Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет**»

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №9

Дисциплина: «Объектно-ориентированное программирование»

Тема: Классы и объекты.

Вариант 6

Выполнил:

Студент группы ИВТ-20-2б

Карелов Вадим Андреевич

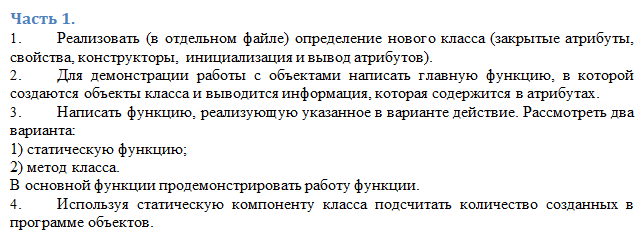
Проверила:

Доцент кафедры ИТАС

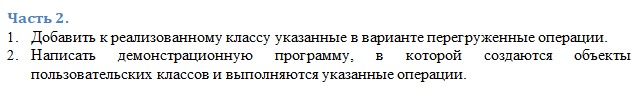
Викентьева О.Л.

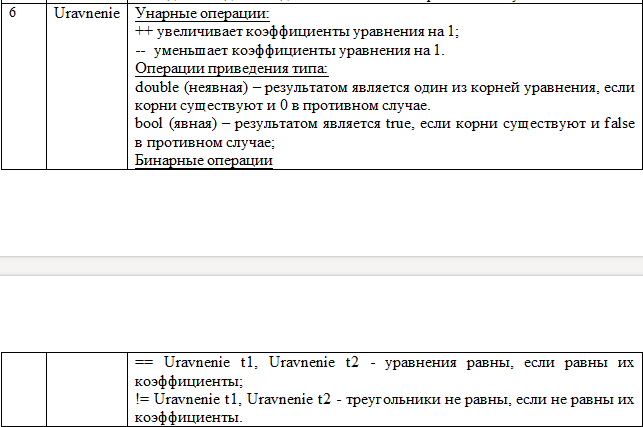
Пермь, 2021

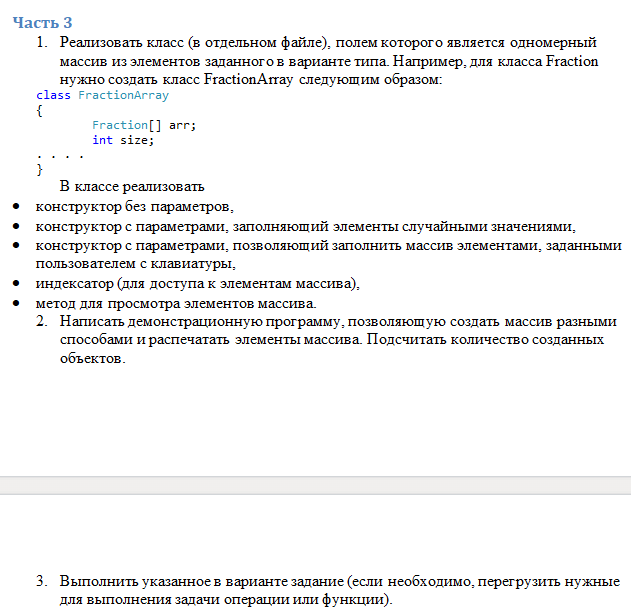
**Постановка задачи**







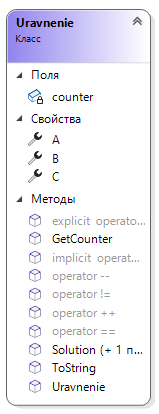




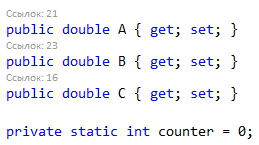


**Анализ**

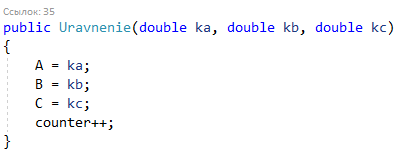
1. Реализация класса Уравнения (class Uravnenie)**.**



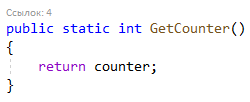
* 1. Поля класса. Класс содержит 3 поля типа int для хранения коэффициентов уравнения и статическую переменную для подсчета количества созданных экземпляров.



