new InternetPhoneTariff(2000,"Мой Интернет", 400, 300, 25000 ));  
 add(new InternetPhoneTariff(800, "Много не бывает", 800, 500, 50000));  
 add(new InternetPhoneTariff(8000, "ИнтернетМинимум", 150,500,3000));  
 add(new PhoneTariff(500,"Болтайте Много", 400, 1000));  
 add(new PhoneTariff(300,"Болтайте Средне", 300, 500));  
 add(new PhoneTariff(100,"Болтайте Мало", 200, 250));  
 add(new TabletTariff(10000,"В машину", 200,25000));  
 }  
 };  
 tariffs.sort(new TariffCostComparator());  
 tariffs.forEach(System.out::println);  
 System.out.println("Общее кол-во пользователей: " + getSummaryUsers(tariffs));  
 Scanner sc = new Scanner(System.in);  
 int startUserRange = 0;  
 int endUserRange = 0;  
 while (startUserRange == 0) {  
 System.out.println("Введите начало диапазона пользователей > 0 ");  
 try {  
 startUserRange = Integer.parseInt(sc.nextLine());  
 if (startUserRange < 0){  
 startUserRange = 0;  
 }  
 } catch (NumberFormatException e){ // Если была введена буква, выполнение не завершится  
 System.err.println("Введите число");  
 }  
 }  
 while (endUserRange == 0) {  
 System.out.println("Введите конец диапазона пользователей > " + startUserRange );  
 try {  
 endUserRange = Integer.parseInt(sc.nextLine());  
 if(startUserRange >= endUserRange){  
 endUserRange = 0;  
 }  
 } catch (NumberFormatException e){  
 System.err.println("Введите число");  
 }  
 }  
 int startCostRange = 0;  
 int endCostRange = 0;  
 while (startCostRange == 0) {  
 System.out.println("Введите начало диапазона стоимости > 0 ");  
 try {  
 startCostRange = Integer.parseInt(sc.nextLine());  
 if (startCostRange < 0){  
 startCostRange = 0;  
 }  
 } catch (NumberFormatException e){ // Если была введена буква, выполнение не завершится  
 System.err.println("Введите число");  
 }  
 }  
 while (endCostRange == 0) {  
 System.out.println("Введите конец диапазона пользователей > " + startCostRange );  
 try {  
 endCostRange = Integer.parseInt(sc.nextLine());  
 if(startCostRange >= endCostRange){  
 endCostRange = 0;  
 }  
 } catch (NumberFormatException e){  
 System.err.println("Введите число");  
 }  
 }  
 System.out.println(MobileTariffAction.filterByUsersAndCost(tariffs,startUserRange,endUserRange,startCostRange,endCostRange));  
 }  
}

Окно вывода:



**Вывод:** мы изучили принципы ООП, приобрели навыки использования наследования и полиморфизма в Java-программах.