|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 3** |

Вариант 6

**Название:** Класса. Наследование. Полиморфизм.

**Дисциплина:** Языки программирования для работы с большими данными

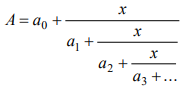
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-23М |  |  | В.А. Гордеев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**ВВЕДЕНИЕ**

**Задание:**

* 1. Определить класс Цепная дробь



Определить методы сложения, вычитания, умножения, деления. Вычислить значение для заданного n, x, a[n].

* 1. Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей, ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.
  2. Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: a) список квартир, имеющих заданное число комнат; b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; c) список квартир, имеющих площадь, превосходящую заданную.
  3. Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль. Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров. Создать массив объектов. Вывести: a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; b) сведения об абонентах, которые пользовались междугородной связью; c) сведения об абонентах в алфавитном порядке.
  4. Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString(). Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завять, вывести на консоль цвет бутона.
  5. Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString(). Создать объект класса Дерево, используя классы Лист. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.
  6. Построить модель программной системы. Система Конструкторское бюро. Заказчик представляет Техническое Задание (ТЗ) на проектирование многоэтажного Дома. Конструктор регистрирует ТЗ, определяет стоимость проектирования и строительства, выставляет Заказчику Счет за проектирование и создает Бригаду Конструкторов для выполнения Проекта.
  7. Построить модель программной системы. Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.

**ОСНОВНАЯ ЧАСТЬ**

**Задание 1.1**

Был определен класс цепная дробь, а также определены методы сложения, вычитания, умножения и деления. Вычислены значения для заданных величин.

Код программы приведен в листингах 1 и 2. Результаты работы программы на рисунке 1.

Листинг 1 – Код класса ChainFraction

public class ChainFraction {  
 public int n;  
 public int x;  
 public double[] a;  
 public ChainFraction(int x, int n, double[] a) {  
 this.x = x;  
 this.n = n;  
 this.a = a;  
 }  
 public ChainFraction(double[] a) {  
 this.a = a;  
 }  
 public double getValue() {  
 double value = a[n-1];  
 for (int i = n - 2; i >= 0; i--) {  
 value = a[i] + x / value;  
 }  
 return value;  
 }  
 public double add(ChainFraction f) {  
 double fValue1 = this.getValue();  
 double fValue2 = f.getValue();  
 return fValue1 + fValue2;  
 }  
 public double subtract(ChainFraction f) {  
 double fValue1 = this.getValue();  
 double fValue2 = f.getValue();  
 return fValue1 - fValue2;  
 }  
 public double multiply(ChainFraction f) {  
 double fValue1 = this.getValue();  
 double fValue2 = f.getValue();  
 return fValue1 \* fValue2;  
 }  
 public double divide(ChainFraction f) {  
 double fValue1 = this.getValue();  
 double fValue2 = f.getValue();  
 return fValue1 / fValue2;  
 }  
}

Листинг 2 – Код класса Main

// Определить класс Цепная дробь.  
// Определить методы сложения, вычитания, умножения, деления.  
// Вычислить значение для заданного n, x, a[n].  
import java.util.Scanner;  
public class Main {  
 public static void main(String[] args) {  
  
 ChainFraction chainFraction1 = *createChainFraction*();  
 ChainFraction chainFraction2 = *createChainFraction*();  
  
 double resultOfAdd = chainFraction1.add(chainFraction2);  
 double resultOfSubtract = chainFraction1.subtract(chainFraction2);  
 double resultOfMultiply = chainFraction1.multiply(chainFraction2);  
 double resultOfDivide = chainFraction1.divide(chainFraction2);  
  
 System.*out*.println("Значение первой цепной дроби: " + chainFraction1.getValue());  
 System.*out*.println("Значение второй цепной дроби: " + chainFraction2.getValue());  
  
 System.*out*.println("Результат сложения двух цепных дробей: " + resultOfAdd);  
 System.*out*.println("Результат вычитания двух цепных дробей: " + resultOfSubtract);  
 System.*out*.println("Результат умножения двух цепных дробей: " + resultOfMultiply);  
 System.*out*.println("Результат деления двух цепных дробей: " + resultOfDivide);  
 }  
  
 public static ChainFraction createChainFraction() {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Введите n - количество коэфициентов: ");  
 int n = scanner.nextInt();  
  
 System.*out*.print("Введите x - числитель цепной дроби: ");  
 int x = scanner.nextInt();  
  
 double[] a = new double[n];  
 System.*out*.println("Введите a[n] - коэфициенты: ");  
 for (int i = 0; i < n; i++) {  
 System.*out*.print("Введите коэфициент № " + (i+1) + " ");  
 a[i] = scanner.nextInt();  
 }  
  
 return new ChainFraction(x, n, a);  
 }  
}

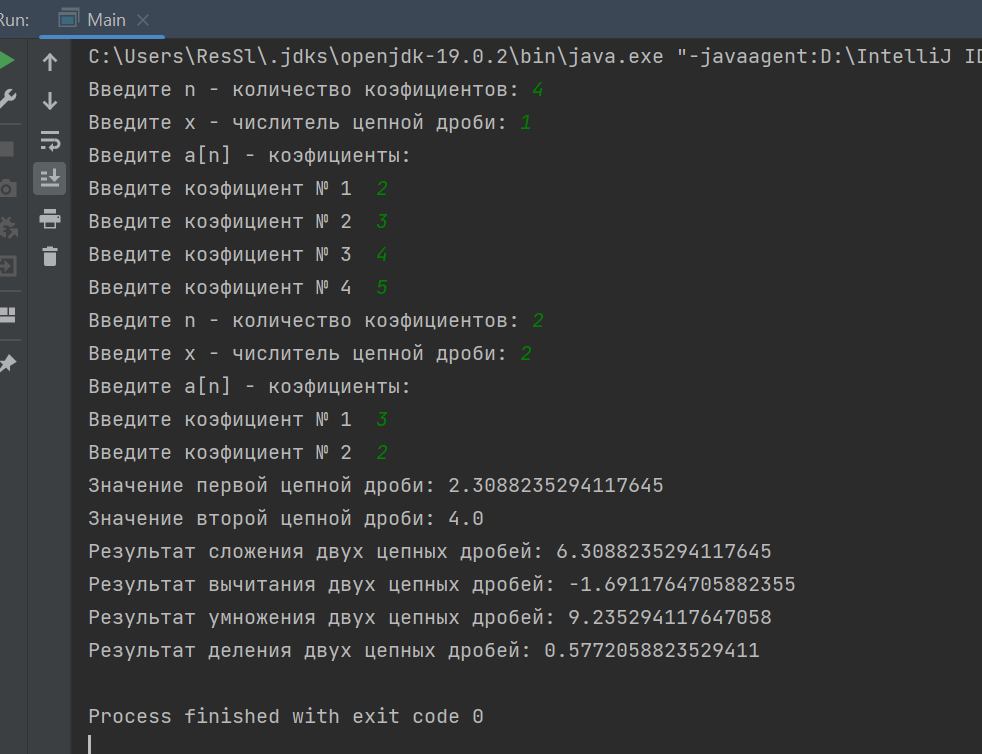


Рисунок 1 – Результат работы программы 1.1

**Задание 1.2**

Определен класс Дробь в виде пары (m,n). Класс содержит несколько конструкторов. Реализованы методы для сложения, вычитания, умножения и деления дробей. Объявлен массив из k дробей, введены/выведены значения для массива дробей. Создан массив объектов и передан в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.

