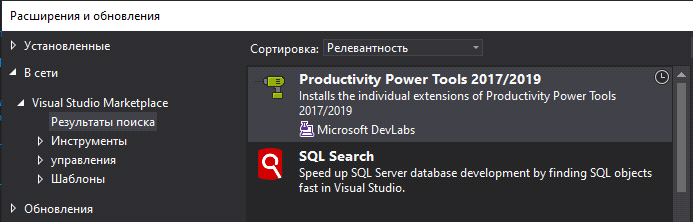
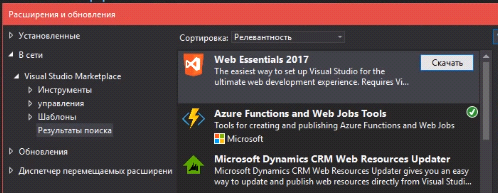
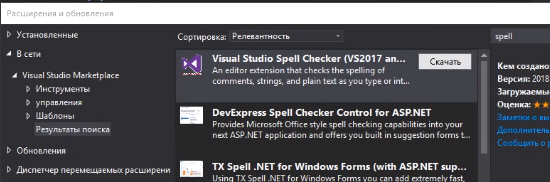
Полезные расширения: 

Для разработки в ВЕБ:



Проверяет грамматику в коде:



- интелисенс – отображает подсказки

**Переменная** – именованная область памяти.

**Stack** (значимые типы) - быстро, но мало

**Heap** (ссылочные типы) - медленно, но много – объем оперативной памяти

**Функциональный стиль** –несколько команд объединяются в именованную функцию, которую можно было вызвать в любом другом участке кода.

**Процедурный стиль** – последовательная запись кода (грубо говоря).

**Принцип черного ящик** – подаем входные данные, не видим внутреннюю часть (то, как работает программа) и получаем готовый результат. Так должны работать классы.

*Пример*: микроволновка. Засунули еду – получили разогретую.

**Принцип белого ящика** – мы вручную настраиваем программу, видим ее «внутренности».

*Пример*: микроволновка. Засунули еду, подрегулировали значения, силу разогрева, кол-во всяких микроволн и прочее, получили результат.

## Парадигмы ООП

**Инкапсуляция** *(сокрытие)* – предоставление только тех инструментов, которыми должен пользоваться пользователь.

**Наследование** – создает зависимость между двумя классами (как бы объединяя их). От базового класса передаем все свойства классу наследнику.

**Полиморфизм** – преобразование класса наследник в базовый класс, откинув все лишние свойства.

**Абстракция/абстрагирование** – учет только тех параметров, которые нужны конкретно в нашей объектной области.

**Посылка и обработка сообщения** – при совершении определенного события, происходит вывод/обработка других событий(-ия).

**Повторение кода** – не изобретаем велосипед, не пишем кода 100500 раз один и тот же, а используем уже существующие библиотеки, функции, методы и тд. (Don’t repeat your self).

//public - публичный открытый - открыт всем и всегда

//internal - открытый в пределах проекта - доступен только в одном проекте

//protected - защищенный - доступный только унаследованному классу от базового, остальным - нет

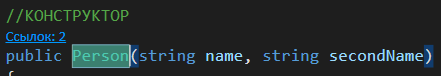
//private - закрытый - принадлежит ТОЛЬКО этому классу

**Экземпляр класса** (англ. instance) — это описание конкретного объекта в памяти. Класс описывает свойства и методы, которые будут доступны у объекта, построенного по описанию, заложенному в классе.

## Методы, свойства, конструкторы

По стандарту все методы, св-ва, конструкторы и другие типы задаются приватными, однако для красоты и эстетики лучше дописать модификатор доступа private.

Все конструкторы создаются от имени класса:



В некоторых ЯП выделяются функции и процедуры. Функции обязательно возвращают нам какое-либо значение. Процедуры же ничего нам не возвращают. В ЯП C# у нас функции и процедуры объединены под одним общим названием - методы. Методы могут как ничего нам не возвращать, так и возвращать значения определенного типа данных.

*Правила именования:* гл.+сущ.

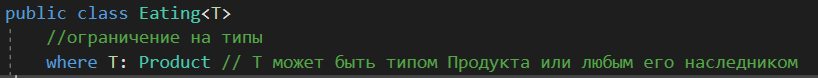
В параметры метода можно передавать аргументы, либо же не передавать их совсем.

