Лекция № 13

ITEA

IT Education Academy

WWW.ITEA.UA

C# Base

Урок 13

Сборщик мусора

План урока

- Понятие сборки мусора
- Неуправляемая и управляемые кучи
- Деструктор
- Класс System.GC
- Интерфейс Idisposable
- Оператор using

Сбор мусора

Сбор мусора — это симуляция бесконечной памяти на машине с конечной памятью.

Специальный механизм, называемый сборщиком мусора **(garbage collector)**, периодически освобождает память, удаляя объекты, которые уже не будут востребованы приложением—то есть производит «сбор мусора».

Сборка мусора была в первые применена Джоном Маккарти в 1959 году в среде программирования на разработанном им функциональном языке программирования Lisp. Впоследствии она применялась в других системах программирования и языках, преимущественно—в функциональных и логических.

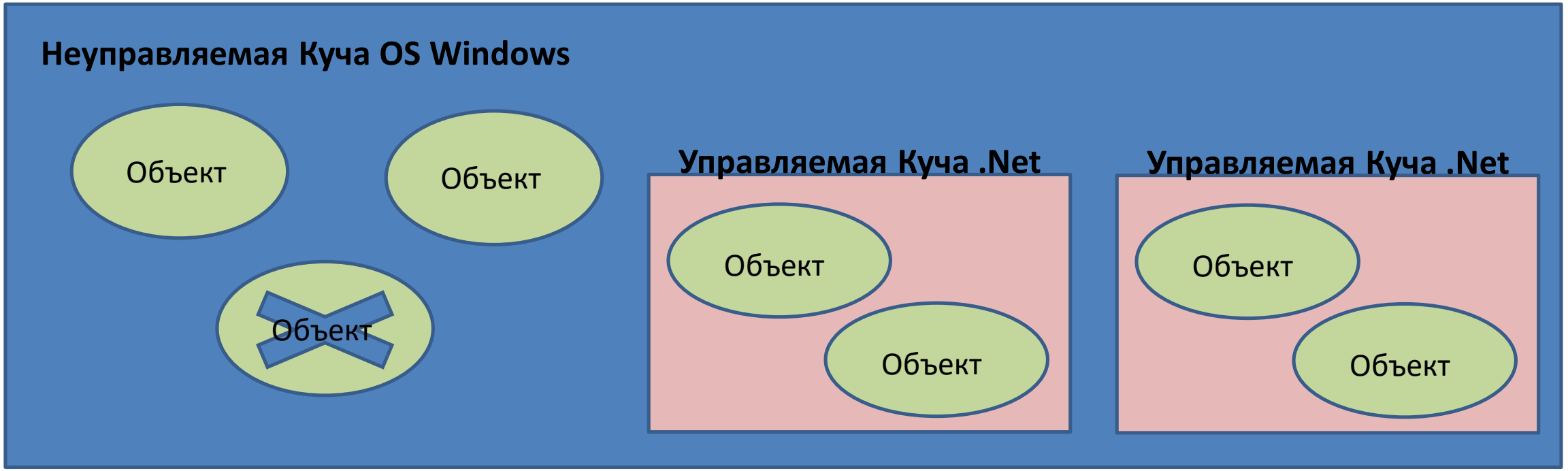
Неуправляемая и управляемые кучи

Программистам на C# никогда не приходится непосредственно удалять управляемый объект из памяти (в языке C# нет даже ключевого слова вроде delete). Вместо этого объекты .NET размещаются в области памяти, которая называется **управляемой кучей (managed heap),** откуда они автоматически удаляются сборщиком мусора, когда наступает "определенный момент в будущем".

Часть объектов связана с операционной и файловой системами, с WinAPI, с драйверами, с видеокартой, с сетью и т.д. Понятно, что все эти внешние ресурсы никак не связаны с менеджером памяти .NET и потому на них не распространяется автоматическое распределение памяти. Такая память и такие объекты **называются неуправляемыми**.

