**Цель работы:**

**Задание 1. Объектно-ориентированное программирование в Java**

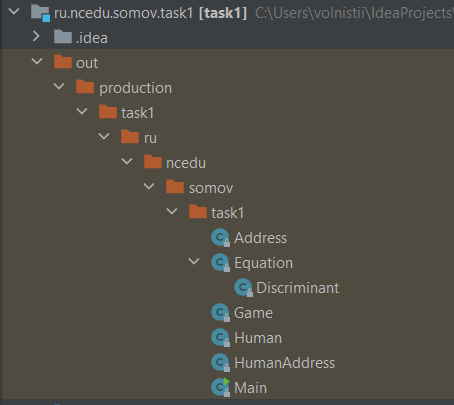
Сформировать навыки работы с основными конструкциями ООП языка Java. Освоить принципы применения базовых объектных технологий java.

**Задание. Напишите программу на языке Java, в которой продемонстрируйте работу основных принципов ООП.**

**Задание 1.1**

Разработайте класс для решения квадратных уравнений. Вычисление дискриминанта должен осуществлять вложенный класс. После компиляции объясните структуру class файлов. Проанализируйте использование вложенного класса.

Структура class файлов.



К заданию 1.1 относятся только 3 class файла *Main*, *Equation*, *Discriminant.*

Class *Main*, в котором осуществляется создание объекта класса *Equation* и вызов его методов.

Class *Equation*, в котором осуществляется решение квадратного уравнение. Данный класс имеет вложенный класс *Discriminant*.

Class *Discriminant*, в котором осуществляется нахождение дискриминанта квадратного уравнения.

Анализ использования вложенного класса.

Class *Discriminant* был реализован в качестве вложенного класса, т.к. данный класс может использоваться только в классе *Equation*.

**Задание 1.2**

Реализуйте игру в кости. Играют N игроков (компьютер в списке последний). Подкидываются одновременно К кубиков. Выигрывает тот, у кого большая сумма очков. Кто выиграл, тот и кидает первым в следующем кону. Игра идет до 7 выигрышей. Начинаете игру Вы.

Структура class файлов.

* *Main*
* *Game*

**Задание 1.3**

Напишите программу «Адрес человека». Есть сущность Человек, которая связана с сущностью Адрес. Считается, что у каждого человека есть только один адрес. Организовать массив объектов Человек (не менее 4) и по массиву:

* осуществить поиск Человека по фамилии;
* осуществить поиск человека по атрибуту адреса;
* вывести людей, родившихся между определенными датами;
* найти самого старого (молодого); − найти людей, проживающих на одной улице.

Структура class файлов.

* *Main*
* *Human*
* *Address*
* *HumanAddress*

**Код программы.**

Класс Main

package ru.ncedu.somov.task1;  
  
import java.text.ParseException;  
import java.util.Arrays;  
import java.util.Date;  
  
*/\*\*  
 \* Основной класс Main, где происходит тестирование остальных классов  
 \*/*public class Main {  
  
 public static void main(String[] args) throws ParseException {  
  
 System.*out*.println("===========================================================================================");  
 //Проверка работы задания 1(уравнение)  
 Equation equation = new Equation();  
 equation.resultEquation(20, -41, 2);  
 System.*out*.println("===========================================================================================");  
  
 //Проверка работы задания 2(игра)  
 Game game = new Game();  
 game.bonesGame(5,5);  
 System.*out*.println("===========================================================================================");  
  
 //Проверка работы задания 3(адрес человека)  
 HumanAddress humanAddress = new HumanAddress();  
  
 System.*out*.println("Поиск людей с фамилией Smith");  
 System.*out*.println(humanAddress.searchBySurname("Smith"));  
  
 System.*out*.println("Поиск людей с атрибутом адреса 2");  
 System.*out*.println(humanAddress.searchByAddressAttribute(2));  
  
 System.*out*.println("Поиск людей, который родились в диапазоне [1997.04.11;1997.05.11]");  
 System.*out*.println(humanAddress.searchByDateOfBirthRange("1997-04-11", "1997-05-11"));  
  
