|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Факультет «Радиотехнический»**

**Кафедра «Системы обработки информации и управления»**

Лабораторная работа № 2

по дисциплине «Базовые компоненты интернет технологий»

Выполнил:

студент(ка) группы № РТ5-31Б

Пакало Александр Сергеевич

подпись, дата

Проверил:

преподаватель каф. ИУ5  
Гапанюк Юрий Евгеньевич

подпись, дата

2020 г.

Оглавление

[Описание задания. 3](#_Toc59479672)

[Диаграмма классов. 3](#_Toc59479673)

[Текст программы. 4](#_Toc59479674)

[Program.cs 4](#_Toc59479675)

[IPrint.cs 5](#_Toc59479676)

[GeometricFigure.cs 5](#_Toc59479677)

[Circle.cs 5](#_Toc59479678)

[Rectangle.cs 6](#_Toc59479679)

[Square.cs 7](#_Toc59479680)

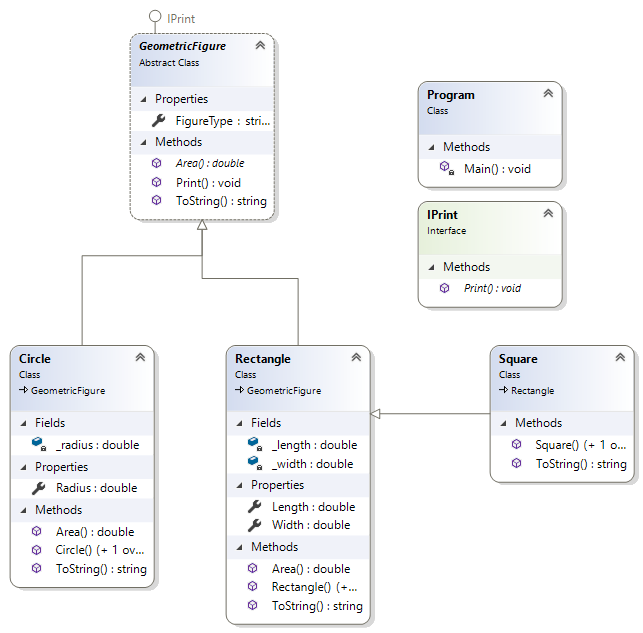
[Анализ результатов. 8](#_Toc59479681)

# Описание задания.

Разработать программу, реализующую работу с классами.

1. Программа должна быть разработана в виде консольного приложения на языке C#.
2. Абстрактный класс «Геометрическая фигура» содержит виртуальный метод для вычисления площади фигуры.
3. Класс «Прямоугольник» наследуется от «Геометрическая фигура». Ширина и высота объявляются как свойства (property). Класс должен содержать конструктор по параметрам «ширина» и «высота».
4. Класс «Квадрат» наследуется от «Прямоугольник». Класс должен содержать конструктор по длине стороны.
5. Класс «Круг» наследуется от «Геометрическая фигура». Радиус объявляется как свойство (property). Класс должен содержать конструктор по параметру «радиус».
6. Для классов «Прямоугольник», «Квадрат», «Круг» переопределить виртуальный метод Object.ToString(), который возвращает в виде строки основные параметры фигуры и ее площадь.
7. Разработать интерфейс IPrint. Интерфейс содержит метод Print(), который не принимает параметров и возвращает void. Для классов «Прямоугольник», «Квадрат», «Круг» реализовать наследование от интерфейса IPrint. Переопределяемый метод Print() выводит на консоль информацию, возвращаемую переопределенным методом ToString().

# Диаграмма классов.



# Текст программы.

## Program.cs

using System;

namespace GeometricFigures

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Лабораторная работа выполнена Пакало Александром Сергеевичем, студентом РТ5-31Б.");

// Figure parameters.

double width,

length,

squareWidth,

radius;

Console.WriteLine("\tRectangle: ");

while (true)

{

Console.Write("Enter the width: ");

width = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter the length: ");

length = Convert.ToDouble(Console.ReadLine());

try

{

Rectangle R = new Rectangle(width, length);

R.Print();

break;

}

catch (ArgumentException e)

{

Console.WriteLine(e.Message);

Console.WriteLine("Please, enter all parameters once again: ");

}

}

Console.WriteLine("\tSquare: ");

while (true)

{

Console.Write("Enter the width: ");

squareWidth = Convert.ToDouble(Console.ReadLine());

try

{

Rectangle S = new Square(squareWidth);

S.Print();

break;

}

catch (ArgumentException e)

{

Console.WriteLine(e.Message);

Console.WriteLine("Please, enter all parameters once again: ");

