Отчет по учебному практическому заданию 1

1. **Решение квадратных уравнений**

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

Комментарии к коду:
Квадратные уравнения описаны в главном классе *QuadraticEquation*, принимающий в качестве входных параметров коэффициенты *a*, *b* и *c*. Основная характеристика квадратных уравнений - дискриминант, он реализован статическим вложенным классом *Discriminant*. Так как класс статический у нас нет необходимости создавать объект класса *Discriminant* для того чтобы вычислить его для очередного уравнения. С целью избежания ошибок, связанных с представлением чисел на ЭВМ, используется метод *isZero* для сравнения вещественных чисел с нулем.

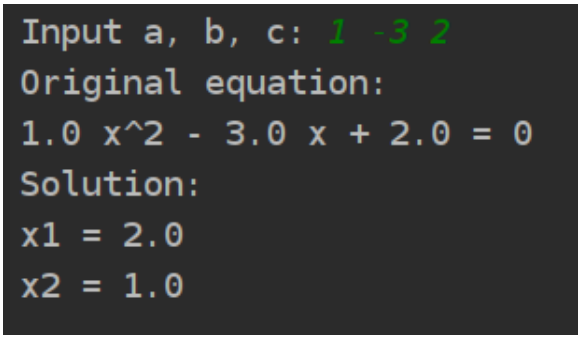
Программный код:

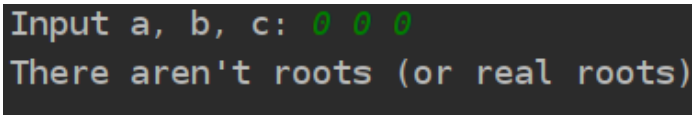
```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import static java.lang.Math.abs;
import static java.lang.Math.sqrt;

public class QuadraticEquation {
    private double a, b, c;
    private List<Double> solution;

    public QuadraticEquation(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
        this.solution = null;
    }

    public static class Discriminant {
        public static double getDiscriminant(double a, double b, double c) {
            return b * b - 4 * a * c;
        }
    }

    // checking for equality to 0. With a comparison with the eps = 10e-12.
    public boolean isZero(double x) {
        double eps = 10e-12;
        return abs(x) < eps;
    }
```

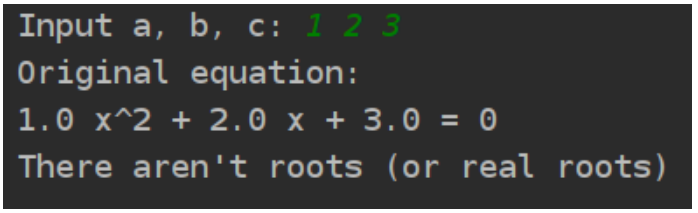
```
    public void getInfo() {
        System.out.print("Original equation:\n" + a + " x^2");
        if (b < 0) {
            System.out.print(" - " + abs(b) + " x");
        } else if (b > 0) {
            System.out.print(" + " + abs(b) + " x");
        }
        if (c < 0) {
            System.out.print(" - " + abs(c));
        } else if (c > 0) {
            System.out.print(" + " + abs(c));
        }
        System.out.println(" = 0");
        if (solution == null) {
            System.out.println("The system has not been resolved yet.(Please run getSolve
first)");
            return;
        }
        if (solution.isEmpty()) {
            System.out.println("There aren't roots (or real roots)");
        } else if (solution.size() == 1) {
            System.out.println("Solution:\nx = " + solution.get(0));
        } else if (solution.size() == 2) {
            System.out.println("Solution:\nx1 = " + solution.get(0) + "\nx2 = " + solution.get(1));
        }
        ;
    }

    public void getSolve() {
        if (solution != null) {
            return;
        }   // If current equation has already been solved there is no need to solve it again
        double tmp = 0.;
        solution = new ArrayList<>();
        if (isZero(a)) a = 0.;
        if (isZero(b)) b = 0.;
        if (isZero(c)) c = 0.;
        if (a == 0. && b == 0.) {
            return;
        } else if (a == 0.) {
            tmp = -c / b;
            if (isZero(tmp)) tmp = 0.;
            solution.add(tmp);
            return;
        }
        double d = Discriminant.getDiscriminant(a, b, c);
        if (isZero(d)) d = 0.;
```