 System.*out*.println("Поиск самого молодого человека в списке");  
 System.*out*.println(humanAddress.searchForTheYoungestHuman());  
  
 System.*out*.println("Поиск людей, живущих на улице Myasnickaya");  
 System.*out*.println(humanAddress.searchPeopleByStreetName("Myasnickaya"));  
 System.*out*.println("===========================================================================================");  
  
 }  
}

Класс Equation и вложенный класс Discriminant (задание 1.1)

package ru.ncedu.somov.task1;  
  
*/\*\*  
 \* В классе Equation происходит рещение квадратного уравнения вида a \* (x ^ 2) + b \* x + c = 0  
 \* Также в классе Equation есть подкласс Discriminant, где происходит расчет дискриминанта уравнения  
 \*/*public class Equation {  
  
 private double result;  
  
 private class Discriminant {  
  
 private double d\_result;  
  
 private double decisionD(double a, double b, double c){  
  
 d\_result = b\*b-4\*a\*c;  
 return d\_result;  
 }  
  
 }  
  
 public void resultEquation(double a, double b, double c) {  
 Discriminant discriminant = new Discriminant();  
 discriminant.decisionD(a,b,c);  
  
 if (discriminant.d\_result > 0 ) {  
 result = (-b + Math.*sqrt*(discriminant.d\_result)) / (2 \* a);  
 System.*out*.println("x1 = " + result);  
  
 result = (-b - Math.*sqrt*(discriminant.d\_result)) / (2 \* a);  
 System.*out*.println("x2 = " + result);  
  
 } else if (discriminant.d\_result == 0) {  
 result = -b / (2 \* a);  
 System.*out*.println("x = " + result);  
  
 } else {  
 System.*out*.println("Корней нет.");  
 }  
  
 }  
  
}

Класс Game (задание 1.2)

package ru.ncedu.somov.task1;  
  
import java.util.Arrays;  
import java.util.Random;  
  
*/\*\*  
 \* В классе Game происходит реализация игры в кости.  
 \* Играют N игроков (компьютер в списке последний). Подкидываются одновременно К кубиков.  
 \* Выигрывает тот, у кого большая сумма очков.  
 \* Кто выиграл, тот и кидает первым в следующем кону(тот, кто был за ним, становится вторым и т.д.).  
 \* Игра идет до 7 выигрышей. Начинаете игру Вы (Пользователь под номером 1).  
 \*/*public class Game {  
  
 */\*\*  
 \*  
 \** ***@param*** *n - число игроков  
 \** ***@param*** *k - колличество кубиков  
 \* В методе bonesGame происходит вся реализация игры в кости.  
 \* В players\_results[i][0] записывается результат, который выбил i-ый игрок, бросив k кубиков.  
 \* В players\_results[i][1] записывается колличество побед i-ого игрока за 7 раундов  
 \*/* public void bonesGame(int n, int k) {  
 int[][] players\_results = new int[n][2];  
 int round;  
 Random rnd = new Random(System.*currentTimeMillis*());  
  
 System.*out*.println("\nПользователь - это игрок под номером 1");  
 System.*out*.println("Если максимальное колличество очков одинаково, то побеждает тот, кто первый бросил кубики. "  
 + "\nПоэтому тот, кто побеждает по очкам в текущем раунде, в следующем раунде бросает первый "  
 + "\nи отсчет начинается с него\n");  
 int winner = 0;  
 for (round = 0; round < 7; round++) {  
  
  
 int z = 0;  
  
  
 for (int i = winner; i < n; i++) {  
 System.*out*.println("Бросает кости игрок по номером " + (i + 1));  
 players\_results[i][0] = 0;  
 int j = 0;  
 while (j < k) {  
 players\_results[i][0] = players\_results[i][0] + 1 + rnd.nextInt(6);  
 j++;  
 }  
 System.*out*.println(" Сумма игрока номер " + (i + 1) + " равна " + players\_results[i][0]);  
 if (i == (n - 1)) {  
 i = -1;  
 }  
 if (z == (n - 1)) break;  
 z++;  
 }  
  
