Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №6

на тему:

**«Наследование»**

БГУИР 6-05-0612-02 34

|  |
| --- |
| Выполнил студент группы 353505  Хорошко Кирилл Николаевич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

Задание 1. Вариант 1.Выделить в предметной области 2-3 варианта сущности (например, "Геометрическая фигура", "Точка", "Треугольник"), отличающиеся несколькими полями и методами. Каждый класс имеет поля, свойства и методы.

1. Спроектировать UML-диаграммы классов.
2. Базовый класс для вашей иерархии объявите абстрактным. Он должен содержать абстрактные методы и методы с реализацией.
3. Один из наследников должен перегружать метод родителя.
4. Один из классов должен содержать виртуальный метод, который

переопределяется в одном наследнике и не переопределяется в другом.

5) Продемонстрировать работу всех объявленных методов.

6) Продемонстрировать вызов конструктора родительского класса при наследовании.

7) Продемонстрировать вызов метода родительского класса при его

Скрытии.

8) Создать класс, закрытый для наследования.

Индивидуальное задание – геометрические фигуры.

# 2 Выполнение работы

Для выполнения данной лабораторной работы первым делом был создан проект. Проект содержал 5 классов, один из них Program, который содержит Main. Сначала был реализован абстрактный класс “Geometry От класса “Geometry” наследуется два класса: “Rectangle” и “Circle”, который объявлен с модификатором “sealed” для того, чтобы от этого класса нельзя было наследоваться.

public abstract class Geometry

{

protected abstract int S();

protected abstract int P();

protected int angel\_value(int amount\_side)

{

return 180 \* (amount\_side - 2);

}

public virtual string print\_figure() {

return "";

}

}

public class Rectangle : Geometry

{

public int a = 0;

public int b = 0;

protected int angle;

public Rectangle()

{

angle = base.angel\_value(4)/4;

}

public Rectangle(int a, int b)

{

this.a = a;

this.b = b;

angle = base.angel\_value(4)/4;

}

protected override int S()

{

return a \* b;

}

protected override int P()

{

return 2 \* (a + b);

}

public sealed override string print\_figure()

{

return "a = " + a + " b = " + b + " S = " + S() + " P = " + P() + " Угол: " + angle;

}

}

public sealed class Circle : Geometry

{

public int R;

public Circle(){ }

public Circle(int R)

{

this.R = R;

}

protected sealed override int S()

{

return 3 \* R;

}

protected sealed override int P()

{

return 6 \* R;

}

public sealed override string print\_figure()

{

return "R = " + R + " S = " + S() + " P = " + P();

}

}

public sealed class Square : Rectangle

{

public Square(int a) : base()

{

this.a = a;

angle = base.angle;

}

new int S()

{

return a \* a;

}

new int P()

{

return 4\*a;

}

public string square()

{

return "a = " + a + " S = " + S() + " P = " + P() + " Угол: " + angle + " Rectangle P: " + base.P();

}

}

class Program

{

static void Main(string[] args)

{

Rectangle rect = new Rectangle(5, 10);

Square square = new Square(20);

Circle circle = new Circle(40);

Console.WriteLine(rect.print\_figure());

Console.WriteLine(square.square());

Console.WriteLine(circle.print\_figure());

}

}



Рисунок 1 — Результат выполнения программы

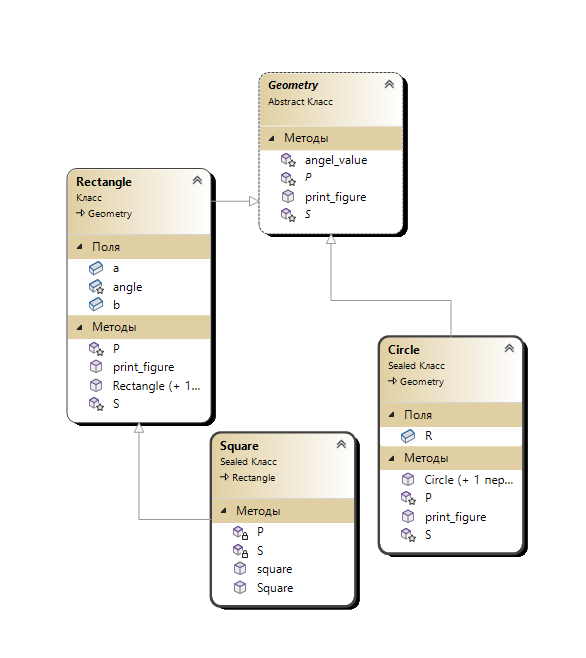


Рисунок 2 — UML-диаграмма

# Вывод

В ходе лабораторной работы мною было изучено наследование в языке C#. Мною были углублены знания по работе с классами в объектно-ориентированным языком программирования C#. Были рассмотрены обращения к классу-родителю из класса-наследник для вызова методов класса-родителя. Была изучена схема создания UML-диаграмм, которые позволяют отображать отношения между классами, что позволяет более простым способом их анализировать. Было установлено, что при создании объекта класса-наследника вызывается конструктор класса-родителя. Конструктор класса-родителя, который будет вызываться можно установить с помощью ключевого слова “base”.