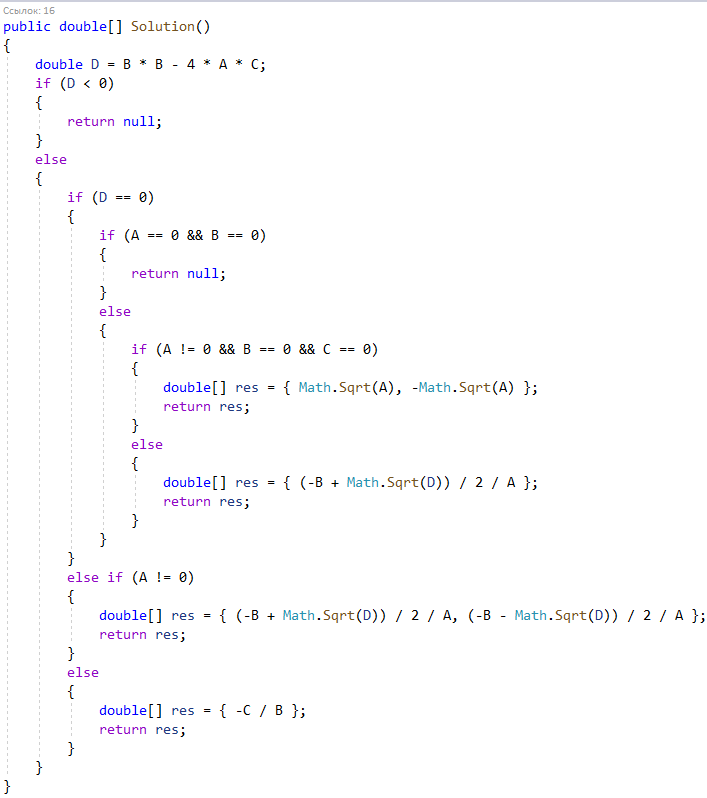
* 1. В классе реализован конструктор с параметрами. В параметры передаются коэффициенты уравнения, которые заносятся в поля.



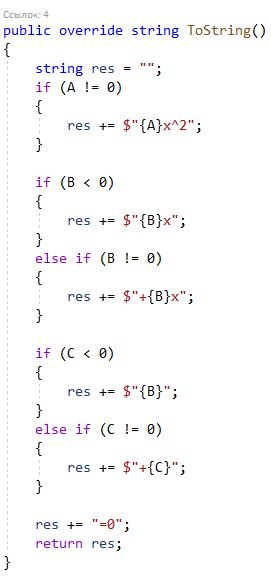
* 1. Метод возвращающий количество созданных экземпляров класса.



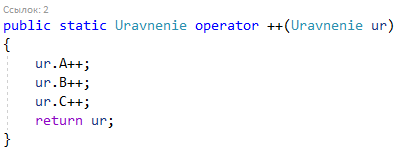
* 1. Метод для получения решения уравнения. Метод возвращает массив типа double, в котором может храниться одно или 2 решения. В случае, если решений нет, то возвращается null.

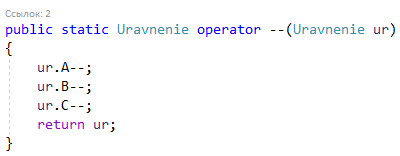


* 1. Статический метод для получения решения уравнения. Работает аналогично нестатическому, но коэффициенты передаются в качестве аргументов.
  2. Переопределенный метод ToString(). Выводит уравнение в виде Ax^2+Bx+C=0.

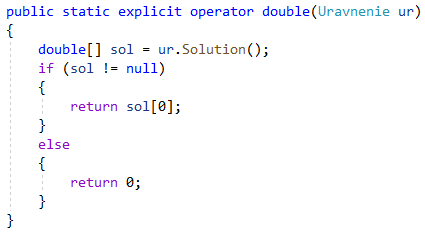


* 1. Определение инкремента и декремента.