 int max = players\_results[0][0];  
 int max\_index = 0;  
  
 for (int i = 1; i < n; i++) {  
 if (players\_results[i][0] > max) {  
 max = players\_results[i][0];  
 max\_index = i;  
 }  
 }  
 players\_results[max\_index][1]++;  
 System.*out*.println("Побеждает игрок номер " + (max\_index + 1) + " поэтому он начинает следующий раунд");  
 System.*out*.println("--------------------------------------------------Конец " + (round + 1) + " раунда");  
 winner = max\_index;  
  
 }  
 System.*out*.println("Колличество побед " + (1) + "игрока = " + players\_results[0][1]);  
 System.*out*.println("Колличество побед " + (2) + "игрока = " + players\_results[1][1]);  
 System.*out*.println("Колличество побед " + (3) + "игрока = " + players\_results[2][1]);  
 System.*out*.println("Колличество побед " + (4) + "игрока = " + players\_results[3][1]);  
 System.*out*.println("Колличество побед " + (5) + "игрока = " + players\_results[4][1]);  
 }  
  
}

Класс Human (задание 1.3)

package ru.ncedu.somov.task1;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.GregorianCalendar;  
  
*/\*\*  
 \* Класс Human описывает данные о конкретном человеке.  
 \*/*public class Human {  
 private int personId;  
 private String name;  
 private String surname;  
 private Date birthday;  
 private int addressId;  
  
 */\*\*  
 \* Конструктор класса  
 \** ***@param*** *personId  
 \** ***@param*** *name  
 \** ***@param*** *surname  
 \** ***@param*** *dateOfBirth  
 \** ***@param*** *addressId  
 \** ***@throws*** *ParseException  
 \*/* public Human(int personId, String name, String surname, String dateOfBirth, int addressId) throws ParseException {  
 this.personId = personId;  
 this.name = name;  
 this.surname = surname;  
 SimpleDateFormat calendar = new SimpleDateFormat("yyyy-MM-dd");  
 this.birthday = calendar.parse(dateOfBirth);  
 this.addressId = addressId;  
 }  
  
 public int getPersonId() {  
 return personId;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public Date getBirthday() {  
 return birthday;  
 }  
  
 public int getAddressId() {  
 return addressId;  
 }  
  
 */\*\*  
 \* Переопределение метода toString  
 \** ***@return*** *\*/* @Override  
 public String toString() {  
 return "Human [HumanID: " + personId + ", Name: " + name + ", Surname: " + surname + ", DateOfBirth: " + birthday + "]";  
 }  
  
}

Класс Address (задание 1.3)

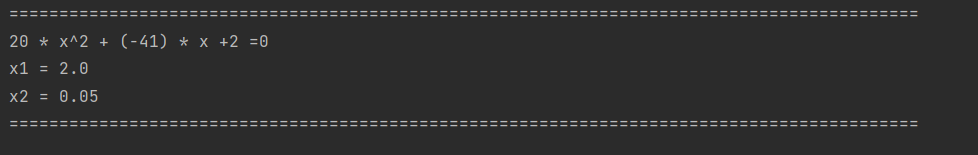
package ru.ncedu.somov.task1;  
  
*/\*\*  
 \* Класс Address описывает данные о конкретном адресе  
 \*/*public class Address {  
 private int addressId;  
 private String country;  
 private String city;  
 private String street;  
 private String domNumber;  
 private int humanId;  
  