Код представлен в листингах 3, 4. Результат работы программы показан на рисунке 2.

Листинг 3 – Код класса Fraction

public class Fraction {  
 private int m; // числитель  
 private int n; // знаменатель  
 public Fraction() { //Конструктор при числителе 0  
 m = 0;  
 n = 1;  
 }  
 public Fraction(int m) { // Конструктор при знаменателе 0  
 this.m = m;  
 this.n = 1;  
 }  
 public Fraction(int m, int n) { //Конструктор обыкновенной дроби  
 this.m = m;  
 this.n = n;  
 }  
 public Fraction add(Fraction f) { // Метод сложения дробей  
 int nm = m \* f.n + n \* f.m;  
 int nn = n \* f.n;  
 return new Fraction(nm, nn);  
 }  
 public Fraction subtract(Fraction f) { // Метод вычитания дробей  
 int nm = m \* f.n - n \* f.m;  
 int nn = n \* f.n;  
 return new Fraction(nm, nn);  
 }  
 public Fraction multiply(Fraction f) { // Метод умножения дробей  
 int nm = m \* f.m;  
 int nn = n \* f.n;  
 return new Fraction(nm, nn);  
 }  
 public Fraction divide(Fraction f) { // Метод деления дробей  
 int nm = m \* f.n;  
 int nn = n \* f.m;  
 return new Fraction(nm, nn);  
 }  
 public void print() { // Метод вывода  
 System.*out*.println(m + "/" + n);  
 }  
}

Листинг 4 – Код класса Main

// Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов.  
// Реализовать методы для сложения, вычитания, умножения и деления дробей.  
// Объявить массив из k дробей, ввести/вывести значения для массива дробей.  
// Создать массив объектов и передать его в метод,  
// который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.  
import java.util.Scanner;  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Введите количество дробей: ");  
 int count = scanner.nextInt();  
 Fraction[] fractions = new Fraction[count];  
  
 for (int i = 0; i < count; i++) {  
 System.*out*.print("Введите числитель дроби № " + (i+1) + " ");  
 int m = scanner.nextInt();  
 if (m == 0) {  
 fractions[i] = new Fraction();  
 continue;  
 }  
 System.*out*.print("Введите знаменатель дроби № " + (i+1) + " ");  
 int n = scanner.nextInt();  
 if (n == 0) {  
 fractions[i] = new Fraction(m);  
 continue;  
 }  
 fractions[i] = new Fraction(m, n);  
 }  
  
 System.*out*.println("Введенный массив дробей: ");  
 for (int i = 0; i < count; i++) {  
 fractions[i].print();  
 }  
  
 *modifyArray*(fractions);  
  
 System.*out*.println("Полученный массив дробей:");  
 for (int i = 0; i < count; i++) {  
 fractions[i].print();  
 }  
 }  
 public static void modifyArray(Fraction[] fractions) {  
 int count = fractions.length;  
 for (int i = 0; i < count; i += 2) {  
 if (i+1 < count) {  
 fractions[i] = fractions[i].add(fractions[i+1]);  
 }  
 }  
 }  
}

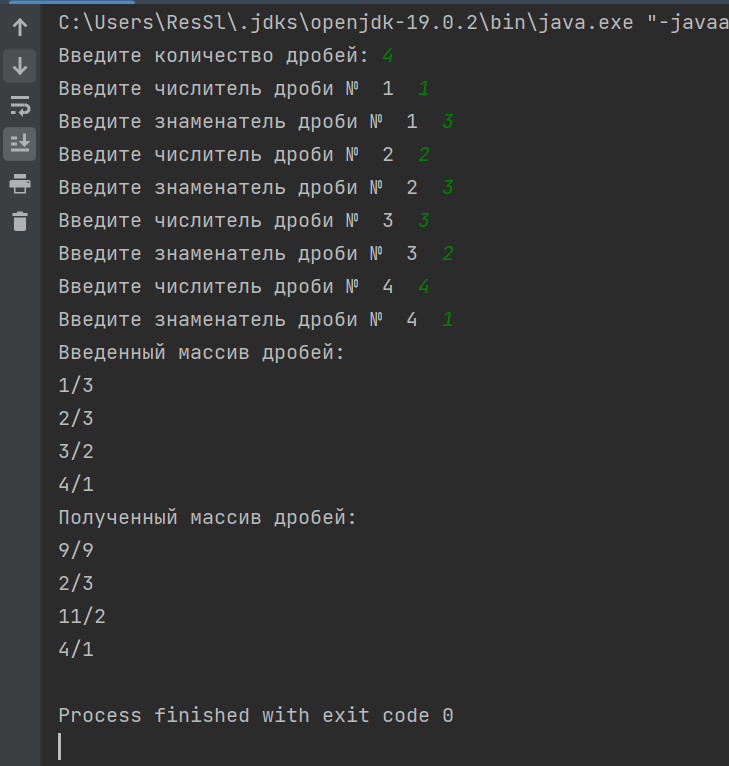


Рисунок 2 – Результат работы программы 1.2

**Задание 2.1**

Созданы классы, спецификации которых приведены ниже. Определены конструкторы и методы setТип(), getТип(), toString(). Определены дополнительно методы в классе, создающем массив объектов. Заданы критерии выбора данных и выведены эти данные на консоль.

Класс: House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создан массив объектов. Возможно вывести: a) список квартир, имеющих заданное число комнат; b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; c) список квартир, имеющих площадь, превосходящую заданную.

Код программы представлен в листингах 5, 6, 7. Результат работы программы представлен на рисунке 3.