Неуправляемая и управляемые кучи

Под каждое приложение своя куча



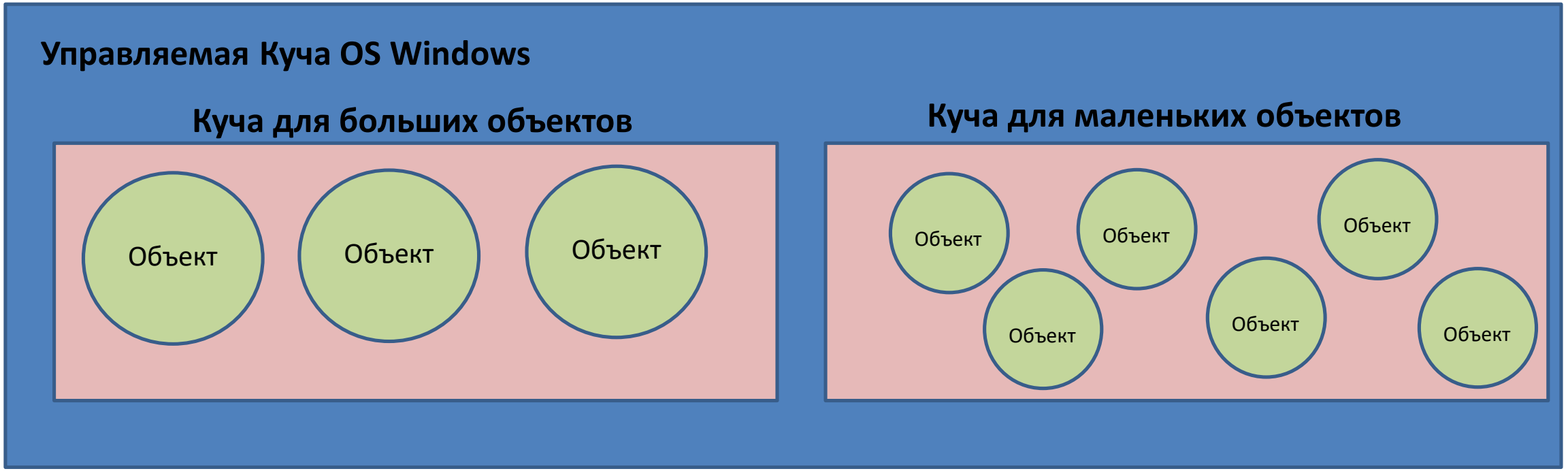
Garbage collector

Деструктор

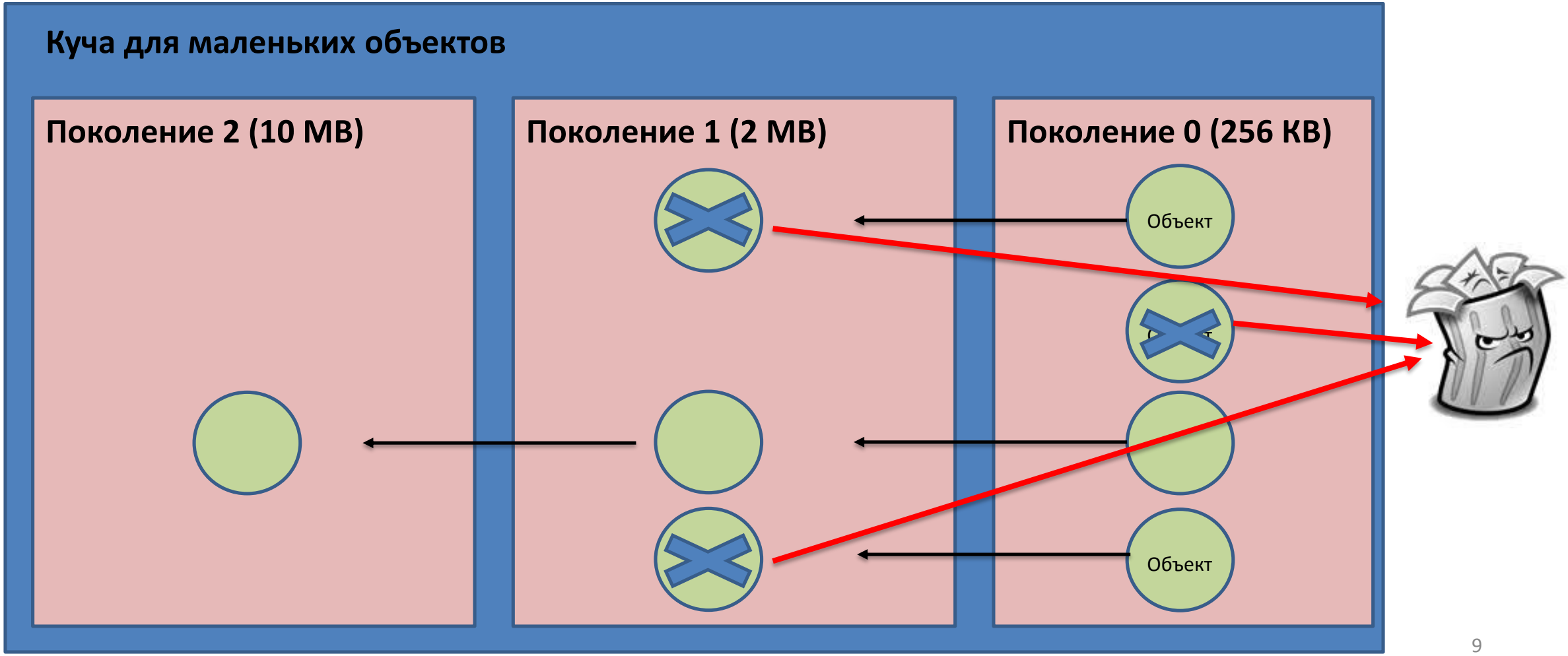
В языке C# имеется возможность определить метод, который будет вызываться непосредственно перед окончательным уничтожением объекта системой "сборки мусора". Такой метод называется деструктором и может использоваться в ряде особых случаев, чтобы гарантировать четкое окончание срока действия объекта.

```
class SomeClass
{
    // Деструктор
    ~SomeClass()
    {
        //Тело дестуктора
    }
}
```

