Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 16

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Инкапсуляция»

Выполнил: ст. гр. ТУУ-111

Грачева Н.С.

Вариант №5

Проверил: к.т.н., доц. Сафронов А.И.

Москва – 2024 г.

## 1. Цель работы

Разработать приложение с инкапсулированным классом.

## 2. Формулировка задачи

Вариант 05. Создайте класс «квадратное уравнение». Все поля класса инкапсулированы в классе. В качестве исходных данных вводятся коэффициенты квадратного уравнения: a, b, c. Значения коэффициентов фиксируются при создании экземпляра класса и далее экземпляр может только сообщать их, не позволяя изменять (get). В программном обеспечении предусмотрена подача команды на решение квадратного уравнения по нажатии на экранную кнопку. В результате выполнения команды с строку статусного сообщения выводится один из исходов: «Решено» или «Не решено» (для отрицательных значений дискриминантов). После получения положительного исхода открывается возможность для просмотра значения дискриминанта (get), значений каждого из корней (get). Иметь возможность хранения квадратных уравнений с различными коэффициентами в списке. Иметь возможность очистки списка ото всех ранее введённых квадратных уравнений по нажатии на соответствующую экранную. Иметь возможность очистки решения для каждого для каждого квадратного уравнения по нажатии на соответствующую экранную кнопку. Реализовать расчёт / очистку результата квадратного уравнения через выставление логического признака (get; set), управляющего возможностью передачи свойствами (properties) расчётных значений дискриминанта и корней.

