

Лабораторная работа 5. Наследование.

Наследование один из принципов объектно-ориентированного программирования наряду с полиморфизмом и инкапсуляцией. Говоря о наследовании, имеют в виду, наследование одним классом (дочерним) свойств другого класса (родительского) с возможностью добавления новых полей, свойств или методов или изменением родительских полей, свойств, методов.

Например класс Прямоугольник, может наследовать класс Квадрат, но иметь в дополнение еще одну сторону В.

Принципы наследования на примерах:

1. При объявление Наследующего класса, после имени наследуемого, указывают **через двоеточие** Наследуемый класс:

```
class Kvadrat
{
    //все о КВАДРАТЕ
}

class Pryamougolnik:Kvadrat
{
    //+все о ПРЯМОУГОЛЬНИКЕ
}
```

2. Наследующий класс может использовать ВСЕ поля наследуемого класса, если они объявлены как **public** или как **protected**.

```
class Kvadrat
{
    public int a;
}
class Pryamougolnik:Kvadrat
{
    public int b;
}
class Program
{
    static void Main(string[] args)
    {
        Pryamougolnik AB = new Pryamougolnik();
        AB.a = 5;    // поле a наследуется от класса квадрат
        AB.b = 10;   // поле b задано в прямоугольнике явно
    }
}
```

3. При создании дочернего объекта **ПЕРВЫМ** выполняется конструктор родителя, а затем – **ВТОРЫМ**, конструктор дочернего объекта. Проанализируйте код (при необходимости *наберите* его) и подумайте, какие числа будут выведены на экран в результате работы программы? Почему?

```
class Kvadrat
{
    public int a = 11;
    public Kvadrat()
    {
        a = 10;
    }
}
```

```

class Pryamougolnik : Kvadrat
{
    public int b = 22;
    public Pryamougolnik()
    {
        b = 20;
    }

    public void soobshit()
    {
        Console.WriteLine(a.ToString());
        Console.WriteLine(b.ToString());
    }
}

class Program
{
    static void Main(string[] args)
    {
        Pryamougolnik x = new Pryamougolnik();
        x.soobshit();
        Console.ReadLine();
    }
}

```

4. При наследовании класса, если конструктор класса родителя использует какие-либо параметры, то можно эти параметры передать дочернему классу. Для вызова родительского конструктора используется ключевое слово **base**. Например, в конструкторе родительского класса Квадрат одна переменная, а в конструкторе дочернего класса Прямоугольник две переменных, причем одна из этих переменных такая же, как и у родительского конструктора. С помощью **base** можно указать какую же из двух переменных следует использовать в конструкторе родительского класса.

```

class Kvadrat
{
    protected int a = 11;
    public Kvadrat(int x)
    {
        a = x;
    }
}

class Pryamougolnik : Kvadrat
{
    protected int b = 0;
    public Pryamougolnik(int x, int y) : base(x)
    {
        b = y * x;
    }

    public void soobshit()
    {
        Console.WriteLine(a.ToString());
        Console.WriteLine(b.ToString());
    }
}

```

```

class Program
{
    static void Main(string[] args)
    {
        Pryamougolnik x = new Pryamougolnik(2, 10);
        x.sobshit();
        Console.ReadLine();
    }
}

```

Что будет выведено на экран в результате работы программы? Что произойдет если поменять в выражении *base (x)* переменную *x* на *y*?