 */\*\*  
 \* Конструктор класса Address  
 \** ***@param*** *addressId  
 \** ***@param*** *country  
 \** ***@param*** *city  
 \** ***@param*** *street  
 \** ***@param*** *domNumber  
 \** ***@param*** *humanId  
 \*/* public Address(int addressId, String country, String city, String street, String domNumber, int humanId) {  
 this.addressId =addressId;  
 this.country = country;  
 this.city = city;  
 this.street = street;  
 this.domNumber = domNumber;  
 this.humanId = humanId;  
 }  
  
 public int getAddressId() {  
 return addressId;  
 }  
  
 public String getCountry() {  
 return country;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public String getStreet() {  
 return street;  
 }  
  
 public String getDomNumber() {  
 return domNumber;  
 }  
  
 public int getHumanId() {  
 return humanId;  
 }  
}

Класс HumanAddress (задание 1.3)

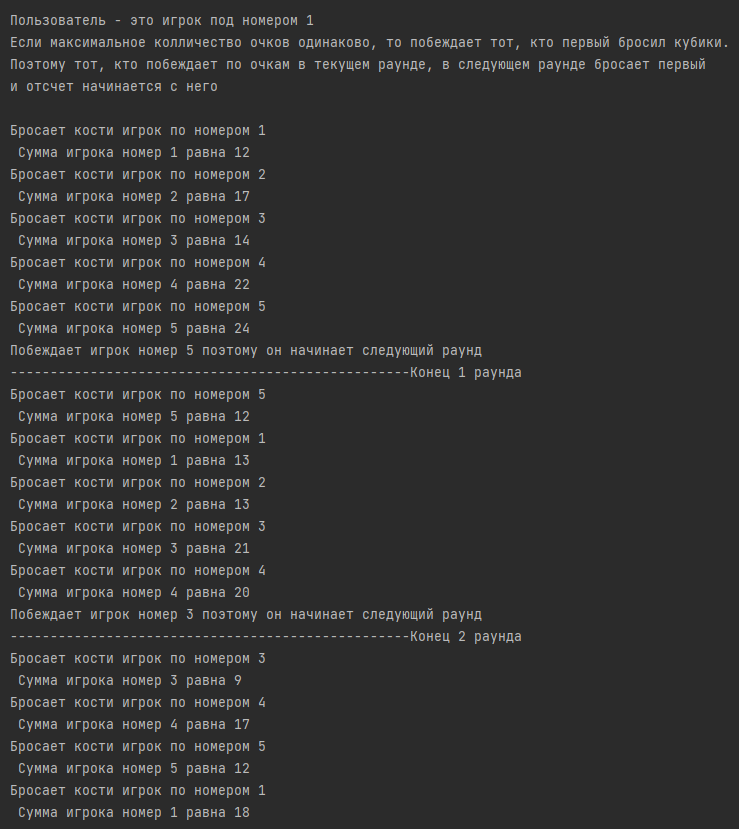
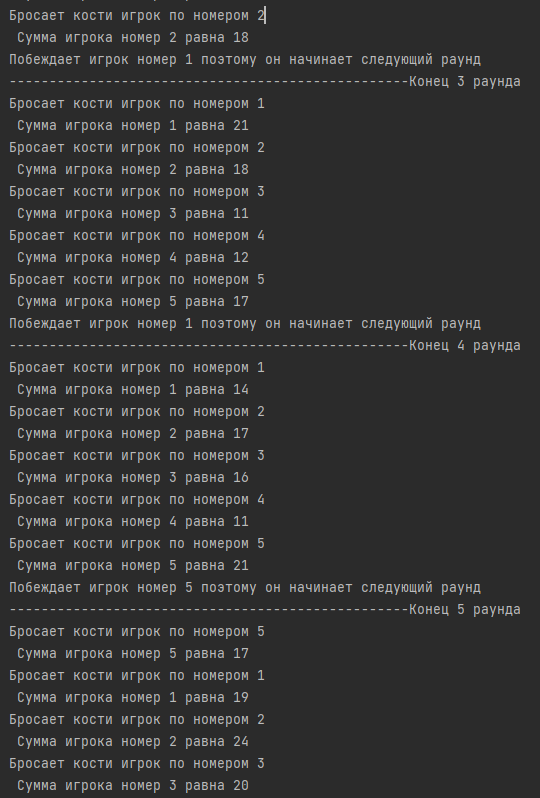
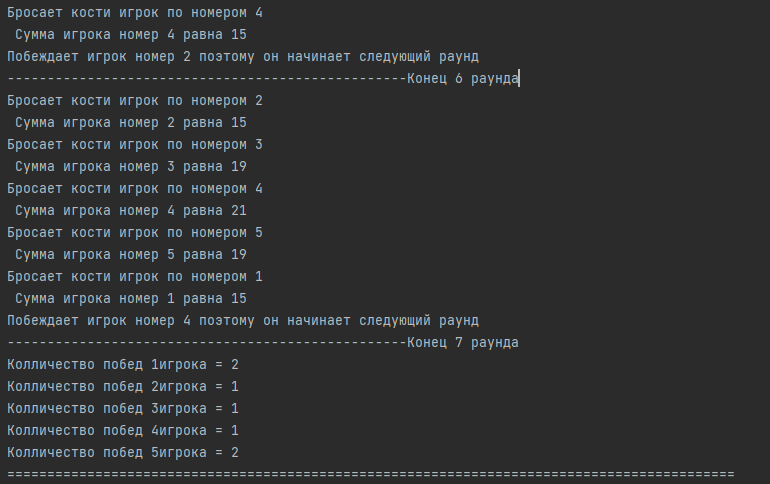
package ru.ncedu.somov.task1;  
  