## 3. Сети Петри запрограммированного технологического процесса

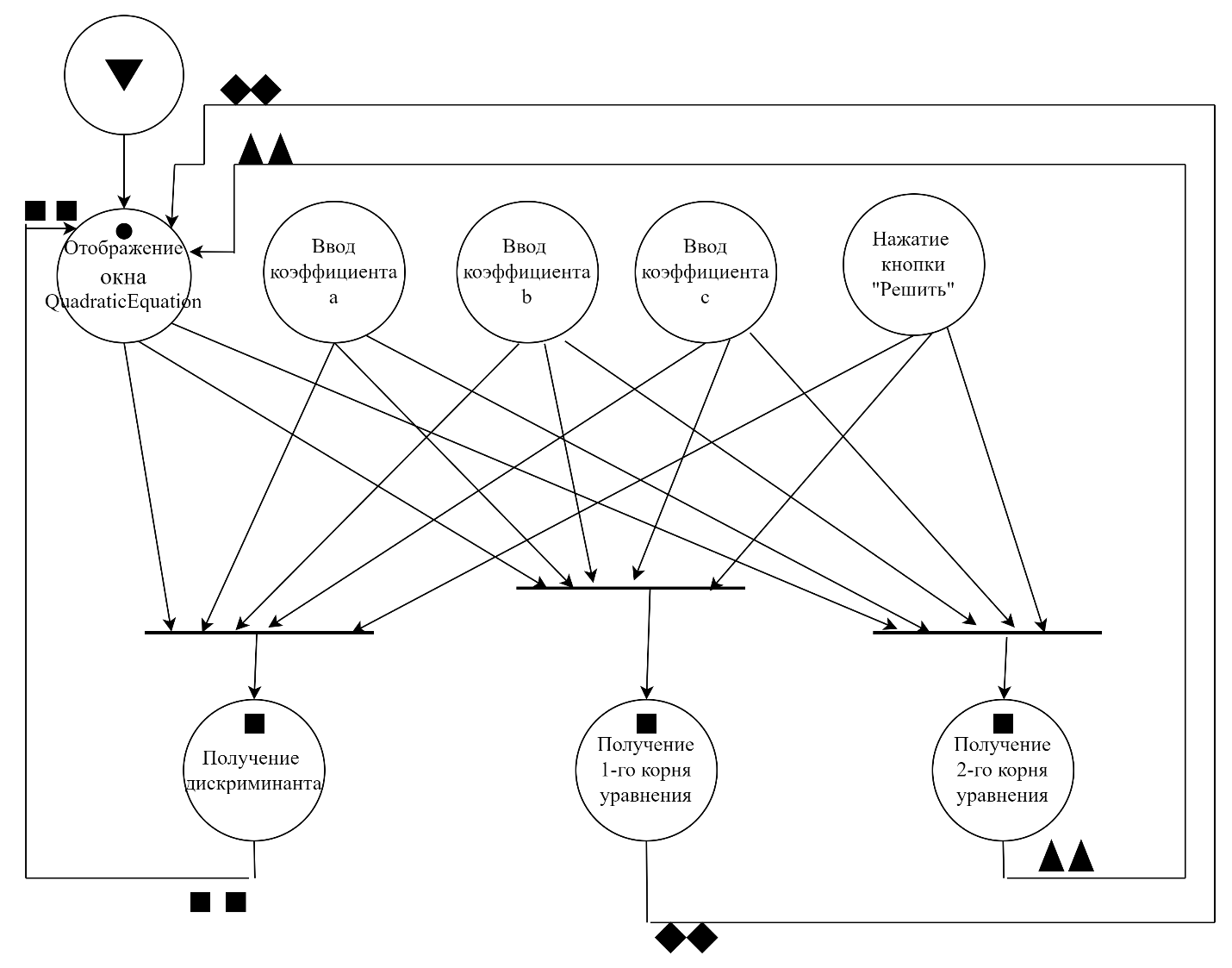


Рисунок 1 - Сеть Петри для случая, когда все коэффициенты введены

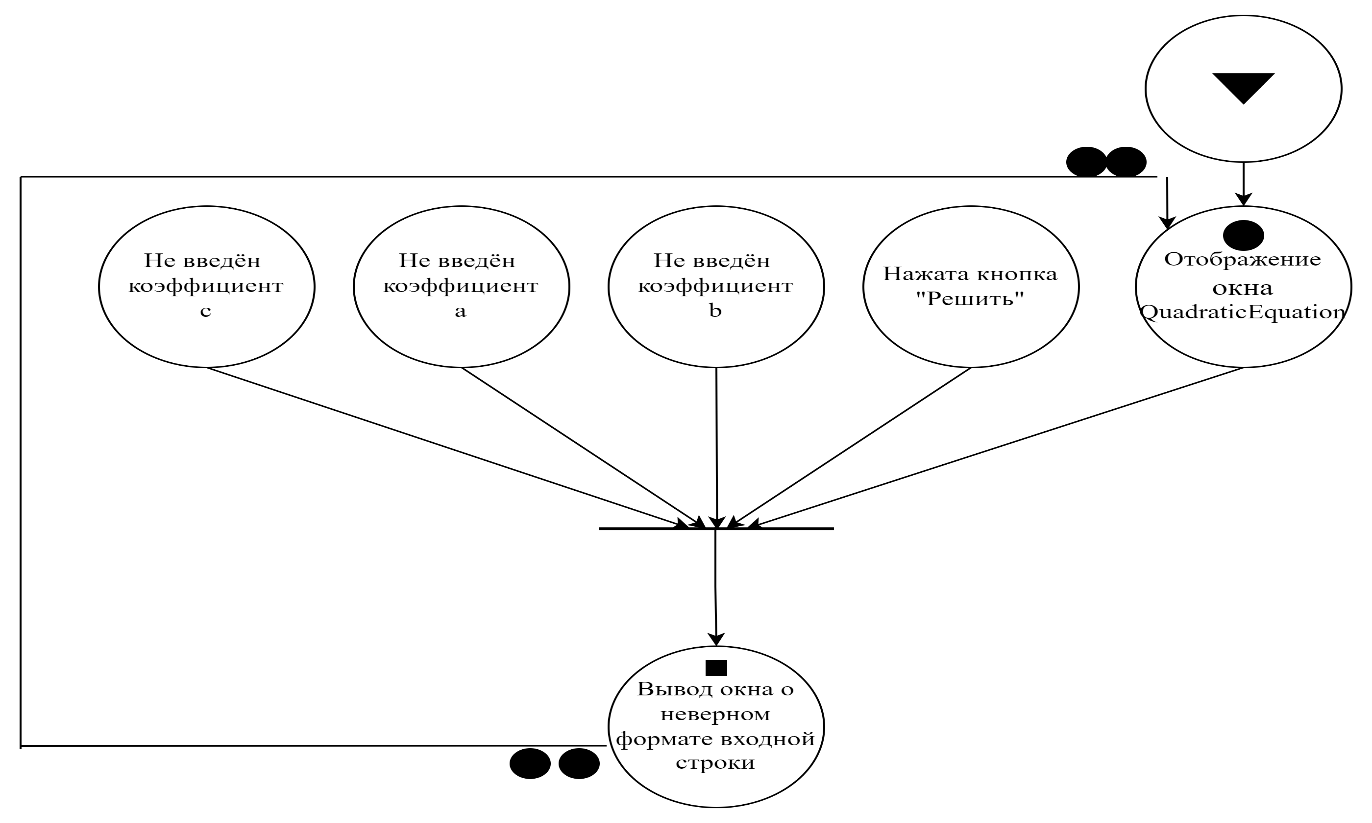


Рисунок 2 - Сеть Петри для случая, когда коэффициенты не введены

## 4. Схемы алгоритмов методов в составе решения

НАЧАЛО

Ввод:  
a, b, c

equation = new QuadraticEquation(a, b, c);  
equation.Solve()

equation.IsSolved

да

нет

Вывод:  
“Решено”;  
equation.Discriminant;  
equation.Root1;  
equation.Root2

equationsList.Add(equation);  
equationsListBox.Items.Add($"Уравнение {equationsList.Count}")