}

}

Console.WriteLine("\tCircle: ");

while (true)

{

Console.Write("Enter the radius: ");

radius = Convert.ToDouble(Console.ReadLine());

try

{

Circle C = new Circle(radius);

C.Print();

break;

}

catch (ArgumentException e)

{

Console.WriteLine(e.Message);

Console.WriteLine("Please, enter all parameters once again: ");

}

}

}

}

}

## IPrint.cs

namespace GeometricFigures

{

public interface IPrint

{

void Print();

}

}

## GeometricFigure.cs

using System;

namespace GeometricFigures

{

public abstract class GeometricFigure : IPrint

{

public string FigureType { get; protected set; }

public abstract double Area();

public override string ToString()

{

return String.Format("\t{0}: ", this.FigureType) + "\n" + "Area = " + Area();

}

public void Print()

{

Console.WriteLine(this.ToString());

}

}

}

## Circle.cs

using System;

namespace GeometricFigures

{

public class Circle : GeometricFigure

{

private double \_radius;

public double Radius

{

get

{

return \_radius;

}

private set

{

if (value < 0) throw new ArgumentException(String.Format("Argument Exception: all dimensions of a geometric figure should be >= 0. {0} < 0!", value), "Radius");

\_radius = value;

}

}

public Circle()

{

FigureType = "Circle";

}

public Circle(double radius) : this()

{

Radius = radius;

}

public override double Area()

{

return Math.PI \* Radius \* Radius;

}

public override string ToString()

{

return base.ToString() + "\n" +

"Radius = " + Radius;

}

}

}

## Rectangle.cs

using System;

namespace GeometricFigures

{

public class Rectangle : GeometricFigure

{

private double \_width;

public double Width

{

get

{

return \_width;

}

private set

{

if (value < 0) throw new ArgumentException(String.Format("Argument Exception: all dimensions of a geometric figure should be >= 0. {0} < 0!", value), "Width");

\_width = value;

}

}

private double \_length;

public double Length

{

get

{

return \_length;

}

private set

{

if (value < 0) throw new ArgumentException(String.Format("Argument Exception: all dimensions of a geometric figure should be >= 0. {0} < 0!", value), "Length");

\_length = value;

}

}

public Rectangle()

{

FigureType = "Rectangle";

}

public Rectangle(double width, double length) : this()

{

Width = width;

Length = length;

}

public override double Area()

{

return Width \* Length;

}

public override string ToString()

{

return base.ToString() + "\n" +

"Width = " + Width + "\n" +

"Length = " + Length;

}

}

}

## Square.cs

using System;

namespace GeometricFigures

{

public class Square : Rectangle

{

public Square()

{

FigureType = "Square";

}

public Square(double width) : base(width, width)

{

FigureType = "Square";

}

public override string ToString()

{

return String.Format("\t{0}: ", this.FigureType) + "\n" +

"Area = " + Area() + "\n" +

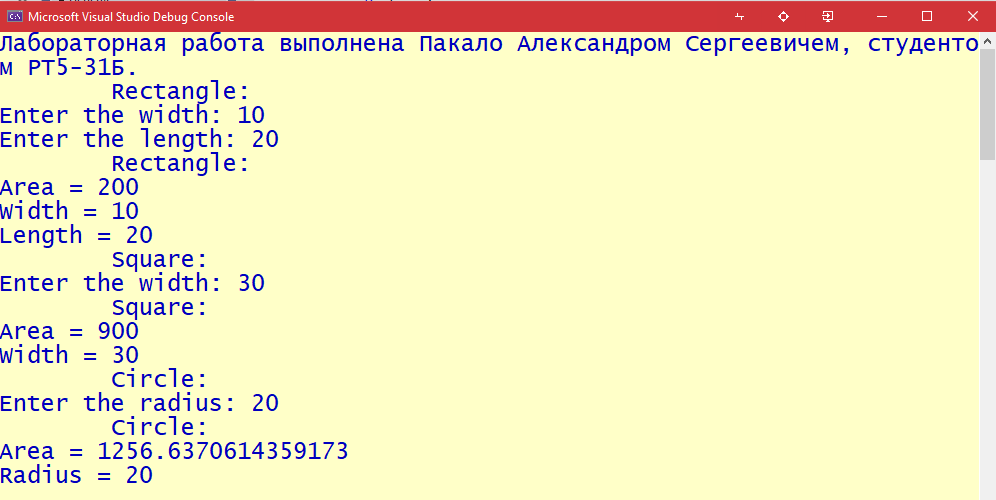
"Width = " + Width;

}

}

}

# Анализ результатов.



Скриншот 1 (демонстрация работы программы).