Управляемая куча



Куча для маленьких объектов



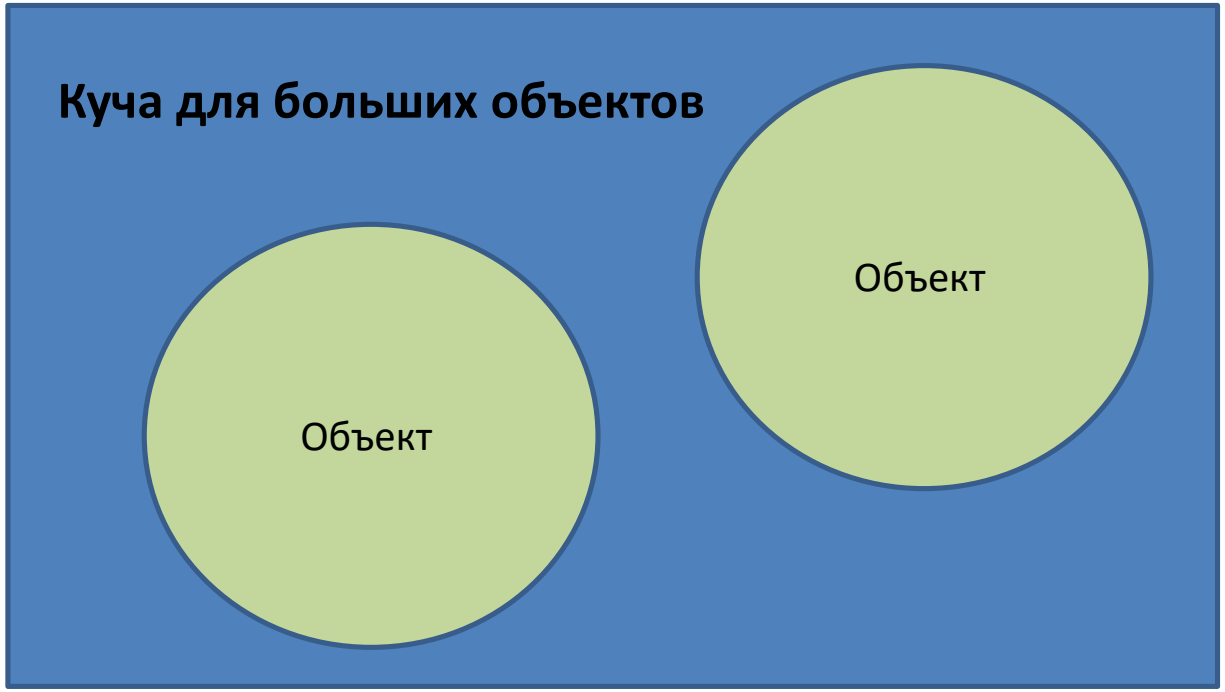
Управляемая куча

Сборщик мусора работает на основании таких предположений:

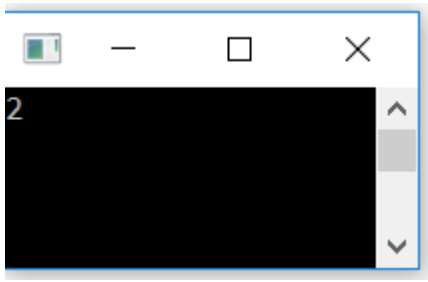
- Чем младше объект, тем короче время его жизни.
- В нулевой генерации объект живет мало.
- Чем старше объект, тем длине время его жизни.
- Сбор мусора в части кучи выполняется быстрее, чем во всей куче.
- Если есть еще место в первой генерации, тогда не выполняется сбор мусора во второй генерации, аналогично для остальных генераций, так как своей работой сборщик мусора замедляет работу приложения.