Вывод:  
“Не решено”;

КОНЕЦ

Рисунок 3 - Блок-схема обработчика кнопки “Решить уравнение”

НАЧАЛО

discriminant = b \* b - 4 \* a \* c

discriminant < 0

да

нет

isSolved = false

isSolved = true;  
root1 = (-b + Math.Sqrt(discriminant)) / (2 \* a);  
root2 = (-b - Math.Sqrt(discriminant)) / (2 \* a);

КОНЕЦ

Рисунок 4 - Блок-схема метода решения квадратного уравнения (в классе).

5. **Подбор тестовых примеров**

При вводе коэффициентов квадратного уравнения можно узнать его дискриминант, первый и второй корень.

1. Ввод коэффициентов a, b, c не происходит, но нажимается кнопка «Решить». В таком случае всплывет окно «Входная строка имела неверный формат», нельзя создать квадратное уравнение, не указав его коэффициенты.
2. Вводится: коэффициент a – 5, коэффициент b – 3, коэффициент с – 6. При нажатии соответствующей кнопки уравнение не создается, появляется окно с надписью «Не решено», так как дискриминант будет отрицательным.
3. Вводятся одинаковые значения: коэффициент a – 2, коэффициент b – 2, коэффициент с – 2. При нажатии соответствующей кнопки уравнение не создается, появляется окно с надписью «Не решено», так как дискриминант будет отрицательным.
4. Вводится: коэффициент a – 2, коэффициент b – (-7), коэффициент с – 3. При нажатии соответствующей кнопки уравнение создается, появляется окно с надписью «Решено». В поле «Дискриминант» значение станет равным 25. В поле «Корень 1» значение станет равным 3. В поле «Корень 2» значение станет равным 0.5.

**6. Листинг (код) составленного программного обеспечения**

Класс для решения квадратных уравнений

using System;

namespace QuadraticEquationSolver

{

public class QuadraticEquation

{

// Коэффициенты квадратного уравнения

private readonly double a;

private readonly double b;

private readonly double c;

// Расчетные свойства

private double discriminant;

private double? root1;

private double? root2;

// Флаг, указывающий на наличие решения

private bool isSolved;

// Конструктор для инициализации коэффициентов

public QuadraticEquation(double a, double b, double c)

{

this.a = a;

this.b = b;

this.c = c;

isSolved = false;

}

// Свойство для доступа к дискриминанту (только чтение)

public double Discriminant

{

get { return discriminant; }

}

// Свойство для доступа к первому корню (только чтение)

public double? Root1

{

get { return root1; }

}

// Свойство для доступа ко второму корню (только чтение)

public double? Root2

{

get { return root2; }

}

// Свойство для доступа к флагу решения

public bool IsSolved

{

get { return isSolved; }

set { isSolved = value; }

}

// Метод для решения квадратного уравнения

public void Solve()

{

discriminant = b \* b - 4 \* a \* c;

if (discriminant < 0)

{

isSolved = false;

}

else

{

isSolved = true;

root1 = (-b + Math.Sqrt(discriminant)) / (2 \* a);

root2 = (-b - Math.Sqrt(discriminant)) / (2 \* a);

}

}

}

}

Код обработчиков событий формы приложения

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace QuadraticEquationSolver

{

public partial class MainForm : Form

{

private List<QuadraticEquation> equationsList = new List<QuadraticEquation>();

public MainForm()

{

InitializeComponent();

}

// Обработчик нажатия кнопки "Решить уравнение"

private void solveButton\_Click(object sender, EventArgs e)

{

double a = double.Parse(coeffATextBox.Text);

double b = double.Parse(coeffBTextBox.Text);

double c = double.Parse(coeffCTextBox.Text);

QuadraticEquation equation = new QuadraticEquation(a, b, c);

equation.Solve();

if (equation.IsSolved)

{

statusLabel.Text = "Решено";

discriminantLabel.Text = $"Дискриминант: {equation.Discriminant}";

root1Label.Text = $"Корень 1: {equation.Root1}";

root2Label.Text = $"Корень 2: {equation.Root2}";

// Добавление уравнения в список и в ListBox

equationsList.Add(equation);

equationsListBox.Items.Add($"Уравнение {equationsList.Count}"); // Можно задать другое отображение в ListBox

}

else

{

statusLabel.Text = "Не решено";

discriminantLabel.Text = "";

root1Label.Text = "";

root2Label.Text = "";

}

}

// Обработчик нажатия кнопки "Очистить"

private void clearButton\_Click(object sender, EventArgs e)

