

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра программных систем

**Задание №2**

**Обработка исключений**

Выполнил: Мороз

Илья Олегович

Группа № K3220

Преподаватель: Иванов Сергей Евгеньевич

Санкт-Петербург

2020

**Задание:** с помощью классов в методе нахождения обратной матрицы (задание 1) реализовать обработку исключения в случае операций деления на ноль и неособенной невырожденной, не сингулярной квадратной матрицы.

**Исходный код:**

**Program.cs**

using System;  
  
namespace task2  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 Matrix matrix = new Matrix();  
 Console.WriteLine(matrix);  
 Console.WriteLine(matrix.InverseMatrix());  
 }  
 }  
}

**Matrix.cs**

using System;  
using Microsoft.VisualBasic.CompilerServices;  
  
namespace task2  
{  
 */\*  
 \* Class implements matrix and methods for working with it  
 \*/* public class Matrix  
 {  
 */\*  
 \* Variable with matrix data  
 \*/* private double[,] matrix;  
  
 */\*  
 \* Constructor for matrix object  
 \* Create matrix by user input  
 \*/* public Matrix()  
 {  
 Console.WriteLine("Enter the matrix:");  
 this.MatrixInput();  
 }  
  
 */\*  
 \* Constructor for matrix  
 \* Create matrix by array of double  
 \*/* public Matrix(double[,] matrix)  
 {  
 this.matrix = matrix;  
 }  
  
 */\*  
 \* Override ToString method for matrix objects  
 \*/* public override string ToString()  
 {  
 string result = "\n";  
 int rows = this.matrix.GetLength(0);  
 int columns = this.matrix.GetLength(1);  
 for (int i = 0; i < rows; i++)  
 {  
 for (int j = 0; j < columns; j++)  
 {  
 result += $"{this.matrix[i, j]}\t";  
 }  
  
 result += "\n";  
 }  
  
 return result;  
 }  
  
 private void MatrixInput()  
 {  
 try  
 {  
 Console.Write("Count of rows: ");  
 int n = Int32.Parse(Console.ReadLine());  
 Console.Write("Count of columns: ");  
 int m = Int32.Parse(Console.ReadLine());  
 this.matrix = new double[n, m];  
 Console.WriteLine("Enter matrix elements: ");  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < m; j++)  
 {  
 this.matrix[i, j] = Double.Parse(Console.ReadLine());  
 }  
 }  
 }  
 catch (FormatException e)  
 {  
 ErrorHandler.WriteErrorInConsole(e, "Parse error. Maybe, you write NaN element.");  
 }  
 }  
  
 */\*  
 \* Return minor for element at position [row, column]  
 \*/* private static double[,] MatrixMinor(double[,] matrix, int row, int column)  
 {  
 int n = matrix.GetLength(0);  
 int x = 0;  
 double[,] result = new double[n - 1, n - 1];  
 for (int i = 0; i < n; i++)  
 {  
 */\*  
 \* Skip element row  
 \*/* if (i == row) continue;  
 int y = 0;  
 for (int j = 0; j < n; j++)  
 {  
 */\*  
 \* Skip element column  
 \*/* if (j == column) continue;  
 result[x, y] = matrix[i, j];  
 y++;  
 }  
  
 x++;  
 }  
  
 return result;  
 }  
  
 */\*  
 \* Return matrix determinant  
 \*/* private static double MatrixDeterminant(double[,] matrix)  
 {  
 double result = 0;  
 int n = matrix.GetLength(0);  
 if (n == 1) return matrix[0, 0];  
 if (n == 2) return matrix[0, 0] \* matrix[1, 1] - matrix[0, 1] \* matrix[1, 0];  
 for (int i = 0; i < n; i++)  
 {  
 result += Math.Pow(-1, i + 2) \* matrix[i, 0] \* MatrixDeterminant(MatrixMinor(matrix, i, 0));  
 }  
  
 return result;  
 }  
  
 */\*  
 \* Return inverse matrix  
 \*/* public Matrix InverseMatrix()  
 {  
 int n = this.matrix.GetLength(0);  
 int m = this.matrix.GetLength(1);  
 double[,] result = new double[n, m];  
 try  
 {  
 if (n != m) throw new NonSquareMatrixException();  
 if (MatrixDeterminant(this.matrix).Equals(0.0))  
 throw new DivideByZeroException("Matrix determinant is zero. You can't find inverse matrix.");  
 double[,] transposedMatrix = new double[n, m];  
  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < m; j++)  
 {  
 transposedMatrix[i, j] = matrix[j, i];  
 }  
 }  
  
