Практическая работа №2

Наследование в языке C#

Цель работы: изучить наследование в языке C#.

Теоретическая часть

Наследование определяет способность языка позволять строить новые определения классов на основе определений существующих классов. По сути, наследование позволяет расширять поведение базового класса, наследуя основную функциональность в производном подклассе.

Всякий раз, когда один класс наследует от другого, после имени производного класса указывается имя базового класса, отделяемое двоеточием.

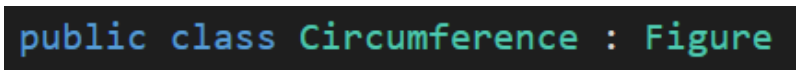

Рисунок 1 – Объявление родительского класса

Производный класс может иметь только один прямой базовый класс.

Однако наследование является транзитивным. Если ClassC является производным от ClassB, а ClassB — от ClassA, ClassC наследует члены, объявленные в ClassB и ClassA.

Абстрактные классы

Такие классы предназначены для описания сущностей, которые не имеют конкретного воплощения.

Абстрактный (abstract) класс можно использовать, только если новый класс является производным от него.

Абстрактный класс может содержать один или несколько сигнатур методов, которые сами объявлены в качестве абстрактных. Эти сигнатуры

задают параметры и возвращают значение, но не имеют реализации (тела метода).

Абстрактному классу необязательно содержать абстрактные члены; однако если класс все же содержит абстрактный член, то сам класс должен быть объявлен в качестве абстрактного.

Производные классы, которые сами не являются абстрактными, должны предоставить реализацию для любых абстрактных методов из абстрактного базового класса.

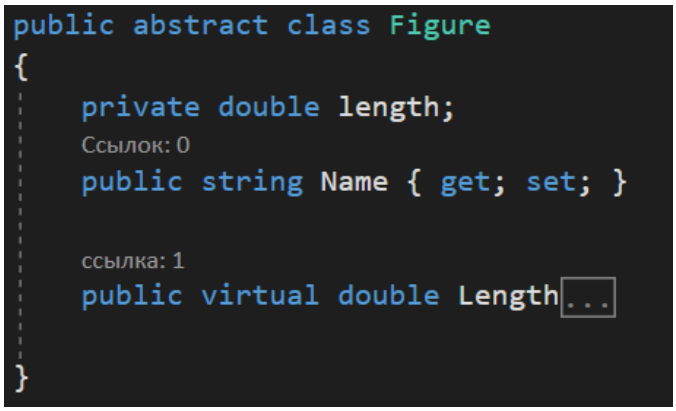

Рисунок 2 – Объявление абстрактного класса

Абстрактные и виртуальные методы

Когда базовый класс объявляет метод как virtual, производный класс может переопределить (override) метод с помощью своей собственной реализации. Если базовый класс объявляет метод как abstract, этот метод должен быть переопределен в любом неабстрактном классе, который прямо наследует от этого класса. Если производный класс сам является абстрактным, то он наследует абстрактные члены, не реализуя их.

Помимо методов, виртуальными могут быть свойства, события и индексаторы.

```
public override double Length
{
    get
    {
        return 2*Math.PI*Radius;
    }
}
```

Рисунок 3 – Переопределение виртуального свойства Length

Ключевое слово base

Ключевое слово base используется для доступа к членам базового из производного класса в следующих случаях:

- Вызов метода базового класса, который был переопределен другим методом.
- Определение конструктора базового класса, который должен вызываться при создании экземпляров производного класса.

Рассмотрим пример. Дан базовый класс Item, для него описан конструктор, который принимает два аргумента.

```
public class Item
{
    ссылка: 1
    public string Name { get; private set; }
    ссылка: 1
    public int Price { get; private set; }

    ссылка: 1
    public Item (string name, int price)
    {
        this.Name = name;
        this.Price = price;
    }
}
```

Рисунок 4 – Класс Item

В производном классе Table конструктор использует конструктор базового класса Item.

```
public class Table : Item
{
    ссылка: 1
    public string Color { get; private set; }
    Ссылок: 0
    public Table (string tableName,
                  int tablePrice,
                  string color)
                  : base (tableName, tablePrice)
    {
        this.Color = color;
    }
}
```

Рисунок 5 – Класс Table

Практическая часть

Необходимо разработать систему классов:

1. Фигура
    a. Круг
        i. Эллипс
        ii. Кольцо
    b. Параллелограмм
        i. Прямоугольник
        ii. Ромб
    c. Правильный n-угольник
        i. Квадрат
        ii. Треугольник

Разработанная программа должна рассчитывать периметр и площадь заданной фигуры. Также необходимо переопределить метод ToString(), который будет выводить информацию о фигуре в виде:

Фигура {Название}. Цвет {цвет}. Периметр {периметр}. Площадь {площадь}.