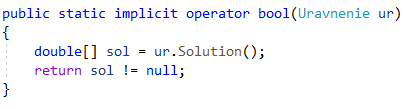




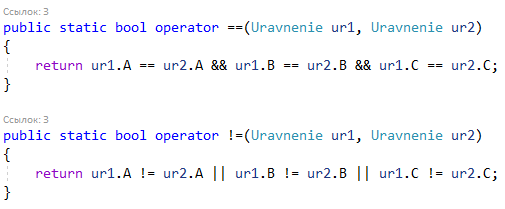
* 1. Явное приведение к типу double. Возвращается один из корней уравнения или 0, если решения нет.



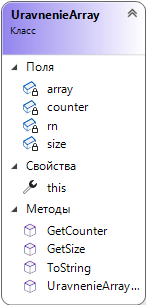
* 1. Неявное приведение к типу bool. Возвращается true, если есть решения уравнения.



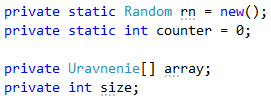
* 1. Определение операторов сравнения.



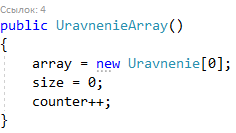
1. Класс, хранящий массив с уравнениями (UravnenieArray)

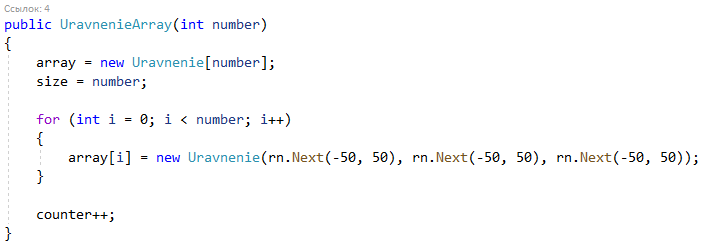


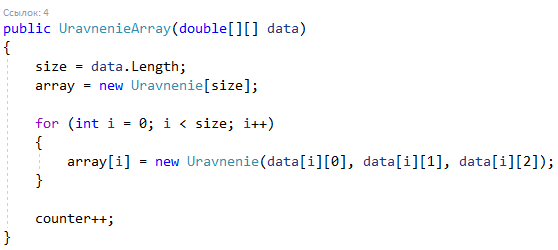
* 1. Поля класса. Статические переменнные – датчик случайных чисел, счетчик созданных экземпляров класса. Класс хранит массив типа Uravnenie и размер коллекции.



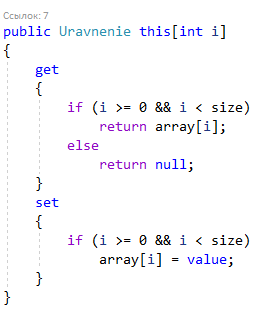
* 1. Три вида конструкторов: без параметров, создает пустую коллекцию, с заданным размером, коллекция заполняется случайным образом, с заданными коэффициентами уравнения.



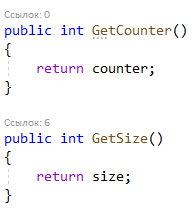




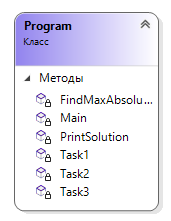
* 1. Индексатор.



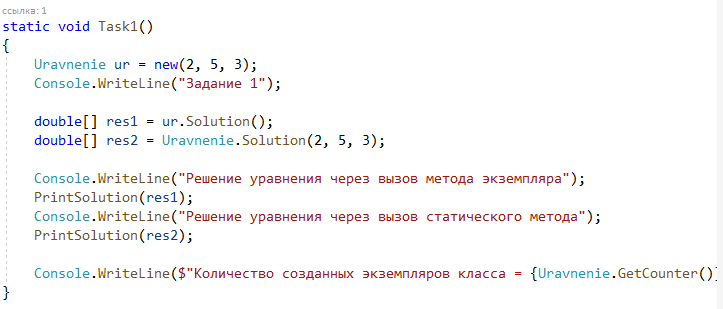
* 1. Методы, возвращающие размеры коллекции и количество созданных экземпляров.