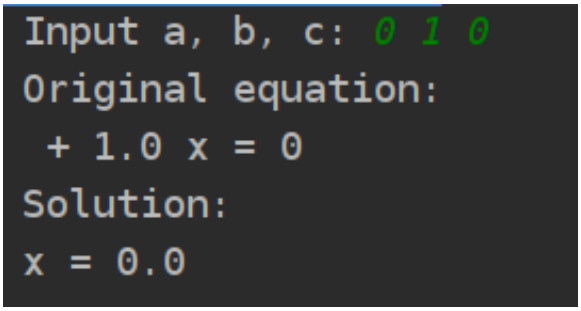
```
        if (d == 0.) {
            tmp = -b / (2. * a);
            if (isZero(tmp)) tmp = 0.;
            solution.add(tmp);
        } else if (d > 0.) {
            tmp = (-b + sqrt(d)) / (2. * a);
            if (isZero(tmp)) tmp = 0.;
            solution.add(tmp);
            tmp = (-b - sqrt(d)) / (2. * a);
            if (isZero(tmp)) tmp = 0.;
            solution.add(tmp);
        }
    }

    public static void main(String[] args) throws Exception {

        Scanner in = new Scanner(System.in);
        double a = 0., b = 0., c = 0.;
        try {
            System.out.print("Input a, b, c: ");
            a = in.nextDouble();
            b = in.nextDouble();
            c = in.nextDouble();
        } catch (Exception e) {
            System.out.println(e.getMessage());
            return;
        } finally {
            try {
                in.close();
            } catch (Exception e) {
                System.out.println(e.getMessage());
                return;
            }
        }

        QuadraticEquation equ1 = new QuadraticEquation(a, b, c)
        equ1.getSolve();
        equ1.getInfo();
    }
}
```

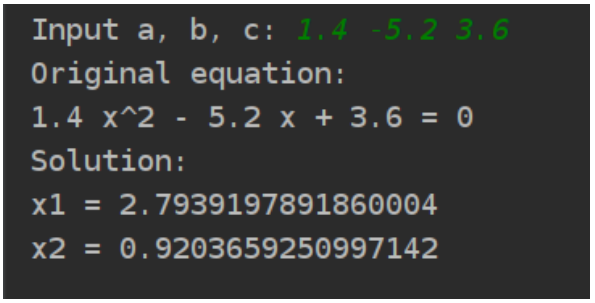
Скриншоты результатов работы











2. **Игра в кости**

Задание: 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.

Комментарии к коду:
Игра реализована в классе *DiceGame* с вложенным классом игроков *Player*, в котором осуществлена перегрузка метода *compareTo* (из интерфейса *Comparable*) для сравнения по количеству выпавших очков на костях.

Программный код:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class DiceGame {
   private int numOfPlayers;
```

```
private List<Player> players;
private int numOfDice;
private int gamesCounter;

public DiceGame(List<String> players, int numOfDice) {
    this.gamesCounter = 0;
    this.numOfDice = numOfDice;
    this.numOfPlayers = players.size();
    this.players = new ArrayList<>();
    System.out.println("Number of dine: " + numOfDice);
    for (int i = 0; i < this.numOfPlayers; i++) {
        this.players.add(new Player(players.get(i)));
    }
}

public int rollDice() {
    int score = 0;
    for (int i = 0; i < numOfDice; i++) {
        score += (int) (Math.random() * 6) + 1;
    }
    return score;
}

public void printInfo() {
    System.out.println("**********Round " + gamesCounter + "**********");
    for (int i = 0; i < numOfPlayers; i++) {
        System.out.println(players.get(i).name + ": \n" +
                "   current score: " + players.get(i).score +
                "\n   wins: " + players.get(i).winsCounter);
    }
    System.out.println("****************************");
}

public void startGame() {
    boolean f = true;
    while (f) {
        gamesCounter++;
        for (int i = 0; i < numOfPlayers; i++) {
            if (players.get(i).name == "You") {
                System.out.println("Your turn! Roll the dice.(Press ENTER, for example)");
                Scanner scanner = new Scanner(System.in);
                String tap = scanner.nextLine();
                players.get(i).setScore(rollDice());
                System.out.println("Your score is: " + players.get(i).score);
            } else {
                players.get(i).setScore(rollDice());
                //System.out.println(players.get(i).name + "'s score is: " + players.get(i).score);
            }
```

```
        }
        Collections.sort(players);
        System.out.println("Winer of this round is " + players.get(0).name);
        players.get(0).incWinsCounter();
        printInfo();
        if (players.get(0).winsCounter == 7) {
            System.out.println("End game. Winer is: " + players.get(0).name);
            f = false;
        }
    }

}

public static void main(String[] args) {
    List<String> players = new ArrayList<>();
    players.add("You");
    players.add("Tom");
    players.add("John");
    players.add("Bot");
    DiceGame game1 = new DiceGame(players, 3);
    game1.startGame();
}


class Player implements Comparable<Player> {
    private String name;
    private int score;
    private int winsCounter;

    public Player(String name) {
        this.name = name;
        score = 0;
        winsCounter = 0;
    }

    public String getName() {
        return this.name;
    }

    public int getScore() {
        return this.score;
    }

    public int getWinsCounter() {
        return this.winsCounter;
    }

    public void setScore(int score) {
```

```
            this.score = score;
        }

        public void incWinsCounter() {
            this.winsCounter++;
        }

        @Override
        public int compareTo(Player player) {
            if (player.score < this.score) {
                return -1;
            } else if (player.score > this.score) {
                return 1;
            } else {
                return 0;
            }
        }
    }
}
```

Скриншоты результатов работы