Листинг 5 – Код класса City

import java.util.Arrays;  
public class City {  
 House[] houses;  
 public City(House[] houses) {  
 this.houses = houses;  
 }  
  
 public void roomFilter(int numberOfRooms) { //Квартиры, имеющие заданное число комнат  
 System.*out*.println("numberOfRooms filter: NumberOfRooms = " + numberOfRooms);  
 Arrays.*stream*(this.houses)  
 .filter(elem -> elem.getNumberOfRooms() == numberOfRooms)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 public void floorAndRoomFilter(int numberOfRooms, int minFloor, int maxFloor) { // Расположенных на этаже, который находится в заданном промежутке  
 System.*out*.println("floorRoom filter. Room: " + numberOfRooms + "; " + minFloor + "<= FLOOR <=" + maxFloor);  
 Arrays.*stream*(this.houses)  
 .filter(elem -> elem.getFloor() >= minFloor && elem.getFloor() <= maxFloor)  
 .filter(elem -> elem.getNumberOfRooms() == numberOfRooms)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 public void squareFilter(int square) { // список квартир, имеющих площадь, превосходящую заданную  
 System.*out*.println("square filter: SQUARE >= " + square);  
 Arrays.*stream*(this.houses)  
 .filter(elem -> elem.getSquare() >= square)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
}

Листинг 6 – Код класса House

// Создать классы, спецификации которых приведены ниже.  
// Определить конструкторы и методы setТип(), getТип(), toString().  
// Определить дополнительно методы в классе, создающем массив объектов.  
// Задать критерий выбора данных и вывести эти данные на консоль.  
  
//House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации.  
// Создать массив объектов.  
// Вывести:  
// a) список квартир, имеющих заданное число комнат;  
// b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке;  
// c) список квартир, имеющих площадь, превосходящую заданную.  
public class House {  
 private int id;  
 private int number;  
 private double square;  
 private int floor;  
 private int numberOfRooms;  
 private String street;  
 private String typeBuilding;  
 private int yearsOfExploitation;  
 public House(int id,  
 int number,  
 double square,  
 int floor,  
 int numberOfRooms,  
 String street,  
 String typeBuilding,  
 int yearsOfExploitation) {  
 this.id = id;  
 this.number = number;  
 this.square = square;  
 this.floor = floor;  
 this.numberOfRooms = numberOfRooms;  
 this.street = street;  
 this.typeBuilding = typeBuilding;  
 this.yearsOfExploitation = yearsOfExploitation;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public int getNumber() {  
 return number;  
 }  
  
 public void setNumber(int number) {  
 this.number = number;  
 }  
  
 public double getSquare() {  
 return square;  
 }  
  
 public void setSquare(double square) {  
 this.square = square;  
 }  
  
 public int getFloor() {  
 return floor;  
 }  
  
 public void setFloor(int floor) {  
 this.floor = floor;  
 }  
  
 public int getNumberOfRooms() {  
 return numberOfRooms;  
 }  
  
 public void setNumberOfRooms(int numberOfRooms) {  
 this.numberOfRooms = numberOfRooms;  
 }  
  
 public String getStreet() {  
 return street;  
 }  
  
 public void setStreet(String street) {  
 this.street = street;  
 }  
  
 public String getTypeBuilding() {  
 return typeBuilding;  
 }  
  
 public void setTypeBuilding(String typeBuilding) {  
 this.typeBuilding = typeBuilding;  
 }  
  
 public int getYearsOfExploitation() {  
 return yearsOfExploitation;  
 }  
  
 public void setYearsOfExploitation(int yearsOfExploitation) {  
 this.yearsOfExploitation = yearsOfExploitation;  
 }  
  
 public String toString() {  
 return "House: { " +  
 "id: " + id + "; " +  
 "number: " + number + "; " +  
 "square: " + square + "; " +  
 "floor: " + floor + "; " +  
 "numberOfRooms: " + numberOfRooms + "; \n" +  
 "street: " + street +  
 "; type: " + typeBuilding +  
 "; yearsOfExploitation: " + yearsOfExploitation +  
 "}";  
 }  
}

Листинг 7 – Код класса Main

public class Main {  
 public static void main(String[] args) {  
 House[] houses = new House[9];  
  
 houses[0] = new House(1, 22, 30, 5, 3, "Gurieva", "dom", 12);  
 houses[1] = new House(2, 23, 50, 1, 5, "Krasnoarmeyskaya", "sky", 8);  
 houses[2] = new House(3, 24, 40, 8, 2, "Popova", "boat", 6);  
 houses[3] = new House(4, 24, 40, 10, 1, "Dergaevskaya", "sky", 6);  
 houses[4] = new House(5, 12, 60, 2, 5, "Gurieva", "sky", 2);  
 houses[5] = new House(6, 20, 65, 3, 2, "Gurieva", "dom", 6);  
 houses[6] = new House(7, 22, 10, 3, 2, "Popova", "boat", 2);  
 houses[7] = new House(8, 22, 20, 5, 1, "Dergaevskaya", "dom", 3);  
 houses[8] = new House(9, 12, 40, 1, 3, "Dergaevskaya", "dom", 10);  
  
 City moscow = new City(houses);  
  
 moscow.roomFilter(2);  
 moscow.floorAndRoomFilter(3,3, 6);  
 moscow.squareFilter(60);  
 }  
  
}

Рисунок 3 – Результат работы программы 2.1

**Задание 2.2**

Созданы классы, спецификации которых приведены ниже. Определены конструкторы и методы setТип(), getТип(), toString(). Определены дополнительно методы в классе, создающем массив объектов. Задан критерий выбора данных и выведены эти данные на консоль.

Классы: Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров. Создан массив объектов. Выведены: a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; b) сведения об абонентах, которые пользовались междугородной связью; c) сведения об абонентах в алфавитном порядке.

Код программы представлен в листингах 8, 9, 10. Результат работы программы показан на рисунке 4.

Листинг 8 – Код класса Phone

// Создать классы, спецификации которых приведены ниже.  
// Определить конструкторы и методы setТип(), getТип(), toString().  
// Определить дополнительно методы в классе, создающем массив объектов.  
// Задать критерий выбора данных и вывести эти данные на консоль.  
  
// Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит,  
// Время городских и междугородных разговоров.  
// Создать массив объектов. Вывести:  
// a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное;  
// b) сведения об абонентах, которые пользовались междугородной связью;  
// c) сведения об абонентах в алфавитном порядке.  
public class Phone {  
 private int id;  
 private String firstName;  
 private String lastName;  
 private String patronymic;  
 private String address;  
 private int numberCreditCard;  
 private int debet;  
 private int kredit;  
 private int cityTalkTime;  
 private int distanceTalkTime;  
  
 public Phone(int id,  
 String firstName,  
 String lastName,  
 String patronymic,  
 String address,  
 int numberCreditCard,  
 int debet,  
 int kredit,  
 int cityTalkTime,  
 int distanceTalkTime) {  
 this.id = id;  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.patronymic = patronymic;  
 this.address = address;  
 this.numberCreditCard = numberCreditCard;  
 this.debet = debet;  
 this.kredit = kredit;  
 this.cityTalkTime = cityTalkTime;  
 this.distanceTalkTime = distanceTalkTime;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getPatronymic() {  
 return patronymic;  
 }  
  
 public void setPatronymic(String patronymic) {  
 this.patronymic = patronymic;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public int getNumberCreditCard() {  
 return numberCreditCard;  
 }  
  
 public void setNumberCreditCard(int numberCreditCard) {  
 this.numberCreditCard = numberCreditCard;  
 }  
  
 public int getDebet() {  
 return debet;  
 }  
  
 public void setDebet(int debet) {  
 this.debet = debet;  
 }  
  
 public int getKredit() {  
 return kredit;  
 }  
  
 public void setKredit(int kredit) {  
 this.kredit = kredit;  
 }  
  
 public int getCityTalkTime() {  
 return cityTalkTime;  
 }  
  
 public void setCityTalkTime(int cityTalkTime) {  
 this.cityTalkTime = cityTalkTime;  
 }  
  
 public int getDistanceTalkTime() {  
 return distanceTalkTime;  
 }  
  
 public void setDistanceTalkTime(int distanceTalkTime) {  
 this.distanceTalkTime = distanceTalkTime;  
 }  
  
  
 public String toString() {  
 return "House: { " +  
 "id: " + id + "; " +  
 "firstname: " + firstName + "; " +  
 "lastName: " + lastName + "; " +  
 "patronymic: " + patronymic + "; " +  
 "address: " + address + "; \n" +  
 "numberCreditCard: " + numberCreditCard +  
 "; debet: " + debet +  
 "; kredit: " + kredit +  
 "; cityTalkTime: " + cityTalkTime +  
 "; distanceTalkTime: " + distanceTalkTime +  
 "}";  
 }  
}

Листинг 9 – Код класса DatabasePhones

import java.util.Arrays;  
import java.util.Comparator;  
  
public class DatabasePhones {  
 Phone[] phones;  
  
 public DatabasePhones(Phone[] phones) {  
 this.phones = phones;  
 }  
  
 public void sortUsers() { // Cведения об абонентах в алфавитном порядке  
 System.*out*.println("Sorting users: ");  
 Arrays.*stream*(this.phones).sorted(new PhoneComparator())  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 public void usingDistanceTalkingFilter() { // сведения об абонентах, которые пользовались междугородной связью  
 System.*out*.println("using distance talking filter. DistanceTalkTime > 0");  
 Arrays.*stream*(this.phones)  
 .filter(elem -> elem.getDistanceTalkTime() > 0)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
 public void cityTalkFilter(int cityTime) { // сведения об абонентах, у которых время внутригородских разговоров превышает заданное;  
 System.*out*.println("city talk filter: cityTalkTime > " + cityTime);  
 Arrays.*stream*(this.phones)  
 .filter(elem -> elem.getCityTalkTime() > cityTime)  
 .forEach(System.*out*::println);  
 System.*out*.println();  
 }  
}  
  
class PhoneComparator implements Comparator<Phone> {  
 public int compare(Phone a, Phone b) {  
 return a.getLastName().toUpperCase().compareTo(b.getLastName().toUpperCase());  
 }  
}

Листинг 10 – Код класса Main

public class Main {  
 public static void main(String[] args) {  
 Phone[] phones = new Phone[9];  
  
 phones[0] = new Phone(1, "Vyacheslav", "Gordeev", "Aleksanrovich", "Ramenskoe",  
 123442, 123, 900, 12, 8);  
 phones[1] = new Phone(1, "Jonathan", "Jablonowski", "Batkovich", "Ramenskoe",  
 312423, 321, 344, 20, 0);  
 phones[2] = new Phone(1, "Keith", "Markovic", "Batkovich", "Ramenskoe",  
 423534, 423, 534, 19, 0);  
 phones[3] = new Phone(1, "Russel", "Van Dulken", "Alekseevich", "Moscow",  
 938494, 949, 932, 17, 0);  
 phones[4] = new Phone(1, "Kirill", "Karasiow", "Yanovich", "Moscow",  
 423949, 432, 926, 2, 15);  
 phones[5] = new Phone(1, "Robin", "Kool", "Borisovich", "Bataysk",  
 429394, 932, 238, 32, 0);  
 phones[6] = new Phone(1, "Žygimantas", "Chmieliauskas", "Aleksanrovich", "Krasnodar",  
 319024, 883, 493, 19, 8);  
 phones[7] = new Phone(1, "Mike", "Grzesiek", "Alekseevich", "Kazan",  
 942395, 612, 329, 11, 0);  
 phones[8] = new Phone(1, "Marcelo", "David", "Alekseevich", "Moscow",  
 150340, 322, 293, 12, 0);  
  
 DatabasePhones dbPhones = new DatabasePhones(phones);  
  
 dbPhones.sortUsers();  
 dbPhones.cityTalkFilter(20);  
 dbPhones.usingDistanceTalkingFilter();  
 }  
}

