C#
IV
OOP

Types

План

1. Классы (типы)

2. Поля, методы

3. Модификаторы доступа

Классы (типы)

```
public class Yabloko // вместо public может быть internal
{
        public Yabloko(ConsoleColor color = ConsoleColor.Green) // конструктор
        {

        }
}
```

```
// Объявляем переменную типа яблоко
Yabloko yabl; // сейчас она равна null

// Создаем объект типа яблока и кладем его в переменную yabl
yabl = new Yabloko(ConsoleColor.Red); // теперь у нас есть красное яблоко
```

Поля

- Определяют свойства объекта. Его состояние в данный момент.
- Могут изменяться в результате работы методов.

```
public class Yabloko // вместо public может быть internal
{
        private ConsoleColor color; // это поле. Оно описывает состояние – цвет
                                    // яблока. Впоследствии яблоко может,
                                    // например, сгнить и изменить свой цвет
       // конструктор (задание свойств, выполнение нек-ых действий при создании объекта)
        public Yabloko(ConsoleColor color = ConsoleColor.Green) {
            // Задаем цвет яблоку полученному из конструктора
            this.color = color; // this - указывает на объект
        }
}
```

Модификаторы доступа

Могут применяться к полям, методам и свойствам и определяют, возможен ли доступ к данным элементам извне.

public	Доступно вне класса
private	Доступно только из класса
protected	Не доступно вне класса, но доступно в наследующих данный класс типах

Методы (действия)…

- Методы (аналог, функции) – действия которые совершаются над объектом, вызывая, возможно, изменение его состояния (полей) и / или возвращают значение вычисленное на основе состояния объекта.
- Название должно быть глаголом.
- Метод должен быть коротким и выполнять одно действие

Синтаксис рассматривался в первой лекции.

…перегрузка методов

Одинаковые имена, но разные сигнатуры

Сигнатура:

- Имя метода
- количество и тип(ы) аргументов
- и их последовательность

```
int Multiply(int n1, int n2) { return n1 * n2; }

// Отличается количеством аргументов
int Multiply(int n1, int n2, int n3) { return n1 * n2 * n3; }

// Отличается типом аргументов
double Multiply(double n1, double n2) { return n1 * n2; }
// ошибка. Сигнатура одинакова с первым методом
double Multiply(int n1, int n2) { return n1 * n2; }
// (возвращаемое значение не входит в сигнатуру)
```


Спасибо за внимание!