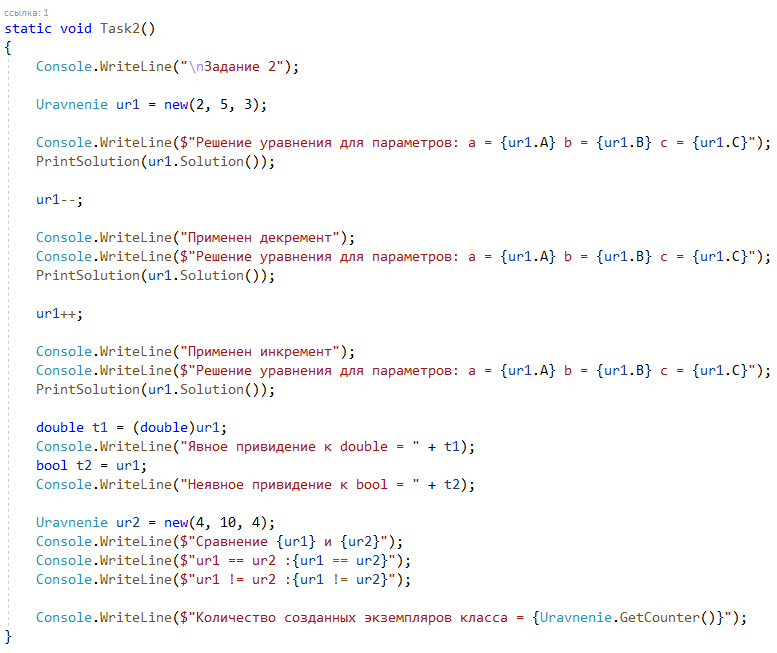
1. Класс программы



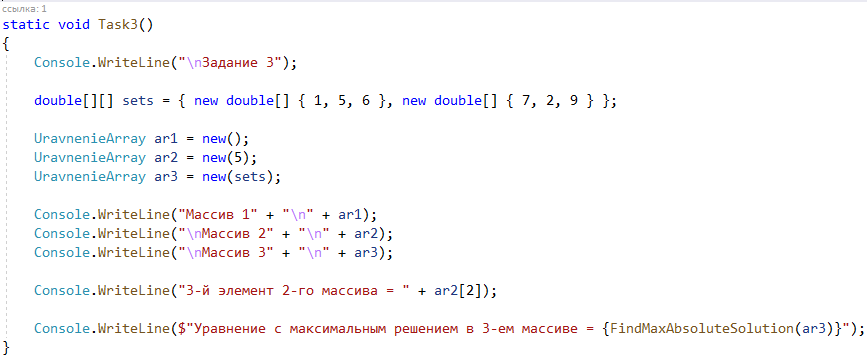
* 1. Задание 1. Демонстрация различных видов конструкторов и методов решения уравнения.



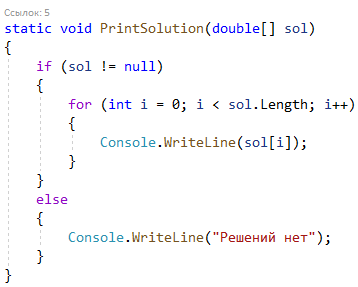
* 1. Задание 2. Демонстрация работы инкремента, декремента, привидения типов, операторов сравнения.



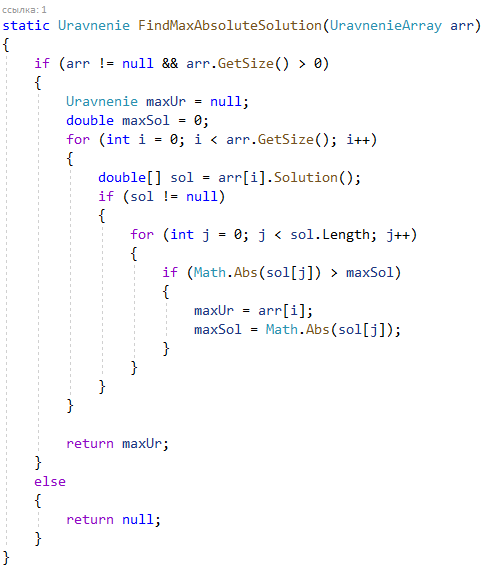
* 1. Задание 3. Демонстрация всех видов конструкторов, индексатора.



