Отношения между классами. Клиенты и наследники.

Лекция 13

Наследование

- **Наследование** — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью.

Наследование

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

Наследование

- Класс, члены которого наследуются, называется *базовым классом*, а класс, который наследует эти члены, называется *производным классом*. Производный класс может иметь только **один** непосредственный базовый класс.

Наследование

- [атрибуты]
  [модификаторы] **class** Имя_класса : [родитель]
  {
  }

Конструктор базового типа

- Когда конструкторы определяются как в базовом, так и в производном классе, процесс построения объекта усложняется, поскольку должны выполняться конструкторы обоих классов.

Конструктор базового типа

Необходимо обращаться к ключевому слову base, которое находит двоякое применение:

- для вызова конструктора базового класса;
- для доступа к члену базового класса, скрывающегося за членом производного класса.

Конструктор базового типа

```
конструктор_производного_класса(список_параметров) : base(список_аргументов)
{
// тело конструктора
}
```

```
namespace ConsoleApplication1
{
    class MyClass
    {
        public int x, y, z;
        // Конструктор базового класса
        public MyClass(int x, int y, int z)
        {
            this.x = x;
            this.y = y;
            this.z = z;
        }
    }
```

```
class ClassA : MyClass
   {
      int point;

      // Конструктор производного класса
      public ClassA(int point, int x, int y, int z)
         : base(x, y, z)
      {
         this.point = point;
      }

      public void Pointer(ClassA obj)
      {
         obj.x *= obj.point;
         obj.y *= obj.point;
         obj.z *= obj.point;
         Console.WriteLine("Новые координаты объекта: {0} {1}
{2}", obj.x, obj.y, obj.z);
      }
   }
```

```
class Program
    {
        static void Main()
        {
            ClassA obj = new ClassA(10, 1, 4,
3);
            Console.WriteLine("Координаты
объекта: {0} {1} {2}", obj.x, obj.y, obj.z);
            obj.Pointer(obj);
            Console.ReadLine();
        }
    }
}
```

Наследование и сокрытие имен

- В производном классе можно определить член с таким же именем, как и у члена его базового класса. В этом случае член базового класса скрывается в производном классе. Если член базового класса требуется скрыть намеренно, то перед его именем следует указать **ключевое слово new.**

Наследование и сокрытие имен

```
 class MyClass    {
public int x = 10, y = 5, z = 6;
}
class ClassA : MyClass    {        //
   Скрываем члены класса MyClass
public new int x = 12, y = -2, z = -5;          }
```

Абстрактные и виртуальные методы

- Когда базовый класс объявляет метод как виртуальный, производный класс **может переопределить** метод с помощью своей собственной реализации.
- Если базовый класс объявляет член как абстрактный, то этот метод **должен быть переопределен** в любом неабстрактном классе, который прямо наследует от этого класса.

Абстрактные и виртуальные методы

- Если производный класс сам является абстрактным, то он наследует абстрактные члены, не реализуя их.
- Абстрактные и виртуальные члены являются основой для полиморфизма.

Абстрактные базовые классы

- Можно объявить класс как абстрактный, если необходимо предотвратить прямое создание экземпляров с помощью ключевого слова new. При таком подходе класс можно использовать, только если новый класс является производным от него.

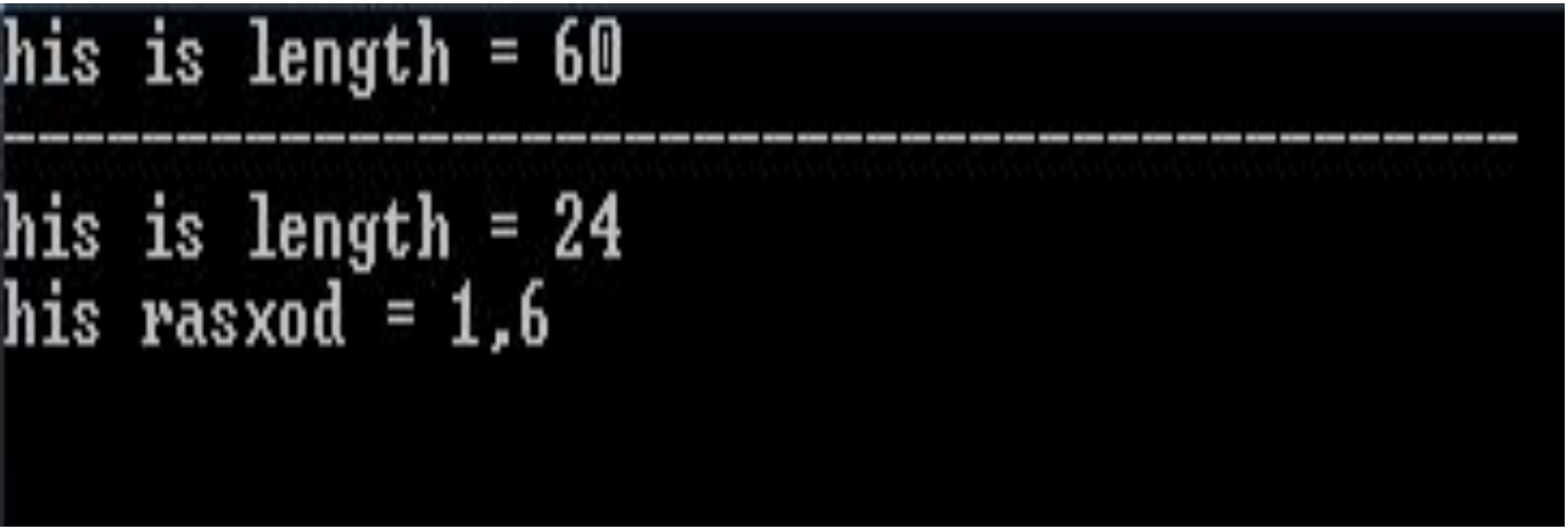
```
ссылка 1
abstract class Mexanithm
{
    ссылок 5
    abstract public void Move(int time);
}
```

```
ссылок 5
class Avto:Mexanithm
{
    public double v;
    ссылок 2
    public Avto(double v)
    {
        this.v = v;
    }
    ссылок 5
    public override void Move(int time)
    {
        Console.WriteLine("his is length = {0}", v * time);
    }
}
```

```
class Moscvich:Avto
{
    double RasxodNa100;
    ссылка 1
    public Moscvich(double RasxodNa100, double v):base(v)
    {
        this.RasxodNa100 = RasxodNa100;
    }

    ссылок 5
    public override void Move(int time)
    {
        Console.WriteLine("----------------------------------------");
        base.Move(time);
        Console.WriteLine("his rasxod = {0}", v*time/RasxodNa100);

    }
}
ссылок 0
class Program
{
    ссылок 0
    static void Main(string[] args)
    {
        Avto avt = new Avto(10);
        Moscvich mock = new Moscvich(15, 3);
        avt.Move(6);
        mock.Move(8);
        Console.ReadKey();
```