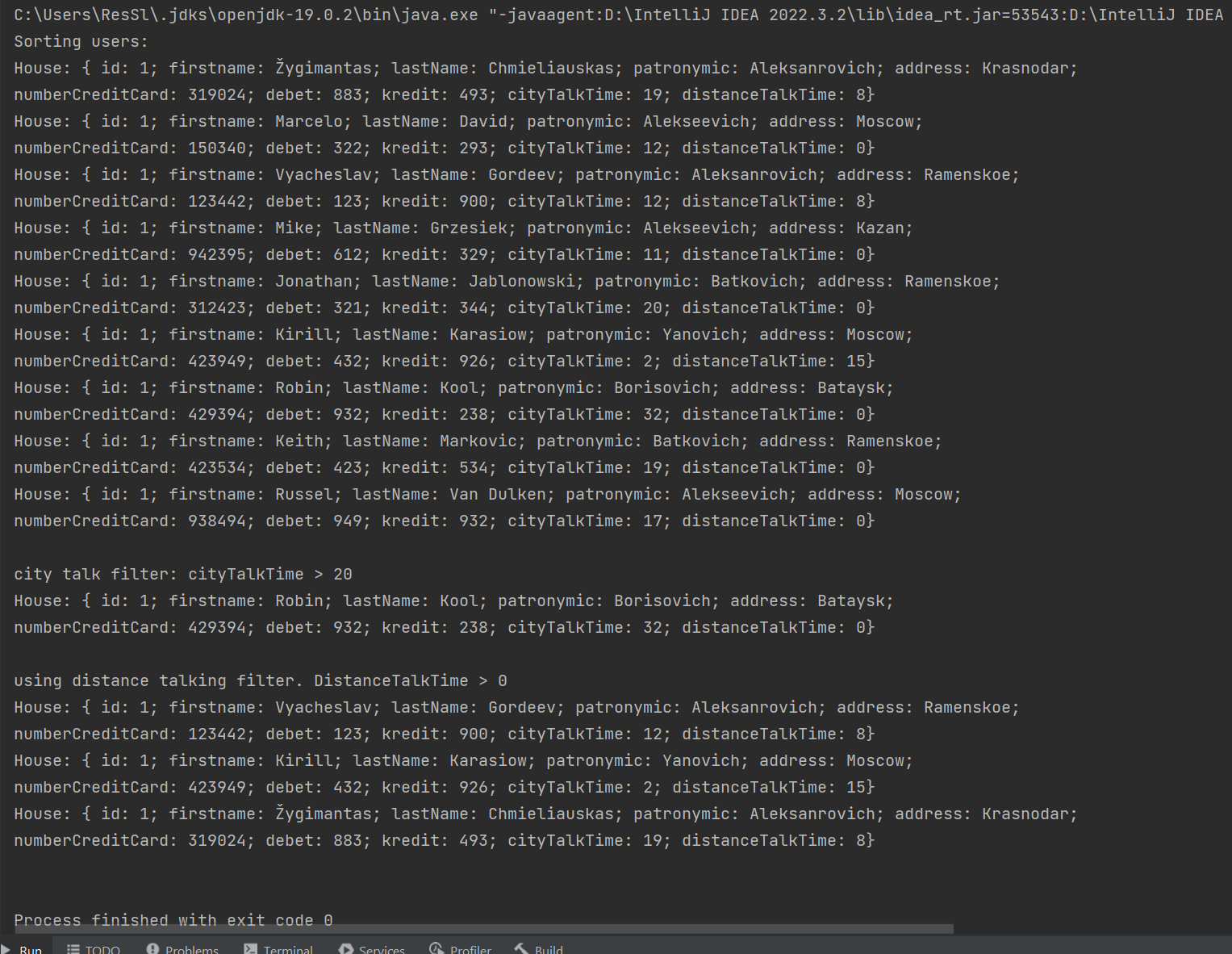


Рисунок 4 – Результат работы программы 2.2

**Задание 3.1**

Создано приложение, удовлетворяющее требованиям, приведенным в задании. Аргументирована принадлежность классу каждого создаваемого метода и корректно переопределены для каждого класса методы equals(), hashCode(), toString(). Создан объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завять, выведен на консоль цвет бутона.

Код программы представлен в листингах 11, 12, 13, 14. Результат работы программы показан на рисунке 5.

Листинг 11 – Код класса Bud

import java.util.Arrays;  
import java.util.Objects;  
  
public class Bud {  
 private Petal[] petal;  
 private String color;  
 private Boolean isBloom = true;  
  
 public Bud(Petal[] petal, String color) {  
 this.petal = petal;  
 this.color = color;  
 }  
  
 public String getColor() {  
 return color;  
 }  
 public void blooming() {  
 this.isBloom = true;  
 System.*out*.println("Бутон расцвел!");  
 }  
 public void wilt() {  
 for (int i = 0; i < this.petal.length; i++) {  
 this.petal[i].wilt();  
 }  
 this.isBloom = false;  
 System.*out*.println("Бутон завял");  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Bud bud)) return false;  
 return Arrays.*equals*(petal, bud.petal) && color.equals(bud.color) && isBloom.equals(bud.isBloom);  
 }  
  
 @Override  
 public int hashCode() {  
 int result = Objects.*hash*(color, isBloom);  
 result = 31 \* result + Arrays.*hashCode*(petal);  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 return "Bud{" +  
 "petal=" + Arrays.*toString*(petal) +  
 ", color='" + color + '\'' +  
 ", isBloom=" + isBloom +  
 '}';  
 }  
}

Листинг 12 – Код класса Petal

import java.util.Objects;  
  
public class Petal {  
 private Boolean isBloom = true;  
 public void wilt() {  
 this.isBloom = false;  
 System.*out*.println("Лепесток завял");  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Petal petal)) return false;  
 return Objects.*equals*(isBloom, petal.isBloom);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(isBloom);  
 }  
  
 @Override  
 public String toString() {  
 return "Petal{" +  
 "isBloom=" + isBloom +  
 '}';  
 }  
}

Листинг 13 – Код класса Rose

import java.util.Objects;  
  
// Создать приложение, удовлетворяющее требованиям, приведенным в задании.  
// Аргументировать принадлежность классу каждого создаваемого метода и  
// корректно переопределить для каждого класса методы equals(), hashCode(), toString().  
// Создать объект класса Роза, используя классы Лепесток, Бутон.  
// Методы: расцвести, завять, вывести на консоль цвет бутона.  
public class Rose {  
 private Bud bud;  
 private Boolean isBloom = true;  
  
 public Rose(Bud bud) {  
 this.bud = bud;  
 }  
 public void blooming() {  
 this.bud.blooming();  
 this.isBloom = true;  
 System.*out*.println("Роза расцвела!");  
 }  
 public void wilt() {  
 this.bud.wilt();  
 this.isBloom = false;  
 System.*out*.println("Роза завяла");  
 }  
 public void getColor() {  
 System.*out*.println("Цвет бутона: " + this.bud.getColor());  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Rose rose)) return false;  
 return bud.equals(rose.bud) && isBloom.equals(rose.isBloom);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(bud, isBloom);  
 }  
  
 @Override  
 public String toString() {  
 return "Rose{" +  
 "bud=" + bud +  
 ", isBloom=" + isBloom +  
 '}';  
 }  
}

Листинг 14 – Код класса Main

public class Main {  
 public static void main(String[] args) {  
 Petal petal1 = new Petal();  
 Petal petal2 = new Petal();  
 Petal petal3 = new Petal();  
 Petal petal4 = new Petal();  
  
 Petal[] petals = new Petal[4];  
 petals[0] = petal1;  
 petals[1] = petal2;  
 petals[2] = petal3;  
 petals[3] = petal4;  
  
 Bud bud = new Bud(petals, "Красный");  
 Rose rose = new Rose(bud);  
  
 rose.getColor();  
 rose.wilt();  
 rose.blooming();  
 }  
}

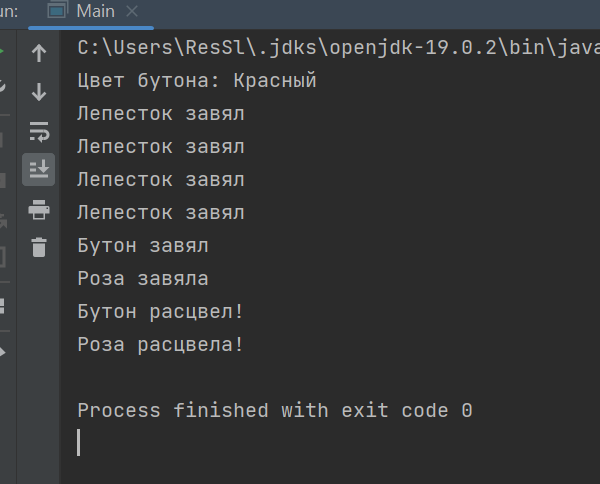


Рисунок 5 – Результат работы программы 3.1

**Задание 3.2**

Создано приложение, удовлетворяющее требованиям, приведенным в задании. Аргументирована принадлежность классу каждого создаваемого метода и корректно переопределены для каждого класса методы equals(), hashCode(), toString(). Создан объект класса Дерево, используя классы Лист. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.