import java.text.ParseException;  
import java.util.ArrayList;  
import java.util.Date;  
import java.text.SimpleDateFormat;  
  
*/\*\*  
 \* Класс HumanAddress, при помощи котрого осуществляется поиск по объектам классов Human и Address  
 \*/*public class HumanAddress {  
 private ArrayList<Human> human = new ArrayList<Human>();  
 private ArrayList<Address> address = new ArrayList<Address>();  
  
 */\*\*  
 \* Конструктор класса HumanAddress, при помощи котрого, при создании объекта класса HumanAddress,  
 \* создаются объекты классов Human и Address и добавляются в ArrayList  
 \** ***@throws*** *ParseException  
 \*/* public HumanAddress() throws ParseException {  
 this.human.add(new Human(1, "Jack", "Smith", "1997-04-12", 1));  
 this.human.add(new Human(2, "Mike", "Smith", "1997-05-07", 2));  
 this.human.add(new Human(3, "Sam", "Rooney", "1990-05-08", 3));  
 this.human.add(new Human(4, "Harry", "Potter", "1992-06-02", 4));  
  
 this.address.add(new Address(1, "Russia", "Moscow", "Myasnickaya", "2/b", 1));  
 this.address.add(new Address(2, "Russia", "Moscow", "Mira", "44/4", 2));  
 this.address.add(new Address(3, "Russia", "Moscow", "Snejnaya", "16k4", 3));  
 this.address.add(new Address(4, "Russia", "Moscow", "Myasnickaya", "8/4", 4));  
  
 }  
  
 */\*\*  
 \* Метод searchBySurname реализовывает поиск объекта класса Human по свойству Surname  
 \** ***@param*** *surname  
 \** ***@return*** *\*/* public ArrayList<String> searchBySurname(String surname) {  
  
 ArrayList<String> ourHuman = new ArrayList<String>();  
  
  
 for (int i = 0; i < this.human.size(); i++) {  
 if (surname == this.human.get(i).getSurname()) {  
 ourHuman.add("\n " + this.human.get(i));  
 }  
 }  
 return ourHuman;  
 }  
  
 */\*\*  
 \* Метод searchByAddressAttribute реализовывает поиск объекта класса Human по свойству addressId  
 \** ***@param*** *addressId  
 \** ***@return*** *\*/* public ArrayList<String> searchByAddressAttribute(int addressId) {  
  
 ArrayList<String> ourHuman = new ArrayList<String>();  
  
  
 for (int i = 0; i < this.human.size(); i++) {  
 if (addressId == this.human.get(i).getAddressId()) {  
 ourHuman.add("\n " + this.human.get(i));  
 }  
 }  
 return ourHuman;  
 }  
  