5. Поля, свойства и методы наследуемого класса могут иметь такое же имя как и имена класса-родителя, но отличаться например типом данных. Для того чтобы была возможно описать переменную с другим типом используют ключевое слово **new**. В случае если необходимо обратиться к полю, которое имеет одинаковое имя у родительского и дочернего класса используют префикс **this** – для этого класса и **base** – для родительского класса.

```

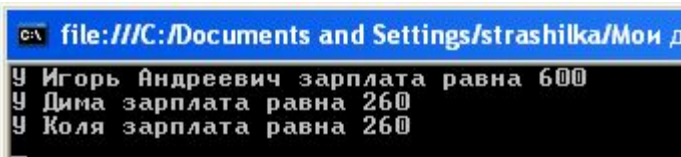
class Kvadrat
{
    protected int a = 11;
    public int c = 33;
    public Kvadrat(int x)
    {
        a = x;
    }
}
class Pryamougolnik : Kvadrat
{
    protected int b = 22;
    new protected double c = 4.5; // переменная имеет другой тип данных!!!
    public Pryamougolnik(double y, int x) : base(x)
    {
        c = y;
    }

    public void sobshit()
    {
        Console.WriteLine(a.ToString()); //Значение поля a установленное ...
                                           //...конструктором класса Квадрат
        Console.WriteLine(b.ToString()); //Значение поля b, установленное
                                           //...при объявлении переменной ...
                                           //...внутри класса Прямоугольник
        Console.WriteLine(c.ToString()); //Значение поля c, установленное...
                                           //...в конструкторе класса Прямоугол
        Console.WriteLine(base.c.ToString()); //Значение поля c, ...
                                           //...установленное в классе Квадрат
        Console.WriteLine(this.c.ToString()); //Значение поля c, установленное...
                                           //в конструкторе класса Прямоугольни
    }
}
class Program
{
    static void Main(string[] args)
    {
        Pryamougolnik x = new Pryamougolnik(7.777, 6);
        x.sobshit();
        Console.ReadLine();
    }
}

```

Какие числа будут выведены на экран, и в каком порядке?

1. Создать класс Рабочий с двумя полями: а) защищенное поле Имя и б) публичное поле Зарплата.
2. Создайте Колю и Диму как экземпляры класса Рабочий.
3. Унаследуйте от класса Рабочий класс Начальник (На данном этапе с полностью идентичными свойствами).
4. Создайте Игоря Андреевича, как экземпляр класса Начальник.
5. Добавьте в класс Рабочий конструктор. Передайте в конструктор один параметр – имя рабочего. Тут же в конструкторе установите начальную заработную плату рабочему в 250 грн.
6. Если в конструкторе родительского класса Рабочий есть параметр, то он должен быть и в конструкторе дочернего класса Начальник. Поэтому добавьте в класс Начальник конструктор. Но начальную заработную плату установите в размере 500 грн. Не забудьте использовать ключевое слово base (см. пример 4).
7. Задайте параметры при вызове конструктора создания Коли, Димы и Игоря Андреевича :) . Запустите.
8. Добавьте в класс Рабочий закрытое поле ЧислоЛетДоПенсии. Число лето до пенсии задайте в конструкторе равным 40.
9. Добавьте в класс Рабочий общедоступный метод Работать. Который будет на один уменьшать число лет до пенсии, а так же на 10 увеличивать зарплату рабочего.
10. Добавьте теперь в класс Начальник метод Работать. В этом методе не трогайте переменную ЧислоЛетДоПенсии, а заработную плату увеличивайте на 10;
11. Заставьте работать всех трех сотрудников.
12. Добавьте (подумайте сами куда) такой метод, что бы он выводил информацию о текущей заработной плате сотрудника в таком виде:



```
C> Игорь Андреевич зарплата равна 600
C> Дима зарплата равна 260
C> Коля зарплата равна 260
```

13. Добавьте в класс Рабочий метод Премия, который будет генерировать случайное число в диапазоне от 50 до 100 и прибавлять его к заработной плате. Вызовите этот метод из основной программы.
14. Для класса Начальник переопределите этот метод так, что бы он сгенерированное число умножал на 3, а затем добавлял его к заработной плате.
15. Добавьте в класс Рабочий метод штраф так, что бы он был доступен только классу рабочий и не был применим к классу Начальник.
16. Добавьте Начальнику метод Установить заработную плату (с одним параметром).
17. А классу Рабочий добавьте метод ВернутьЗарботнуюПлату. Метод должен будет возвращать текущее значение заработной платы, уменьшенное вдвое ;).
18. Используя два последних метода установите заработную плату Начальник втрое больше заработной платы Рабочего...