Код программы представлен в листингах 15, 16, 17. Результат работы программы показан на рисунке 6.

Листинг 15 – Код класса Leaf

import java.util.Objects;  
  
public class Leaf {  
 private String color;  
 private Boolean isFall = false;  
 public Leaf(String color) {  
 this.color = color;  
 }  
  
 protected void turnYellow() {  
 this.color = "yellow";  
 System.*out*.println("Лист пожелтел");  
 }  
  
 protected void fall() {  
 this.isFall = true;  
 System.*out*.println("Лист упал");  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Leaf leaf)) return false;  
 return color.equals(leaf.color) && isFall.equals(leaf.isFall);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(color, isFall);  
 }  
  
 @Override  
 public String toString() {  
 return "Leaf{" +  
 "color='" + color + '\'' +  
 ", isFall=" + isFall +  
 '}';  
 }  
}

Листинг 16 – Код класса Tree

import java.util.Arrays;  
import java.util.Objects;  
  
public class Tree {  
 private Leaf[] leaves;  
 private Boolean isBloom = false;  
 private Boolean isFrost = false;  
 public Tree(Leaf[] leaves) {  
 this.leaves = leaves;  
 }  
 public void bloom() {  
 this.isBloom = true;  
 System.*out*.println("Дерево зацвело");  
 }  
 public void frostOver() {  
 this.isFrost = true;  
 System.*out*.println("Дерево покрылось инеем");  
 }  
  
 public void fall() {  
 for (int i = 0; i < this.leaves.length; i++) {  
 this.leaves[i].fall();  
 }  
 System.*out*.println("Все листья упали");  
 System.*out*.println();  
 }  
  
 public void turnYellow() {  
 for (int i = 0; i < this.leaves.length; i++) {  
 this.leaves[i].turnYellow();  
 }  
 System.*out*.println("Все листья пожелтели");  
 System.*out*.println();  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Tree tree)) return false;  
 return Arrays.*equals*(leaves, tree.leaves) && isBloom.equals(tree.isBloom) && isFrost.equals(tree.isFrost);  
 }  
  
 @Override  
 public int hashCode() {  
 int result = Objects.*hash*(isBloom, isFrost);  
 result = 31 \* result + Arrays.*hashCode*(leaves);  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 return "Tree{" +  
 "leaves=" + Arrays.*toString*(leaves) +  
 ", isBloom=" + isBloom +  
 ", isFrost=" + isFrost +  
 '}';  
 }  
}

Листинг 17 – Код класса Main

// Создать приложение, удовлетворяющее требованиям, приведенным в задании.  
// Аргументировать принадлежность классу каждого создаваемого метода и  
// корректно переопределить для каждого класса методы equals(), hashCode(), toString().  
  
// Создать объект класса Дерево, используя классы Лист.  
// Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.  
public class Main {  
 public static void main(String[] args) {  
 Leaf leaf1 = new Leaf("Зеленый");  
 Leaf leaf2 = new Leaf("Зеленый");  
 Leaf leaf3 = new Leaf("Красный");  
 Leaf leaf4 = new Leaf("Желтый");  
 Leaf leaf5 = new Leaf("Красный");  
 Leaf leaf6 = new Leaf("Коричневый");  
  
 Leaf[] leaves = new Leaf[6];  
 leaves[0] = leaf1;  
 leaves[1] = leaf2;  
 leaves[2] = leaf3;  
 leaves[3] = leaf4;  
 leaves[4] = leaf5;  
 leaves[5] = leaf6;  
  
 Tree tree = new Tree(leaves);  
  
 tree.bloom();  
 tree.frostOver();  
 tree.turnYellow();  
 tree.fall();  
  
