NetCracker Learning center

Учебное практическое задание № 1

Задание 1. Объектно-ориентированное программиорвание в Java

выполнил студент

Яромир Водзяновский

Долгопрудный, 2021

Код лежит на GitHub.

1 Квадратное уравнение

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

```java
import java.util.Scanner;

public class Solver {

    class Discr {

        public double discr_calc(int a, int b, int c) {
            double discr = b*b - 4*a*c;
            return discr;
        }
    }

    public static double[] answer(int a, int b, double discr) {
        double[] res = new double[2];
        for (int i=0; i<2; i++) {
            res[i] = (-b + Math.pow(-1, i) * Math.sqrt(discr)) / (2 * a);
        }
        return res;
    }

    public static void main(String[] args) {

        int[] coeffs;
        double[] answer = new double[2];

        Scanner in = new Scanner(System.in);

        coeffs = new int [3];
        for (int i=0; i<3; i++) {
            System.out.print("Enter coeff " + i + " : ");
            coeffs[i] = in.nextInt();
        }
        in.close();

        int a = coeffs[0];
        int b = coeffs[1];
        int c = coeffs[2];

        Solver solver = new Solver();
        Discr discr = solver.new Discr();

        double discriminante = discr.discr_calc(a, b, c);

        if (a == 0 && b == 0 && c == 0) {
        System.out.println("The Answer: Infinity amount of solutions!");

        } else {
            if (discriminante < 0) {
                System.out.println("The Answer: No solution in Real numbers!");
            } else {
                if (a == 0) {
                    double answer_linear = -c / b;
                    System.out.println("The Answer: " + answer_linear);

                } if (a == 0 && b == 0 && c != 0) {
                    System.out.println("The Answer: No solution");
```

```
                } if (a != 0) {
                answer = answer(a, b, discriminante);
                System.out.println("The Answer: " + answer[0] + " and " + answer[1]);
                }
            }
        }
    }
}
```

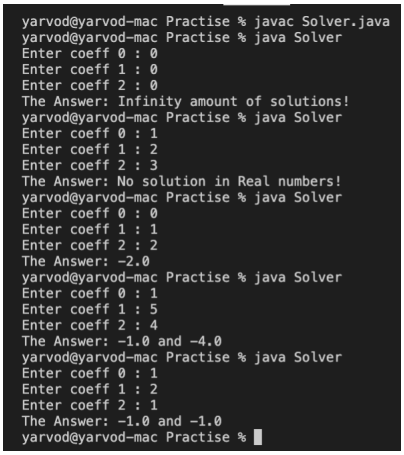


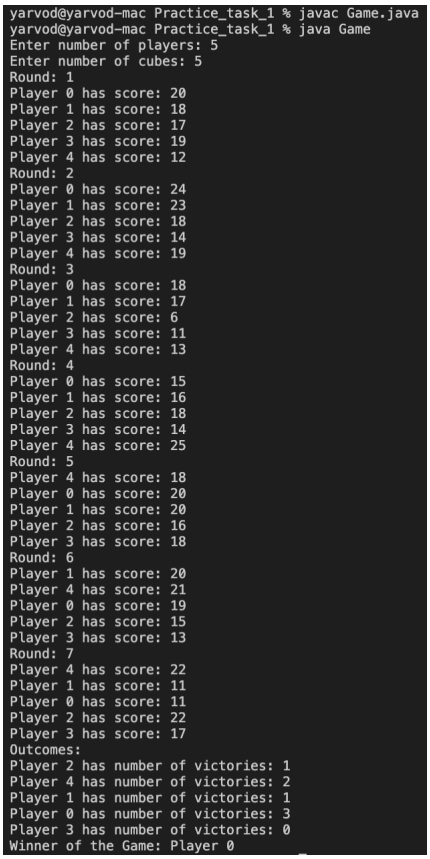
Рис. 1: Демонастрация работы решателя квадратного уравнения