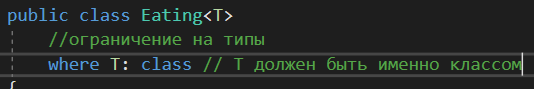
Модификатор доступа static говорит нам о том, что все методы конкретно данного класса могут выполняться без создания объекта от данного класса. Также статические методы не могу работать с нестатическими переменными.

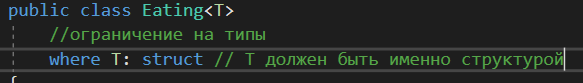
## Перегрузки методов

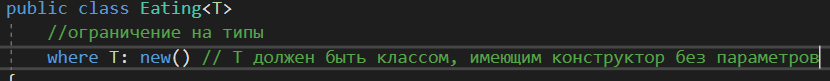
**Сигнатура** - это имя методы и принимаемые типы аргументов и их кол-во.

## Generic/Шаблоны/Универсальные типы



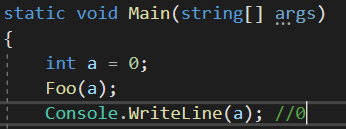


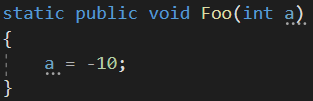




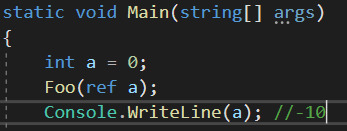
## Ref

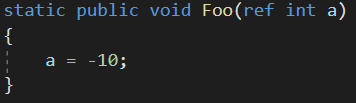
Ключевое слово служит для указания того, что в параметры методы мы будем передавать не ссылку на наши данные, а сами данные (не будем лишний раз их копировать).





В методе мы создали новый элемент (а) и передали ему значения нашей переменной в main (тоже, а). В методе мы присвоили -10 переменной из нашего параметра. После выхода из метода наша «а» в main не изменится.





В данном случае мы указываем, можно сказать, передаем ссылку на сам наш объект (не создавая новый в параметрах => экономим память) и совершаем наши операции с самой переменной («а» из main).



Есть еще *ссылочные локальные* и *возвращаемые ссылочные* значения. Но оно тебе надо, м?

## Строки

Интерполяция строки - замена заполнителей в строке численными значениями.



NULL

Null – пустая ссылка => значимые типы не могу быть null, поскольку не содержат ссылку, а тупа лежат в стеке.

Null можно использовать только с ссылочными типами.

Автоматически причисляется всем объектам класса (ссылочным типам).

Если же мы инициализируем (например) массив, а после присваиваем ему null, то связь с данными в управляемой куче обрывается, и сборщик мусора очищает данные, чтобы не засорять оперативную память.

## Interface (доступно на обзор)

*Правило именования*: I (заглавная «ай») + название (прим.: IPerson, IDesktop).

Интерфейсы полностью публичные, не имеют полей и не могут быть приватными. Интерфейсы не наследуются! Интерфейсы *могут реализовываться* или *имплементироваться* (имплементировать – это реализовывать в ПО какой-либо функционал). Интерфейсы желательно использовать, когда задействованы несколько объектных областей (пример: машина, человек, велосипед могу передвигаться => уже можно реализовать для них интерфейс).

## Паттерны (шаблоны)

Singelton

## Отличие SDK от API

**API** (Application programming interface) - это описание интерфейса чего-либо. Набор правил, по которым что-то должно работать.

**SDK** (Software development kit) - это набор инструментов для работы с чем-то.

Вы можете рассматривать свой любимый язык программирования как API (набор правил синтаксиса и поведения), а свою любимую среду - как SDK (набор инструментов для написания, проверки и отладки кода).

**SDK** - это набор функционала(библиотек) и утилит для разработки. Собственно, **SDK** и предоставляет реализацию некоторого **API**, это оболочка **API's**, которая упрощает работу для разработчиков.

**API:**

набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением для использования во внешних программных продуктах

это **интерфейс**, похоже на спецификацию телефонной системы или электропроводки в вашем доме.

это список того, что можно вызывать и какого ждать результата.

**SDK**:

пакет реальных инструментов внедрения. Это, как комплект, который позволяет вам подключиться к телефонной системе или электрической проводке.

это библиотеки, в которых реализованы вызываемые функции + файлы необходимые для подключения этих библиотек