 for (int i = 0; i < n; i++)  
 {  
 for (int j = 0; j < m; j++)  
 {  
 result[i, j] = Math.Pow(-1, i + j + 2) \*  
 MatrixDeterminant(MatrixMinor(transposedMatrix, i, j)) /  
 MatrixDeterminant(matrix);  
 }  
 }  
 }  
 catch (Exception e)  
 {  
 ErrorHandler.WriteErrorInConsole(e, e.Message);  
 }  
  
 return new Matrix(result);  
 }  
 }  
}

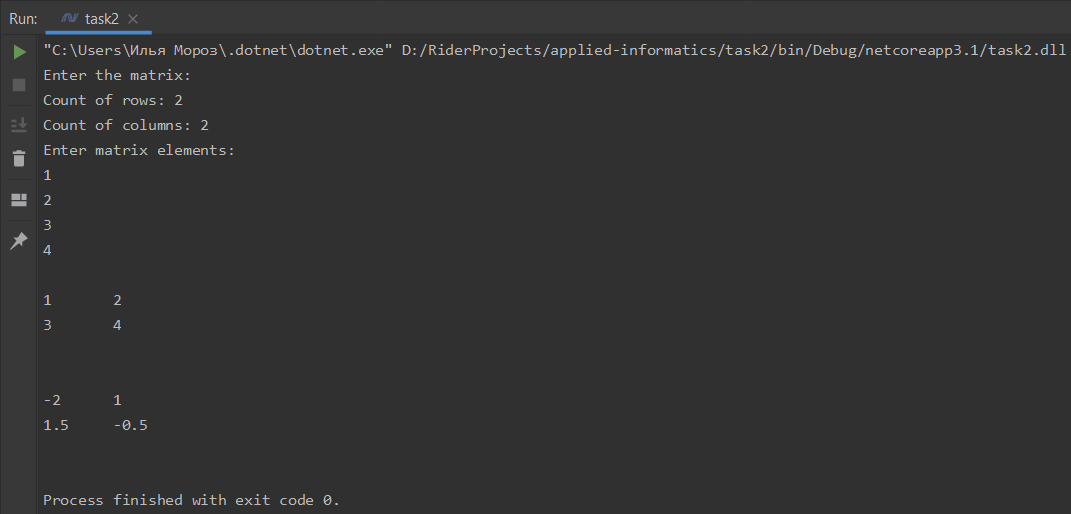
**NonSquareMatrixException.cs**

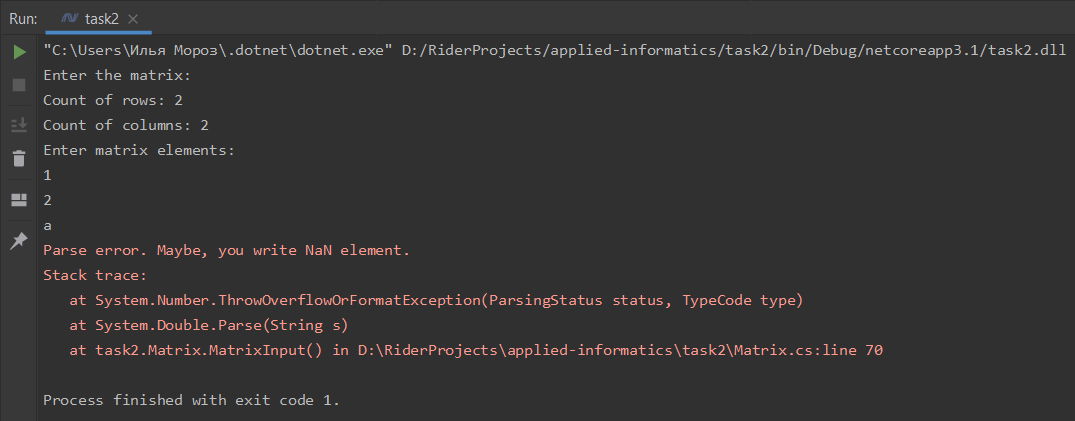
using System;  
  
namespace task2  
{  
 public class NonSquareMatrixException : ArithmeticException  
 {  
 public override string Message { get; } = "The matrix is not square.";  
 }  
}