 */\*\*  
 \* Метод searchByDateOfBirthRange реализовывает поиск объекта класса Human по свойству birthday,  
 \* которое находится в диапазоне [firstDate;secondDate]  
 \** ***@param*** *firstDate  
 \** ***@param*** *secondDate  
 \** ***@return*** *\** ***@throws*** *ParseException  
 \*/* public ArrayList<String> searchByDateOfBirthRange(String firstDate, String secondDate) throws ParseException {  
  
 ArrayList<String> ourHuman = new ArrayList<String>();  
 Date ourFirstDate;  
 Date ourSecondDate;  
  
 SimpleDateFormat calendar = new SimpleDateFormat("yyyy-MM-dd");  
 ourFirstDate = calendar.parse(firstDate);  
 ourSecondDate = calendar.parse(secondDate);  
  
  
 for (int i = 0; i < this.human.size(); i++) {  
 if (ourFirstDate.before(this.human.get(i).getBirthday())) {  
 if (this.human.get(i).getBirthday().before(ourSecondDate)) {  
 ourHuman.add("\n " + this.human.get(i));  
 }  
 }  
 }  
 return ourHuman;  
 }  
  
 */\*\*  
 \* Метод searchForTheYoungestHuman реализовывет поиск объекта класса Human,  
 \* который имеет наибольшее значение в свойстве birthday (наименьший возраст)  
 \** ***@return*** *\*/* public ArrayList<String> searchForTheYoungestHuman() {  
  
 ArrayList<String> ourHuman = new ArrayList<String>();  
 Date youngestHuman = this.human.get(0).getBirthday();  
 ourHuman.add("\n " + this.human.get(0));  
  
 for (int i = 1; i < this.human.size(); i++) {  
 if (this.human.get(i).getBirthday().after(youngestHuman)) {  
 youngestHuman = this.human.get(i).getBirthday();  
 ourHuman.set(0, "\n" + this.human.get(i));  
 }  
 }  
 return ourHuman;  
 }  
  
 */\*\*  
 \* Метод searchPeopleByStreetName реализовывается поиск объектов по двум классам Human и Address.  
 \* Сначала выполняетя поиск свойства Street класса Address.  
 \* Затем берет свойство humanId объекта класса Street.  
 \* Далее выполняется поиск объекта класса Human по свойству personId (address.humanId == human.personId)  
 \** ***@param*** *street  
 \** ***@return*** *\*/* public ArrayList<String> searchPeopleByStreetName(String street) {  
  
 ArrayList<String> ourHuman = new ArrayList<String>();  
  
 for (int i = 0; i < this.address.size(); i++) {  
  
 if (address.get(i).getStreet() == street) {  
 int humanId = address.get(i).getHumanId();  
  
 for (int j = 0; j < this.human.size(); j++) {  
 if(human.get(j).getPersonId() == humanId) {  
 ourHuman.add("\n " + this.human.get(j));  
 }  
 }  
 }  
 }  
 return ourHuman;  
  
 }  
}

**Демонстрация работы программы.**

Задание 1.1.

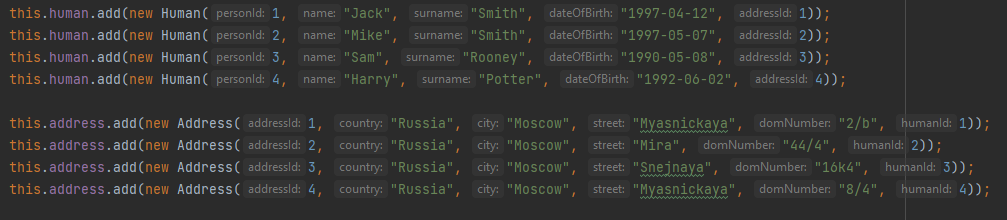


Задание 1.2

Задание 1.3

Данные об людях и их адресах:



Вывод.

