Основы программирования

Основы ООП

Введение в классы

...

Класс — это шаблон, определяющий форму объекта. Он задает как данные, так и код, который оперирует этими данными.

В C# *классы* используются для создания *объектов.*

Объекты — это *экземпляры* класса.

Таким образом, *класс* — это множество намерений (планов), определяющих, как должен быть построен объект.

...

Класс — это *логическая абстракция.* О ее реализации нет смысла говорить до тех пор, пока не создан объект класса, и в памяти не появилось физическое его представление.

Методы, поля, свойства, индексаторы, операторы, … составляющие класс, называются *членами* класса.

Члены класса (состав класса)

Члены данных (характеристики, свойства):

- ***поля*** (переменные);
- ***константы***;
- ***статические поля/переменные***.

Методы-члены (поведение):

- ***методы*** (функции);
- ***конструкторы*** и деструкторы;
- ***свойства*** и ***индексаторы***;
- ***операторы;***
- события.

Могут также быть описаны вложенные классы.

Члены класса (состав класса)

```
class Room
    {
        public double length;
        public double height;
        public double width;
        public bool windows;

        public double SRoom(double x, double y)
        {
            return x * y;
        }

        public double VRoom(double x, double y, double z)
        {
            return x * y * z;
        }
    }
```

Члены класса (состав класса)

```
class Program
{
    static void Main(string[] args)
    {
        Room myRoom = new Room();
        myRoom.length = 7;
        myRoom.height = 2.5;
        myRoom.width = 4;
        myRoom.windows = true;
        Console.WriteLine("Объем комнаты равен = " + myRoom.VRoom(myRoom.length,myRoom.hei
        Console.WriteLine("Площадь комнаты равна = " + myRoom.SRoom(myRoom.length, myRoom.
        if (myRoom.windows) Console.WriteLine("В комнате имеются окна."); else Console.Wri
        Console.ReadLine();
    }

}
```

Синтаксис описания класса

```
атрибуты модификаторы
            class Имя_класса : предки
{
      //тело класса
}
```

Имя класса составляется по нотации Паскаля.

Синтаксис описания класса

```
//объявление полей экземпляров:
модификаторы тип имя_переменной;

//объявление методов:
модификаторы тип_возврата Имя_метода (параметры)
        {
                // тело метода
        }
```

Атрибуты класса

Задают дополнительную информацию о классе и будут рассматриваться по мере необходимости.

Модификаторы класса (или спецификаторы)

- Модификаторы доступа указывают доступность класса из других классов программы.
- Модификаторы задают для класса некоторые свойства.

При описании класса могут быть указаны несколько модификаторов – простых и доступа.

Модификаторы доступа

Все члены класса (поля, методы, свойства, …)
имеют модификаторы доступа.

Модификаторы доступа позволяют задать допустимую область видимости для членов класса. То есть контекст, в котором можно употреблять данную переменную или метод.

Модификаторы доступа

- `public` – общедоступный член класса или класс. Доступен из любого места в коде, а также из других программ и сборок.
- `private` – закрытый член класса или класс. Доступен только из кода в том же классе.
- `protected` – доступен из любого места в текущем классе или в производных классах (потомках), которые могут располагаться в других сборках.

Модификаторы доступа

- `internal` – доступны в любом месте той же сборки (сравните с модификатором `public`).
  Это значение по умолчанию.
- `protected internal` – совмещает функционал двух модификаторов. Доступны из самого класса, производных классов и текущей сборки.
- `private protected` – такой член класса доступен из любого места в текущем классе или в производных классах, которые определены в той же сборке.

Модификаторы доступа и инкапсуляция

Благодаря такой системе модификаторов доступа можно скрывать некоторые моменты реализации класса от других частей программы.

Такое сокрытие и называется *инкапсуляцией*.