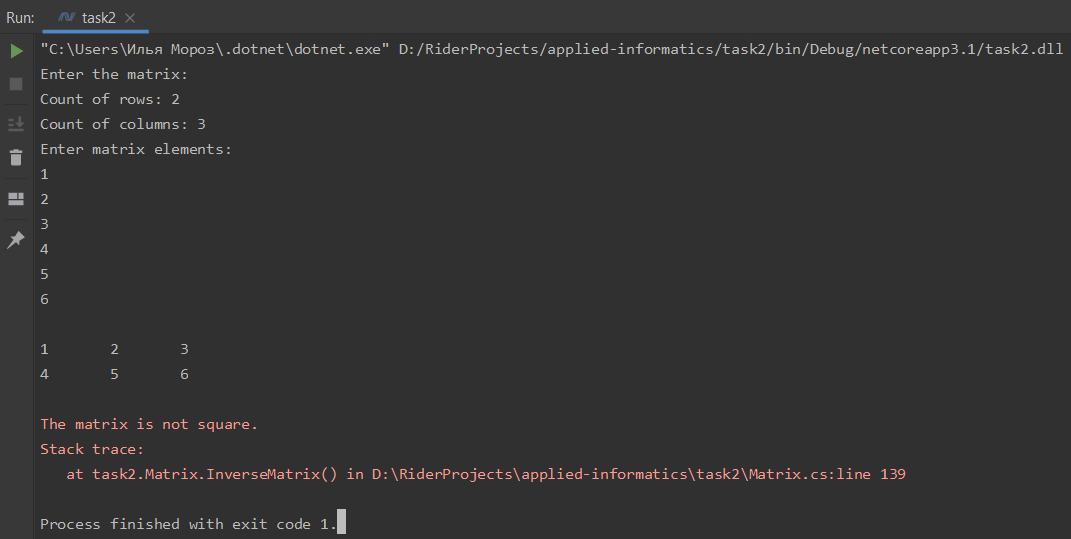
**ErrorHandler.cs**

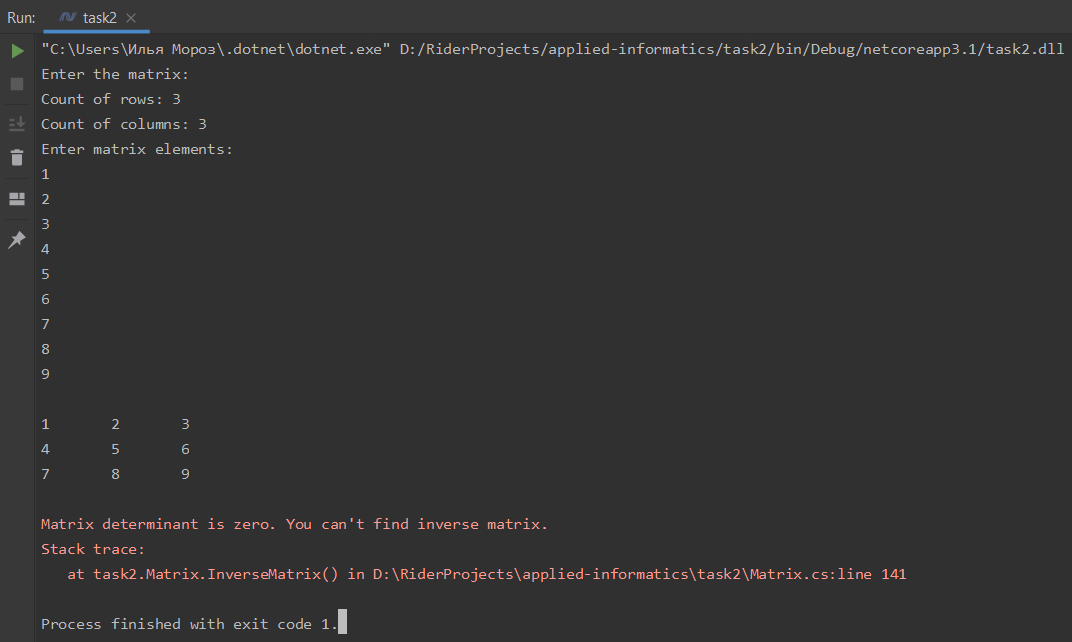
using System;  
  
namespace task2  
{  
 public class ErrorHandler  
 {  
 public static void WriteErrorInConsole(Exception exception, string message = "Something went wrong!")  
 {  
 Console.ForegroundColor = ConsoleColor.**Red**;  
 Console.WriteLine(message);  
 Console.WriteLine("Stack trace:");  
 Console.WriteLine(exception.StackTrace);  
 Console.ResetColor();  
 System.Environment.Exit(1);  
 }  
 }  
}

**Результат работы:**

****

****

****



**Вывод:** с помощью класса Matrix была реализована абстракция над двумерным массивом. С помощью try-catch были обработаны ошибки, которые могут возникнуть при выполнении программы.

Исходный код также доступен на GitHub: [ilyamore88/applied-informatics/task2](https://github.com/ilyamore88/applied-informatics/tree/master/task2).