2 Игра в кости

Цель: Реализуйте игру в кости. Играют N игроков (компьютер в списке последний). Подкидываются одновременно K 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.

```
import java.util.Scanner;

public class Game {

    public static int Play(int K) {
        int score = 0;
        for (int j=0; j<K; j++) {
            score += (int) (Math.random() * 6 + 1);
        }
        return score;
    }

    public static int[] maxArray(int[] Array) {
        int index = 0;
        int max = 0;
        int[] maxArray = new int[2];
        for (int i = 0; i<Array.length; i++) {
            if (max <= Array[i]) {
                max = Array[i];
                index = i;
            }
        }
        maxArray[0] = index;
        maxArray[1] = max;
        return maxArray;
    }

    public static int[][] Substitute(int[][] PlayerList, int maxScoreIndex) {
        int[] save_winner = PlayerList[maxScoreIndex];
        for (int k=0; k<maxScoreIndex; k++) {
```

```java
            PlayerList[maxScoreIndex - k] = PlayerList[maxScoreIndex - k-1];
        }
        PlayerList[0] = save_winner;
        return PlayerList;
    }

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);
        int N;
        int K;
        int maxScoreIndex;

        System.out.print("Enter number of players: ");
        N = in.nextInt();

        System.out.print("Enter number of cubes: ");
        K = in.nextInt();
        in.close();

        int[] Scores = new int[N];
        int[][] PlayerList = new int[N][2];
        int[] TotalScores = new int[N];
        int WinnerIndex = 0;


        for (int i=0; i<N; i++) {
            PlayerList[i][0] = i;
            PlayerList[i][1] = 0;
        }

        for (int round=0; round<7; round++) {

            System.out.println("Round: " + (round+1));
            for (int player_num=0; player_num<N; player_num++) {
                Scores[player_num] = Play(K);
            }
            maxScoreIndex = maxArray(Scores)[0];

            PlayerList[maxScoreIndex][1] += 1;

            for (int k=0; k<N; k++) {
                System.out.println("Player " + PlayerList[k][0] + " has score: " + Scores[k]);
            }

            PlayerList = Substitute(PlayerList, maxScoreIndex);
        }

        System.out.println("Outcomes: ");

        for (int k=0; k<N; k++) {
            System.out.println("Player " + PlayerList[k][0] + " has number of victories: " + PlayerList[k][1]);
        }

        for (int i=0; i<N; i++) {
            TotalScores[i] = PlayerList[i][1];
        }

        WinnerIndex = maxArray(TotalScores)[0];

        System.out.println("Winner of the Game: Player " + PlayerList[WinnerIndex][0]);
    }
}
```



```
yarvod@yarvod-mac Practice_task_1 % javac Game.java
yarvod@yarvod-mac Practice_task_1 % java Game
Enter number of players: 5
Enter number of cubes: 5
Round: 1
Player 0 has score: 20
Player 1 has score: 18
Player 2 has score: 17
Player 3 has score: 19
Player 4 has score: 12
Round: 2
Player 0 has score: 24
Player 1 has score: 23
Player 2 has score: 18
Player 3 has score: 14
Player 4 has score: 19
Round: 3
Player 0 has score: 18
Player 1 has score: 17
Player 2 has score: 6
Player 3 has score: 11
Player 4 has score: 13
Round: 4
Player 0 has score: 15
Player 1 has score: 16
Player 2 has score: 18
Player 3 has score: 14
Player 4 has score: 25
Round: 5
Player 4 has score: 18
Player 0 has score: 20
Player 1 has score: 20
Player 2 has score: 16
Player 3 has score: 18
Round: 6
Player 1 has score: 20
Player 4 has score: 21
Player 0 has score: 19
Player 2 has score: 15
Player 3 has score: 13
Round: 7
Player 4 has score: 22
Player 1 has score: 11
Player 0 has score: 11
Player 2 has score: 22
Player 3 has score: 17
Outcomes:
Player 2 has number of victories: 1
Player 4 has number of victories: 2
Player 1 has number of victories: 1
Player 0 has number of victories: 3
Player 3 has number of victories: 0
Winner of the Game: Player 0
```

Рис. 2: Демонстрация работы игры в кости

3 Адрес человека

Цель: 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.

```java
import java.time.LocalDate;
import java.util.Scanner;

public class HumanAdress {

    class Human {
        String name, surname;
        int age, day_birth, month_birth, year_birth;

        public Human(String name, String surname, int age, int day_birth, int month_birth, int year_birth) {
            this.name = name;
            this.surname = surname;
            this.age = age;
            this.day_birth = day_birth;
            this.month_birth = month_birth;
            this.year_birth = year_birth;
        }
```

```
    class Adress {
        int flat, house;
        String street, city, country;

        public Adress(int flat, int house, String street, String city, String country) {
            this.flat = flat;
            this.house = house;
            this.street = street;
            this.city = city;
            this.country = country;
        }
    }
}

static void print_human(Human human) {
    System.out.println("Human name/surname: " + human.name + " " + human.surname + ", date of birth: " + LocalDate
}

static void find_by_surname(String surname, Human[] humans) {
    int k = 0;
    for (int i=0; i<4; i++) {
        if (humans[i].surname == surname) {
            print_human(humans[i]);
            k++;
        }
    }
    if (k == 0) {
        System.out.println("There is no such person here");
    }
}

static void find_age(String args, Human[] humans) {
    int min = 100;
    int max = 0;
    int index = 0;
    if (args == "youngest") {
        for (int i=0; i<4; i++) {
            if (humans[i].age < min) {
                min = humans[i].age;
                index = i;
            }
        }
        System.out.print("The youngest: ");
        print_human(humans[index]);
    } if (args == "oldest") {
        for (int i=0; i<4; i++) {
            if (humans[i].age > max) {
                max = humans[i].age;
                index = i;
            }
        }
        System.out.print("The oldest: ");
        print_human(humans[index]);
    }
}


public static void main(String[] args) {

    HumanAdress human_adress = new HumanAdress();
    Human[] humans = new Human[4];

    humans[0] = human_adress.new Human("Yulya", "Prokhorova", 20, 11, 7, 2001);
    humans[1] = human_adress.new Human("Artem", "Atepalikhin", 21, 29, 4, 2000);
    humans[2] = human_adress.new Human("Alexandrov", "Alexandr", 16, 20, 4, 2005);
```

```
        humans[3] = human_adress.new Human("Diana", "Chikan", 19, 4, 8, 2001);

        System.out.print("Enter a surname to find: ");
        Scanner in = new Scanner(System.in);
        String entered_surname = in.next();

        find_by_surname(entered_surname, humans);

        System.out.println("youngest or oldest human you want to know? ");
        String entered_answer = in.next();
        find_age(entered_answer, humans);

        in.close();
    }
}
```