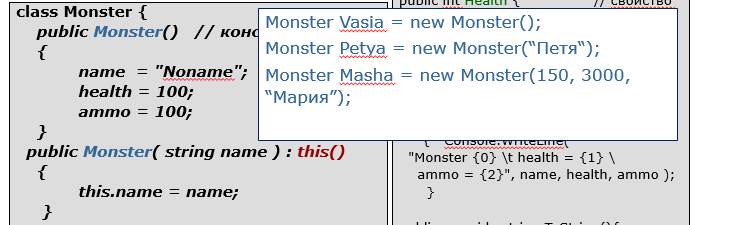
Конструктор – вид метода, предназначенный для инициализации объекта или класса.

Конструктор экземпляра инициализирует данные экземпляра, а конструктор класса – данные класса.

Конструктор вызывается автоматически при создании объекта класса с помощью операции new, имя конструктора совпадает с именем класса.

Свойства конструктора:

* Конструктор не возвращает значение;
* Класс может иметь несколько конструктор с разными параметрами для различных видов инициализации;
* Если программист не указал ни одного конструктора или поля небыли инициализированы, то значимые = 0, а ссылки = NULL;
* Конструктор по умолчанию – конструктор без параметров.



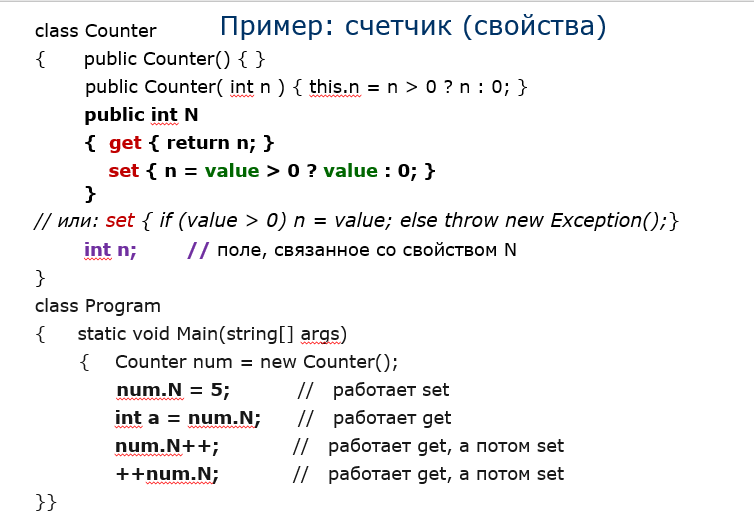
Благодаря public Monster(string name) **:this(); -**  конструктор наследует все поля по умолчанию (если они небыли изменены)

Свойства:

Свойства служат для организации доступа к полям класса. Как правило, свойство определяет методы доступа к закрытому полю.

При обращении к свойству автоматически вызываются указанные в нем блоки чтения (set) и установки (set).

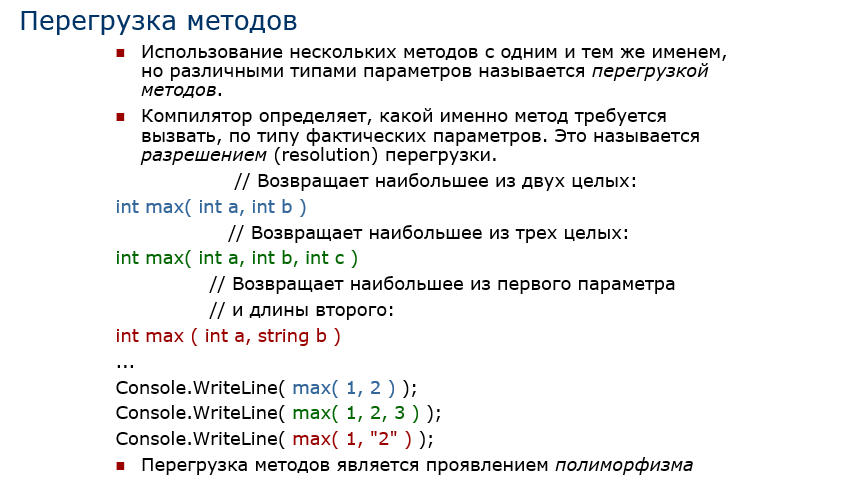
Может отсутствовать либо часть get, либо set, но не обе одновременно. Если отсутствует часть set, свойство доступно только для чтения (read-only), если отсутствует get - только для записи (write-only).