 System.*out*.println(tree.toString());  
 }  
}

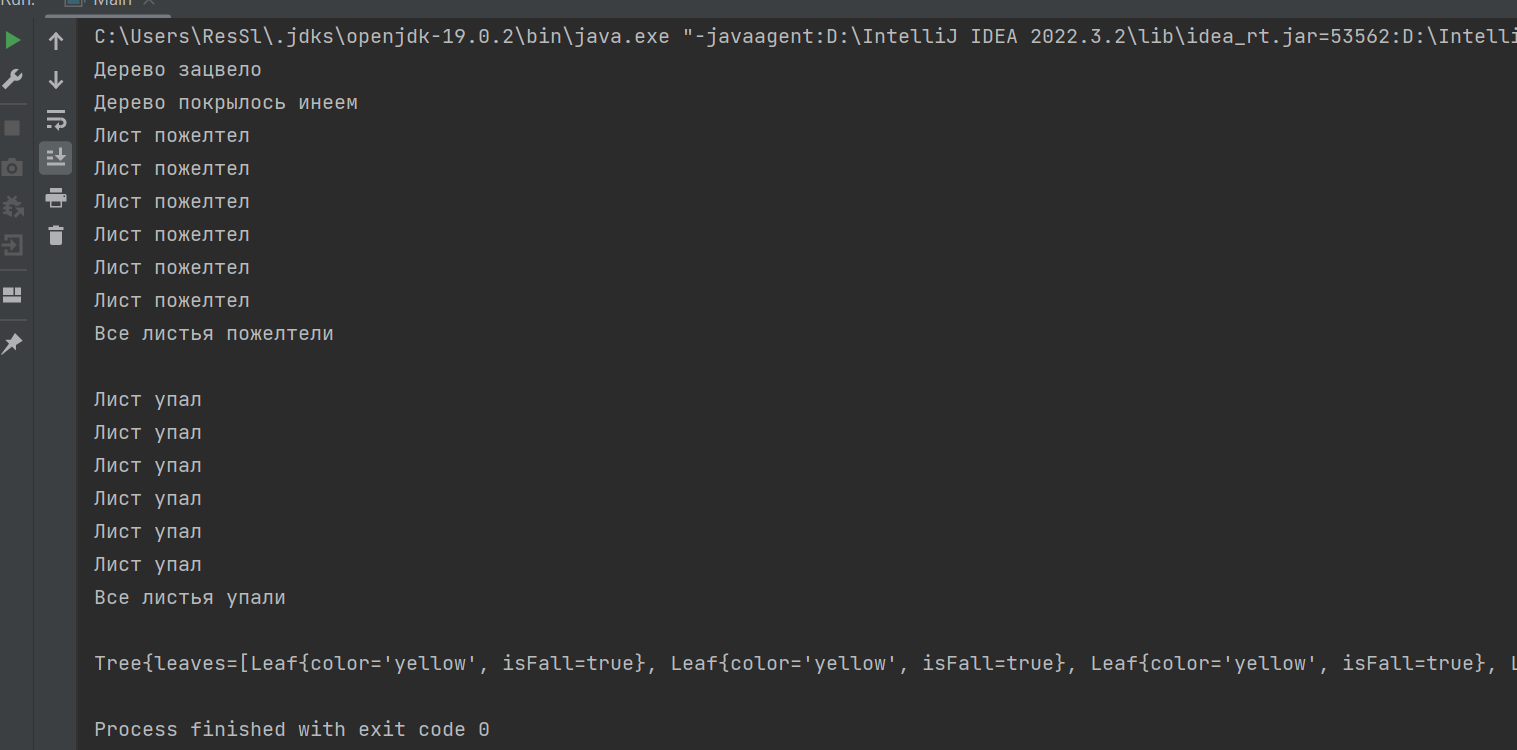


Рисунок 6 – Результат работы программы 3.2

**Задание 4.1**

Построена модель программной системы. Система Конструкторское бюро. Заказчик представляет Техническое Задание (ТЗ) на проектирование многоэтажного Дома. Конструктор регистрирует ТЗ, определяет стоимость проектирования и строительства, выставляет Заказчику Счет за проектирование и создает Бригаду Конструкторов для выполнения Проекта.

Код программы представлен в листингах 18, 19, 20, 21, 22. Результат работы программы показан на рисунке 7.

Листинг 18 – Код класса Project

import java.util.ArrayList;  
  
public class Project {  
 private TechnicalTask technicalTask;  
 private int projectCost;  
 private ConstructionTeam constructionTeam;  
  
 public Project(TechnicalTask technicalTask, ConstructionTeam constructionTeam) {  
 this.technicalTask = technicalTask;  
 this.projectCost = technicalTask.getProjectCost();  
 this.constructionTeam = constructionTeam;  
 }  
  
 public void addConstructorToTeam(Constructor constructor) {  
 constructionTeam.addConstructor(constructor);  
 }  
  
 public void removeConstructorFromTeam(Constructor constructor) {  
 constructionTeam.removeConstructor(constructor);  
 }  
  
 public int getProjectCost() {  
 return projectCost;  
 }  
 public ConstructionTeam getConstructionCompany() {  
 return constructionTeam;  
 }  
  
 public int getFullPriceOfProject() {  
 int fullPrice = getProjectCost() + constructionTeam.getFullSalaryTeam();  
 return fullPrice;  
 }  
}

Листинг 19 – Код класса TechnicalTask

public class TechnicalTask {  
 private String customerName;  
 private String projectName;  
 private int numberOfFloors;  
 private int projectCost;  
  
 public TechnicalTask(String customerName, String projectName, int numberOfFloors) {  
 this.customerName = customerName;  
 this.projectName = projectName;  
 this.numberOfFloors = numberOfFloors;  
 this.projectCost = calculateProjectCost(numberOfFloors);  
 }  
 private int calculateProjectCost(int numberOfFloors) {  
 return numberOfFloors \* 10000;  
 }  
  
 public String getCustomerName() {  
 return customerName;  
 }  
  
 public String getProjectName() {  
 return projectName;  
 }  
  
 public int getNumberOfFloors() {  
 return numberOfFloors;  
 }  
  
 public int getProjectCost() {  
 return projectCost;  
 }  
}

Листинг 20 – Код класса ConstructionTeam

public class ConstructionTeam {  
 private ArrayList<Constructor> constructors;  
  
 public ConstructionTeam() {  
 this.constructors = new ArrayList<>();  
 }  
  
 public void addConstructor(Constructor constructor) {  
 constructors.add(constructor);  
 }  
  
 public void removeConstructor(Constructor constructor) {  
 constructors.remove(constructor);  
 }  
  
 public ArrayList<Constructor> getConstructors() {  
 return constructors;  
 }  
  
 public int getConstructorCount() {  
 return constructors.size();  
 }  
  
 public int getFullSalaryTeam() {  
 int fullSalary = 0;  
 for (int i = 0; i < getConstructorCount(); i++) {  
 fullSalary += constructors.get(i).getSalary();  
 }  
 return fullSalary;  
 }  
}

Листинг 21 – Код класса Constructor

public class Constructor {  
 private String name;  
 private String specialization;  
 private int salary;  
  
 public Constructor(String name, String specialization, int salary) {  
 this.name = name;  
 this.specialization = specialization;  
 this.salary = salary;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getSpecialization() {  
 return specialization;  
 }  
  
 public int getSalary() {  
 return salary;  
 }  
}

Листинг 22 – Код класса Main

// Построить модель программной системы.  
// Система Конструкторское бюро.  
// Заказчик представляет Техническое Задание (ТЗ) на проектирование многоэтажного Дома.  
// Конструктор регистрирует ТЗ, определяет стоимость проектирования и строительства,  
// выставляет Заказчику Счет за проектирование и создает Бригаду Конструкторов для выполнения Проекта.  
  
public class Main {  
 public static void main(String[] args) {  
 TechnicalTask technicalTask = new TechnicalTask(  
 "Гордеев Вячеслав",  
 "Пивной бар",  
 5);  
  
 ConstructionTeam constructionTeam = new ConstructionTeam();  
 Constructor constructor1 = new Constructor("Эрен Йегер", "Строит стены", 1001);  
 Constructor constructor2 = new Constructor("Саске Учиха", "Прораб", 2020);  
 Constructor constructor3 = new Constructor("Наруто Узумаки", "Архитектор", 1300);  
  
 constructionTeam.addConstructor(constructor1);  
 constructionTeam.addConstructor(constructor2);  
  
 Project fullProject = new Project(technicalTask, constructionTeam);  
 fullProject.addConstructorToTeam(constructor3);  
  
 double amountPayable = fullProject.getFullPriceOfProject();  
 System.*out*.println("Стоимость проекта для " + technicalTask.getCustomerName() + ": $" + amountPayable);  
 }  
}

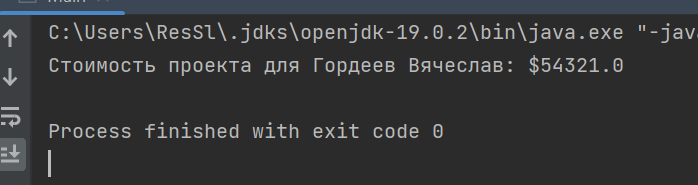


Рисунок 7 – Результат работы программы 4.1

**Задание 4.2**

Построена модель программной системы. Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.