* 1. Вспомогательные методы.
     1. Печать решения.



* + 1. Нахождение максимального абсолютного значения решения уравнения среди массива уравнений.



**Код**

**Program.cs**

using System;

namespace LW\_2\_09

{

class Program

{

static void Main(string[] args)

{

Task1();

Task2();

Task3();

}

static void Task1()

{

Uravnenie ur = new(2, 5, 3);

Console.WriteLine("Задание 1");

double[] res1 = ur.Solution();

double[] res2 = Uravnenie.Solution(2, 5, 3);

Console.WriteLine("Решение уравнения через вызов метода экземпляра");

PrintSolution(res1);

Console.WriteLine("Решение уравнения через вызов статического метода");

PrintSolution(res2);

Console.WriteLine($"Количество созданных экземпляров класса = {Uravnenie.GetCounter()}");

}

static void Task2()

{

Console.WriteLine("\nЗадание 2");

Uravnenie ur1 = new(2, 5, 3);

Console.WriteLine($"Решение уравнения для параметров: a = {ur1.A} b = {ur1.B} c = {ur1.C}");

PrintSolution(ur1.Solution());

ur1--;

Console.WriteLine("Применен декремент");

Console.WriteLine($"Решение уравнения для параметров: a = {ur1.A} b = {ur1.B} c = {ur1.C}");

PrintSolution(ur1.Solution());

ur1++;

Console.WriteLine("Применен инкремент");

Console.WriteLine($"Решение уравнения для параметров: a = {ur1.A} b = {ur1.B} c = {ur1.C}");

PrintSolution(ur1.Solution());

double t1 = (double)ur1;

Console.WriteLine("Явное привидение к double = " + t1);

bool t2 = ur1;

Console.WriteLine("Неявное привидение к bool = " + t2);

Uravnenie ur2 = new(4, 10, 4);

Console.WriteLine($"Сравнение {ur1} и {ur2}");

Console.WriteLine($"ur1 == ur2 :{ur1 == ur2}");

Console.WriteLine($"ur1 != ur2 :{ur1 != ur2}");

Console.WriteLine($"Количество созданных экземпляров класса = {Uravnenie.GetCounter()}");

}

static void Task3()

{

Console.WriteLine("\nЗадание 3");

double[][] sets = { new double[] { 1, 5, 6 }, new double[] { 7, 2, 9 } };

UravnenieArray ar1 = new();

UravnenieArray ar2 = new(5);

UravnenieArray ar3 = new(sets);

Console.WriteLine("Массив 1" + "\n" + ar1);

Console.WriteLine("\nМассив 2" + "\n" + ar2);

Console.WriteLine("\nМассив 3" + "\n" + ar3);

Console.WriteLine("3-й элемент 2-го массива = " + ar2[2]);

Console.WriteLine($"Уравнение с максимальным решением в 3-ем массиве = {FindMaxAbsoluteSolution(ar3)}");

}

static void PrintSolution(double[] sol)

{

if (sol != null)

{

for (int i = 0; i < sol.Length; i++)

{

Console.WriteLine(sol[i]);

}

}

else

{

Console.WriteLine("Решений нет");

}

}

static Uravnenie FindMaxAbsoluteSolution(UravnenieArray arr)

{

if (arr != null && arr.GetSize() > 0)

{

Uravnenie maxUr = null;

double maxSol = 0;

for (int i = 0; i < arr.GetSize(); i++)

{

double[] sol = arr[i].Solution();

if (sol != null)

{

for (int j = 0; j < sol.Length; j++)

{

if (Math.Abs(sol[j]) > maxSol)

{

maxUr = arr[i];

maxSol = Math.Abs(sol[j]);

}

}

}

}

return maxUr;

}

else

{

return null;

}

}

}

}

**Uravnenie.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LW\_2\_09

{

public class Uravnenie

{

public double A { get; set; }

public double B { get; set; }

public double C { get; set; }

private static int counter = 0;

public Uravnenie(double ka, double kb, double kc)

{

A = ka;

B = kb;

C = kc;

counter++;

}

public static int GetCounter()

{

return counter;