{

equationsList.Clear();

equationsListBox.Items.Clear();

}

// Обработчик выбора элемента в списке уравнений

private void equationsListBox\_SelectedIndexChanged(object sender, EventArgs e)

{

int selectedIndex = equationsListBox.SelectedIndex;

if (selectedIndex >= 0 && selectedIndex < equationsList.Count)

{

QuadraticEquation selectedEquation = equationsList[selectedIndex];

statusLabel.Text = selectedEquation.IsSolved ? "Решено" : "Не решено";

discriminantLabel.Text= $"Дискриминант: {selectedEquation.Discriminant}";

root1Label.Text = $"Корень 1: {selectedEquation.Root1}";

root2Label.Text = $"Корень 2: {selectedEquation.Root2}";

}

}

}

}

**7. Графический пользовательский интерфейс программного обеспечения**

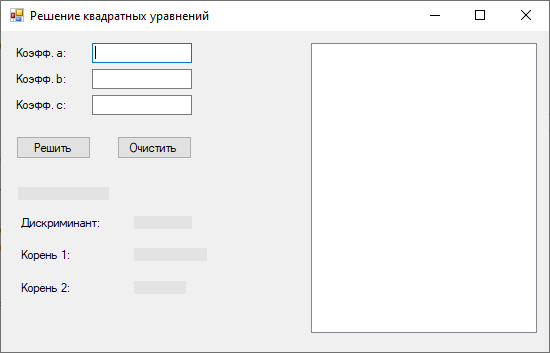


Рисунок 5 - Графический интерфейс

**8. Тестовые примеры в графическом интерфейсе**

****

Рисунок 6 – Расчёт тестового примера №1

В данном примере не введены коэффициенты a, b и с, но нажата кнопка «Решить».

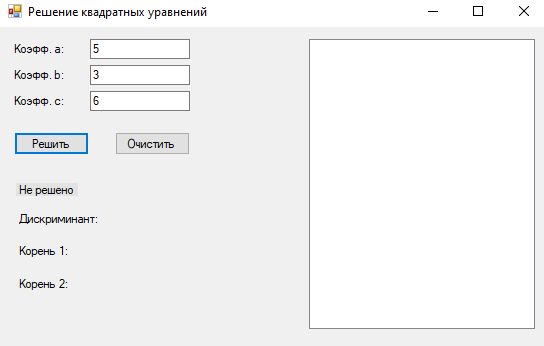
****

Рисунок 7 – Расчёт тестового примера №2

В данном примере квадратное уравнение не создается, появляется окно с надписью «Не решено», так как дискриминант будет отрицательным.

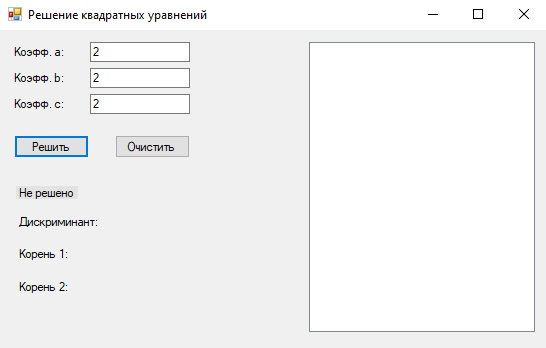
****

Рисунок 8 – Расчёт тестового примера №3

В данном примере введены одинаковые коэффициенты a, b и с, нажата кнопка «Решить», но квадратное уравнение не создается, появляется окно с надписью «Не решено», так как дискриминант будет отрицательным.

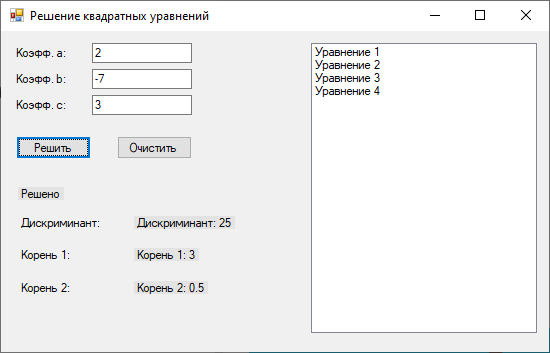


Рисунок 9 – Расчёт тестового примера №4

**9. Вывод о проделанной работе**

В данной работе можно поработать с несколькими классами в Windows Forms, создать конструктор, в состав которого входят: listtBox, label, button. Можно лучше изучить свойства конструктора, например поменять размер шрифта, его начертание. В отличие от предыдущих работ, в данной потребовалось большее количество элементов управления. Таким образом, можно не только научиться писать рабочий код, но и усовершенствовать навыки работы с графическим интерфейсом пользователя, а также это позволило лучше освоить построение Сетей Петри и блок-схем для визуализации и моделирования приложения.