Куча для больших объектов

Любой объект размером больше 83 КВ считается большим.
Только из одного поколения, оно имеет индекс 2.
Куча для больших объектов не дефрагментируемая.




```
static void Main()
{
  Byte[] vs = new byte[85000];
  Console.WriteLine(GC.GetGeneration(vs));
}
```



Класс System.GC

Функционал сборщика мусора в библиотеке классов .NET представляет класс **System.GC.** Через статические методы данный класс позволяет обращаться к сборщику мусора. Как правило, надобность в применении этого класса отсутствует. Наиболее распространенным случаем его использования является сборка мусора при работе с неуправляемыми ресурсами, при интенсивном выделении больших объемов памяти, при которых необходимо такое же быстрое их освобождение.

- **GetGeneration(Object**) позволяет определить номер поколения, к которому относится переданый в качестве параметра объект
- **Collect** приводит в действие механизм сборки мусора. Перегруженные версии метода позволяют указать поколение объектов, вплоть до которого надо произвести сборку мусора

Интерфейс IDisposable

Интерфейс IDisposable объявляет один единственный метод Dispose, в котором при реализации интерфейса в классе должно происходить освобождение неуправляемых ресурсов.

```
public interface IDisposable
{
   void Dispose();
}
```

`void Dispose();` - метод для освобождения ресурсов.

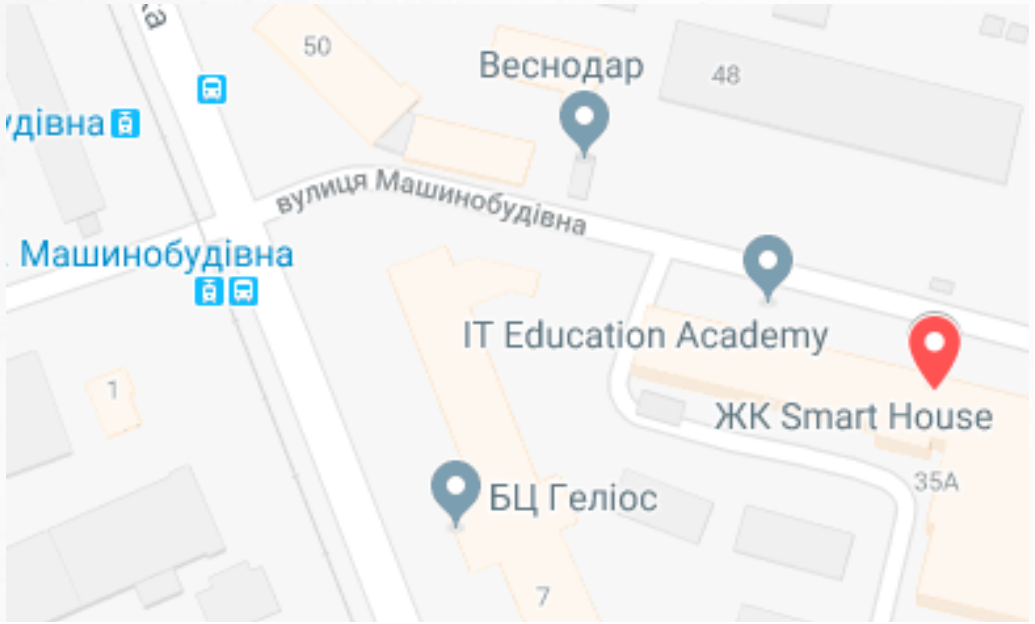
Оператор using

- Оператор using используется для создания блока кода для объектов которого гарантировано будет вызван Dispose().
- Объекты, которые создаются в блоке using обязаны реализовывать IDisposable.

```
using (MyClass myClass = new MyClass())
{
   //Использование объекта myClass

}//Неявно вызивается метод Dispose()
```


Презентация окончена.
Спасибо за внимание!



I T E A

КОНТАКТНЫЕ ДАННЫЕ

ITEA

ЖК “Smart House”, ул. Машиностроительная, 41 (м.Берестейская)

ЖК «Корона» улица Срибнокильская,1 м. Позняки

+38 (044) 599-01-79
facebook.com/Itea
info@itea.ua
itea.ua