/\*get и set – де-факто очень стрёмные функции\*/

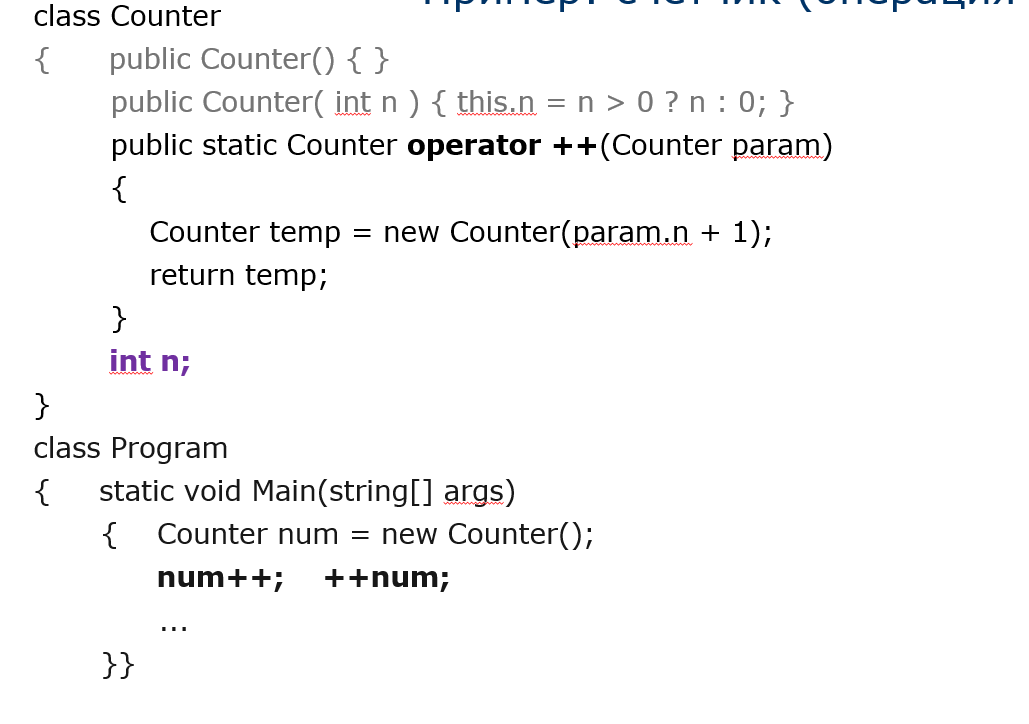
Int a = Vasya.Name – Работает get

Vasya.Name = “aaa” – работает set



**/\*Короче, перегрузка как в плюсах\*/**

**ОЧ важно! Перегрузка методов – проявление полиморфизма**

****

**Це было «перегрузка» операндов.**

**В данном примере, мы берём и посылаем в функцию «++» значение num. Прикол в том, что в данной функции мы как переменную используем экземпляр класса Counter (имя данного экземпляра – param). В связи с этим мы обращаемся к n, как к экземпляру param, и возвращаем значение через временный экземпляр temp.**

Суть в том, что мы может изменить любой операндов (++, --, =), тем самым заставив делать операнд другую херь (напр, ++ будет делать не +1, а +2)

Тут есть ключевое слово **operator,** после которого указывается сам операнд

Windows form.

**Windows приложения -**  имеют стандартный интерфейс, и управляются событийно.

Модальная и немодальная форма:

* Модальная – не позволяет пользователю переключится пользователю на другие окна этого приложения, пока не будет завершена работа с текущим окном (диалоговые окна, окна ообщений)
* Немодальная форма – позволяет переключиться на другие окна того же приложени.