his is length = 60
his is length = 24
his rasxod = 1,6

Доступ к членам базового класса из класса-наследника

- Мы можем получить доступ к членам базового класса которые объявлены как public, protected, internal и protected internal. Члены базового класса с модификатором доступа private также переходят в класс-наследник, но к ним могут иметь доступ только члены базового класса.

Задача

Разработать программу с использованием наследования классов и виртуальных функций реализующую классы:

- –графический объект;(площадь = 0)
- –круг;
- –квадрат.

В каждом объекте должен быть метод вывода на экран площади и ***координат(реализовать с использование виртуальных функций).*** Создайте массив из ссылок на базовый класс. Инициализируйте элементы массива различными объектами и выведите на экран их площадь и координаты.

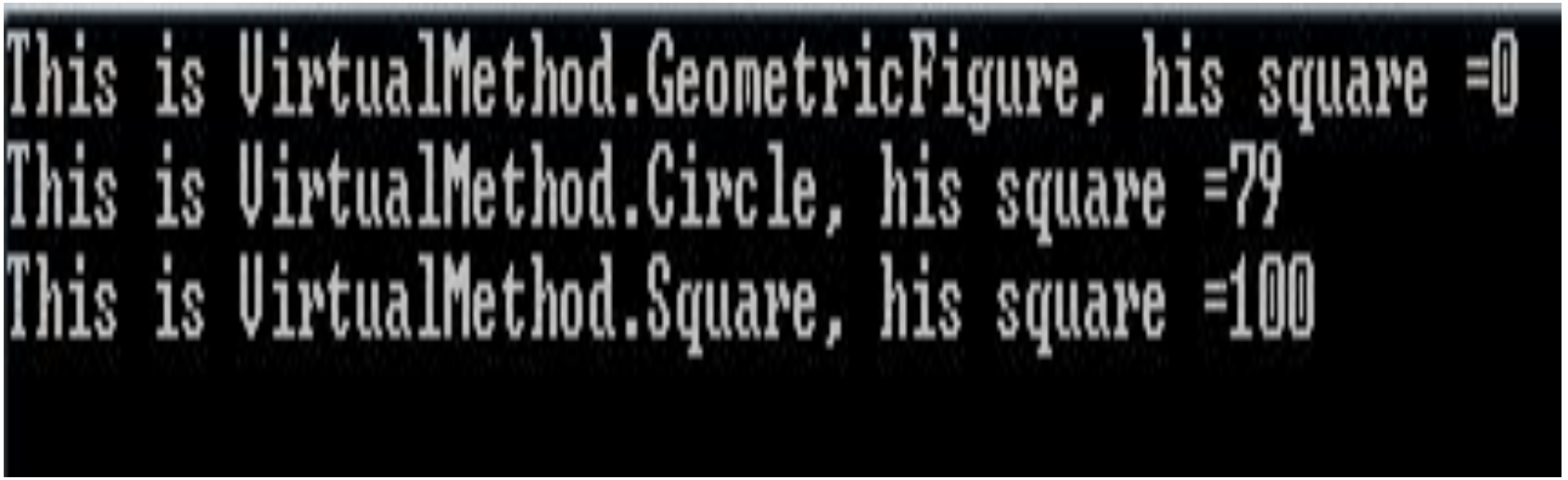
```
class GeometricFigure
{
    ссылок 3
    public virtual void  PrintSquare()
    {
        Console.WriteLine("This is {0}, his square ={1}", this.GetType(), 0);
    }

}
```

```
class Circle:GeometricFigure
{
    int r;
    ссылка 1
    public Circle(int r) { this.r = r; }
    ссылок 3
    public override void PrintSquare()
    {
        Console.WriteLine("This is {0}, his square ={1:##}", this.GetType(), Math.PI*r*r);
    }
}
```

```
class Square:GeometricFigure
{
    int Side;
    ссылка 1
    public Square(int Side) { this.Side = Side; }
    ссылок 3
    public override void PrintSquare()
    {
        Console.WriteLine("This is {0}, his square ={1:##}", this.GetType(), Side*Side);
    }
`
class Program
{
    ссылок 0
    static void Main(string[] args)
    {
        GeometricFigure[] figures = new GeometricFigure[3];

        figures[0] = new GeometricFigure();
        figures[1] = new Circle(5);
        figures[2] = new Square(10);
        foreach (var figure in figures)
            figure.PrintSquare();
        Console.ReadKey();
    }
}
```



Домашнее задание.

Дописать решение к этой задаче.

Решить задачу 8 из аккредитационных.