Модификаторы

- `const` – для объявления констант. Константы не являются переменными и не могут быть изменены.
- `new` – задает новое описание взамен унаследованного от предка явным образом скрывая члены, унаследованные от базового класса. При этом производная версия заменяет версию базового класса;
- `static` – используется для объявления статического члена, принадлежащего собственно типу, а не конкретному объекту;

Модификаторы

- `abstract` – абстрактный. Указывает, что элемент имеет отсутствующую или неполную реализацию;
- `sealed` – запрещает наследование от данного класса;
- `override` – используется для расширения или изменения реализации унаследованных методов-членов;
- …

Создание экземпляров класса

- *неявным* (для системных классов при первом обращении к ним);
- *явным* образом (программистом с помощью операции `new` – операции выделения динамической памяти для хранения объекта).

Результатом выполнения операции является ссылка на созданный объект:

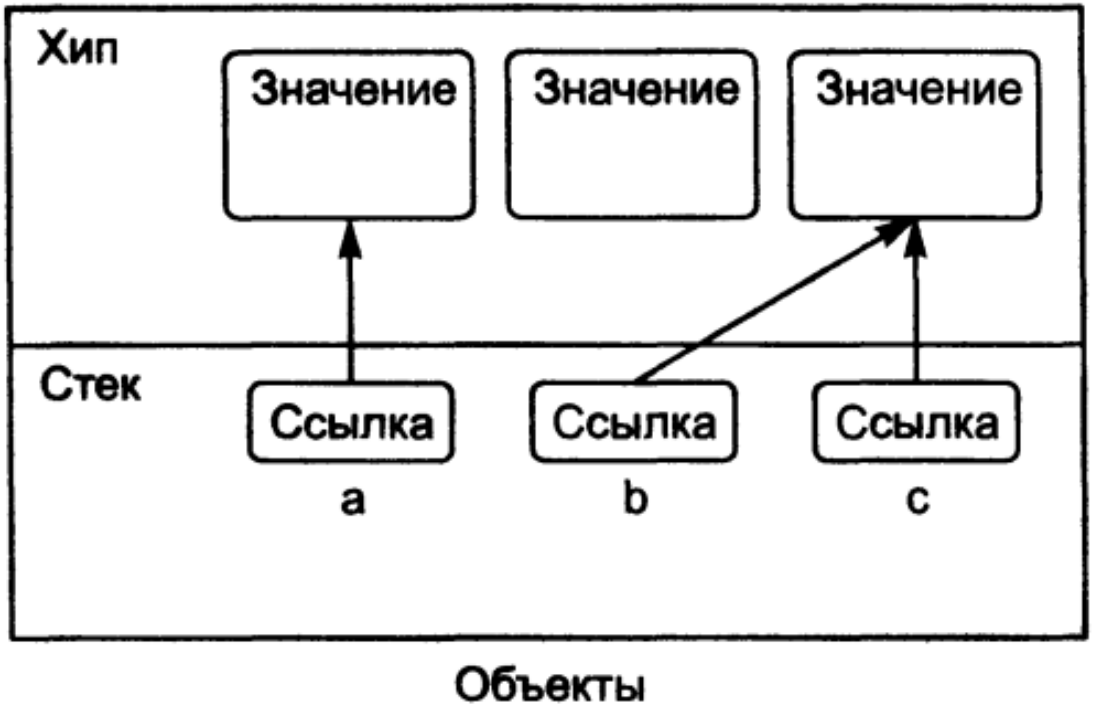
```
MyClass c = new MyClass();
```

Операции присваивания и сравнения объектов

- при присваивании копируется ссылка на объект, а не сам объект
- при сравнении объектов сравниваются ссылки на них. Ссылки равны, если они указывают на один и тот же объект.

```
MyClass c = new MyClass();
MyClass d = new MyClass(); // c ≠ d
MyClass e = c; // c = e
```

Операции присваивания



Состав класса

- поля – состояние объекта;
- константы – поля с неизменяемым значением, как правило, связаны с классом вообще, а не с отдельными экземплярами;
- методы – описание поведения класса;
- конструкторы – инициализаторы экземпляра класса;
- свойства – специальные методы доступа к полям класса;
- индексаторы – специальные методы индексированного доступа к полям класса;

Состав класса

- операции – действия над объектами класса в символах операций;
- деструктор – описание действий, которые необходимо выполнить перед удалением объекта;
- события – уведомления, генерируемые объектами класса, применяются в событийном программировании;
- вложенные классы – классы, описанные внутри класса.