}

public double[] Solution()

{

double D = B \* B - 4 \* A \* C;

if (D < 0)

{

return null;

}

else

{

if (D == 0)

{

if (A == 0 && B == 0)

{

return null;

}

else

{

if (A != 0 && B == 0 && C == 0)

{

double[] res = { Math.Sqrt(A), -Math.Sqrt(A) };

return res;

}

else

{

double[] res = { (-B + Math.Sqrt(D)) / 2 / A };

return res;

}

}

}

else if (A != 0)

{

double[] res = { (-B + Math.Sqrt(D)) / 2 / A, (-B - Math.Sqrt(D)) / 2 / A };

return res;

}

else

{

double[] res = { -C / B };

return res;

}

}

}

public static double[] Solution(double ka, double kb, double kc)

{

double D = kb \* kb - 4 \* ka \* kc;

if (D < 0)

{

return null;

}

else

{

if (D == 0)

{

if (ka == 0 && kb == 0)

{

return null;

}

else

{

if (ka != 0 && kb == 0 && kc == 0)

{

double[] res = { Math.Sqrt(ka), -Math.Sqrt(ka) };

return res;

}

else

{

double[] res = { (-kb + Math.Sqrt(D)) / 2 / ka };

return res;

}

}

}

else if (ka != 0)

{

double[] res = { (-kb + Math.Sqrt(D)) / 2 / ka, (-kb - Math.Sqrt(D)) / 2 / ka };

return res;

}

else

{

double[] res = { -kc / kb };

return res;

}

}

}

public override string ToString()

{

string res = "";

if (A != 0)

{

res += $"{A}x^2";

}

if (B < 0)

{

res += $"{B}x";

}

else if (B != 0)

{

res += $"+{B}x";

}

if (C < 0)

{

res += $"{B}";

}

else if (C != 0)

{

res += $"+{C}";

}

res += "=0";

return res;

}

public static Uravnenie operator ++(Uravnenie ur)

{

ur.A++;

ur.B++;

ur.C++;

return ur;

}

public static Uravnenie operator --(Uravnenie ur)

{

ur.A--;

ur.B--;

ur.C--;

return ur;

}

public static explicit operator double(Uravnenie ur)

{

double[] sol = ur.Solution();

if (sol != null)

{

return sol[0];

}

else

{

return 0;

}

}

public static implicit operator bool(Uravnenie ur)

{

double[] sol = ur.Solution();

return sol != null;

}

public static bool operator ==(Uravnenie ur1, Uravnenie ur2)

{

return ur1.A == ur2.A && ur1.B == ur2.B && ur1.C == ur2.C;

}

public static bool operator !=(Uravnenie ur1, Uravnenie ur2)

{

return ur1.A != ur2.A || ur1.B != ur2.B || ur1.C != ur2.C;

}

}

}

**UravnenieArray.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LW\_2\_09

{

public class UravnenieArray

{

private static Random rn = new();

private static int counter = 0;

private Uravnenie[] array;

private int size;

public UravnenieArray()

{

array = new Uravnenie[0];

size = 0;

counter++;

}

public UravnenieArray(int number)

{

array = new Uravnenie[number];

size = number;

for (int i = 0; i < number; i++)

{

array[i] = new Uravnenie(rn.Next(-50, 50), rn.Next(-50, 50), rn.Next(-50, 50));

}

counter++;

}

public UravnenieArray(double[][] data)

{

size = data.Length;

array = new Uravnenie[size];

for (int i = 0; i < size; i++)

{

array[i] = new Uravnenie(data[i][0], data[i][1], data[i][2]);

}

counter++;

}

public Uravnenie this[int i]

{

get

{

if (i >= 0 && i < size)

return array[i];

else

return null;

}

set

{

if (i >= 0 && i < size)

array[i] = value;

}

}

public override string ToString()

{

string res = "";

if (array != null && array.Length > 0)

{

for (int i = 0; i < size; i++)

{

res += array[i].ToString() + "\n";

}

}

else

{

res = "Массив пуст";

}

return res;

}

public int GetCounter()

{

return counter;

}

public int GetSize()

{

return size;

}

}

}

**Скриншоты/Тесты**

