Московский государственный технический университет и Н.Э. Баумана

Факультет ИУ «Информатика и системы управления»

Кафедра ИУ-5 «Системы обработки информации и управления»

ОТЧЕТ

по лабораторной работе №2

Выполнил:

Студент ИУ5-32Б

Нырков Илья Алексеевич

Проверил:

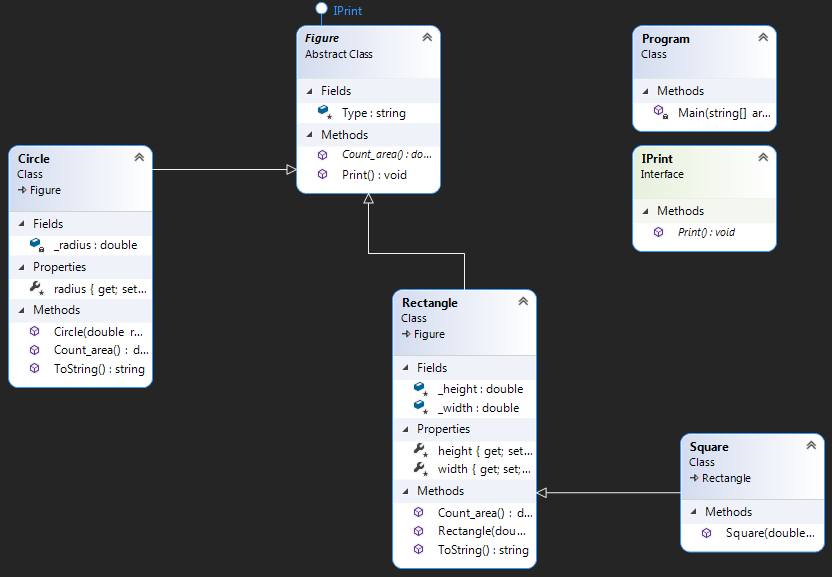
Доцент Гапанюк Ю. Е.

МОСКВА 2020

**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:**

****

**Листинг кода программы:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Geometric\_figures

{

interface IPrint

{

void Print();

}

abstract class Figure : IPrint

{

public abstract double Count\_area();

protected string Type;

public void Print()

{

Console.WriteLine(this.ToString());

}

}

class Rectangle : Figure

{

protected double \_height;

protected double height

{

get { return \_height; }

set

{

if (value < 0)

throw new ArgumentOutOfRangeException(

"height must not be negative");

\_height = value;

}

}

protected double \_width;

protected double width

{

get { return \_width; }

set

{

if (value < 0)

throw new ArgumentOutOfRangeException(

"width must not be negative");

\_width = value;

}

}

public Rectangle(double height, double width)

{

this.height = height;

this.width = width;

this.Type = "Rectangle";

}

public override double Count\_area()

{

return height \* width;

}

public override string ToString()

{

return Type + ": height = " + height.ToString() +

" width = " + width.ToString() + " Area = " + Count\_area().ToString();

}

}

class Square : Rectangle

{

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

{

Type = "Square";

}

}

class Circle : Figure

{

private double \_radius;

protected double radius

{

get { return \_radius; }

set

{

if (value < 0)

throw new ArgumentOutOfRangeException(

"radius must not be negative");

\_radius = value;

}

}

public override double Count\_area()

{

return 3.1415 \* \_radius \* \_radius;

}

public Circle(double radius)

{

this.radius = radius;

Type = "Circle";

}

public override string ToString()

{

return Type + ": radius = " + \_radius + " Area = " + Count\_area();

}

}

class Program

{

static void Main(string[] args)

{

Circle circle = new Circle(4);

Square square = new Square(5);

Rectangle rectangle = new Rectangle(4, 5);

circle.Print();

square.Print();

rectangle.Print();

Console.ReadLine();

}

}

}

**Скриншоты работы программы**