Код программы представлен в листингах 23, 24, 25, 26. Результат работы программы показан на рисунке 8.

Листинг 11 – Код класса Client

import java.util.ArrayList;  
  
public class Client {  
 private String name;  
 private String number;  
 private double balance;  
 private boolean disabled;  
 private ArrayList<Service> services;  
 public Client(String name, String number, double balance, ArrayList<Service> services) {  
 this.name = name;  
 this.number = number;  
 this.balance = balance;  
 this.disabled = false;  
 this.services = services;  
 }  
 public void setNumber(String number) {  
 this.number = number;  
 }  
 public void addService(Service service) {  
 services.add(service);  
 }  
 public void removeService(Service service) {  
 services.remove(service);  
 }  
 public void setDisabled(boolean disabled) {  
 this.disabled = disabled;  
 }  
 public double payForServicesAndCall(double callCost) {  
 double fullCost = 0;  
 for (int i = 0; i < services.size(); i++) {  
 fullCost += services.get(i).getPrice();  
 }  
 fullCost += callCost;  
 balance -= fullCost;  
 if (balance < 0) {  
 setDisabled(true);  
 }  
 return fullCost;  
 }  
 public String getName() {  
 return name;  
 }  
 public String getNumber() {  
 return number;  
 }  
 public double getBalance() {  
 return balance;  
 }  
 public boolean isDisabled() {  
 return disabled;  
 }  
 public ArrayList<Service> getServices() {  
 return services;  
 }  
}

Листинг 12 – Код класса Service

public class Service {  
 private String name;  
 private double price;  
 public Service(String name, double price) {  
 this.name = name;  
 this.price = price;  
 }  
 public String getName() {  
 return name;  
 }  
 public double getPrice() {  
 return price;  
 }  
}

Листинг 13 – Код класса TelephoneStation

import java.util.ArrayList;  
  
public class TelephoneStation {  
 private ArrayList<Client> clients;  
 private ArrayList<Service> services;  
 private double costPerMinute;  
 public TelephoneStation(ArrayList<Client> subscribers, ArrayList<Service> services, double callRate) {  
 this.clients = subscribers;  
 this.services = services;  
 this.costPerMinute = callRate;  
 }  
 public void changeNumberForClient(Client client, String newNumber) {  
 if (clients.contains(client)) {  
 client.setNumber(newNumber);  
 } else {  
 System.*out*.println("Система не поддерживает клиента");  
 }  
 }  
 public void addServiceToClient(Client client, Service service) {  
 if (clients.contains(client) && services.contains(service)) {  
 client.addService(service);  
 } else {  
 System.*out*.println("Система не поддерживает клиента или услугу");  
 }  
 }  
 public void removeServiceFromClient(Client client, Service service) {  
 if (clients.contains(client) && services.contains(service)) {  
 client.removeService(service);  
 } else {  
 System.*out*.println("Система не поддерживает клиента или услугу");  
 }  
 }  
 public void banClient(Client client) {  
 if (clients.contains(client)) {  
 client.setDisabled(true);  
 } else {  
 System.*out*.println("Система не поддерживает клиента");  
 }  
 }  
 public void unbanClient(Client client) {  
 if (clients.contains(client)) {  
 client.setDisabled(false);  
 } else {  
 System.*out*.println("Система не поддерживает клиента");  
 }  
 }  
 public double calculateCallCost(double duration) {  
 return duration \* costPerMinute;  
 }  
 public double payClientForServicesAndCall(Client client, double duration) {  
 if (clients.contains(client)) {  
 double callCost = calculateCallCost(duration);  
 return client.payForServicesAndCall(callCost);  
 } else {  
 System.*out*.println("Система не поддерживает клиента");  
 return 0;  
 }  
 }  
}

Листинг 14 – Код класса Main

// Построить модель программной системы.  
// Система Телефонная станция.  
// Абонент оплачивает Счет за разговоры и Услуги, может попросить  
// Администратора сменить номер и отказаться от услуг.  
// Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.  
import java.util.ArrayList;  
public class Main {  
 public static void main(String[] args) {  
 Service internet = new Service("Интернет", 1000);  
 Service tv = new Service("Телевидение", 500);  
 Service phone = new Service("Телефон", 300);  
  
 ArrayList<Service> client1Services = new ArrayList<>();  
 client1Services.add(internet);  
 client1Services.add(tv);  
 Client client1 = new Client(  
 "Гордеев Вячеслав",  
 "79988129",  
 5000,  
 client1Services);  
  
 ArrayList<Service> client2Services = new ArrayList<>();  
 client2Services.add(phone);  
 Client client2 = new Client(  
 "Эрен Йегер",  
 "8888990",  
 1000,  
 client2Services);  
  
 ArrayList<Client> clients = new ArrayList<>();  
 clients.add(client1);  
 clients.add(client2);  
  
 ArrayList<Service> services = new ArrayList<>();  
 services.add(internet);  
 services.add(tv);  
 services.add(phone);  
  
 TelephoneStation telephoneStation = new TelephoneStation(clients, services, 2.5);  
  
 telephoneStation.changeNumberForClient(client1, "654321");  
  
 telephoneStation.removeServiceFromClient(client1, tv);  
  
 telephoneStation.banClient(client2);  
  
 telephoneStation.addServiceToClient(client1, phone);  
  
 double servicesAndCallCost = telephoneStation.payClientForServicesAndCall(client1, 20);  
  
 System.*out*.println("Стоимость услуг и звонков для " + client1.getName() + ": " + servicesAndCallCost);  
 }  
}

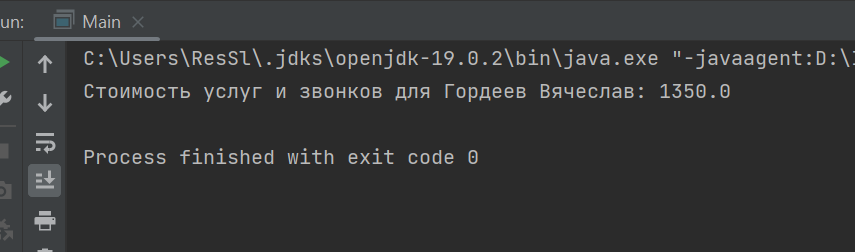


Рисунок 8 – Результат работы программы 4.2

**Вывод**: В результате выполнения лабораторной работы были получены навыки работы с классами на языке Java. Получены практические навыки работы с наследованием, полиморфизмом и композицией.