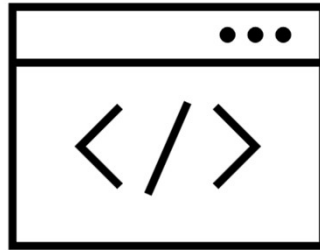


ООП

Инкапсуляция

Наследование

Полиморфизм



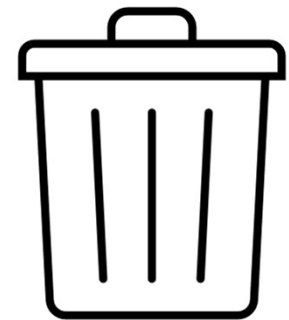
Классы. Деструкторы



КубГТУ, Кафедра «Информационных систем и программирования»

В C# существует специальный вид метода, называемый **деструктором**, который вызывается сборщиком мусора непосредственно перед удалением объекта из памяти.

Сборщик мусора удаляет объекты, на которые нет ссылок. Он работает в соответствии со своей внутренней стратегией в неизвестные для программиста моменты времени.



В деструкторе описываются действия, гарантирующие корректность последующего удаления объекта.

Например, проверяется все ли ресурсы, используемые объектом, освобождены (файлы закрыты, удаленное соединение разорвано и т. п.). Если ваш класс не используется для обеспечения доступа к какому-либо системному ресурсу (например, объекты графического контекста рисования), использование деструкторов настоятельно не рекомендуется.



Синтаксис деструктора

```
[атрибуты] [extern] ~имя_класса()  
{  
    тело_деструктора  
}
```

Деструктор:

- не имеет параметров
- не возвращает значения
- не требует указания спецификаторов доступа

Его имя совпадает с именем класса и предваряется тильдой (~), символизирующей обратные по отношению к конструктору действия. Тело деструктора представляет собой блок или просто точку с запятой.



Добавим в класс Circle следующий деструктор:

```
~Circle()
{
    Console.WriteLine("Удалена {0} с
    центром в точке ({1},{2}), радиусом
    {3}", name, x, y, radius);
}
```

Теперь посмотрим, как выполнится следующая программа:

```
using System;

namespace MyProgram
{
    class Program
    {
        static void Main()
        {
            Circle oneCircle = new Circle(1);
            oneCircle.Show();
            Circle twoCircle=new Circle(1, 1, 100);
            twoCircle.Show();
            Circle threeCircle = new Circle(5, 2, 10);
            threeCircle.Show();
        }
    }
}
```



Результат работы программы:

Окружность с центром в точке (0, 0) радиусом 1
Окружность с центром в точке (1, 1) радиусом 100
Окружность с центром в точке (5, 2) радиусом 10
Удалена Окружность с центром в точке (5, 2)
радиусом 10
Удалена Окружность с центром в точке (1, 1)
радиусом 100
Удалена Окружность с центром в точке (0, 0)
радиусом 1



Обратите внимание на то, что деструкторы были вызваны автоматически.



```

using System;
namespace MyProgram
{
    class Program
    {
        static void Main()
        {
            Circle oneCircle = new Circle(1);
            Circle twoCircle = new Circle(1, 1, 100);
            Circle threeCircle = new Circle(5, 2, 10);

            oneCircle.Show();
            twoCircle.Show();
            threeCircle.Show();
            Console.WriteLine();

            threeCircle = oneCircle;

            threeCircle.Set(-1, -1, 20);

            oneCircle.Show(); twoCircle.Show();
            threeCircle.Show();
            Console.WriteLine();
        }
    }
}

```

Вернемся к проблеме потерянных ссылок на объекты:

//создаем три объекта

//выводим информацию об объектах

// теперь threeCircle и oneCircle ссылаются на один и тот же объект
 //к объекту, представляющему окружность с центром (0, 0) и радиусом 1 не ведет ни одной ссылки

//изменяем значения по ссылке threeCircle

//вновь выводим информацию об объектах

КубГТУ, Кафедра «Информационных систем и программирования»



Результат работы программы:

Окружность с центром в точке (0, 0) радиусом 1
Окружность с центром в точке (1, 1) радиусом 100
Окружность с центром в точке (5, 2) радиусом 10
Окружность с центром в точке (-1, -1) радиусом 20
Окружность с центром в точке (1, 1) радиусом 100
Окружность с центром в точке (-1, -1) радиусом 20
Удалена Окружность с центром в точке (5, 2) радиусом 10
Удалена Окружность с центром в точке (1, 1) радиусом 100
Удалена Окружность с центром в точке (-1, -1) радиусом 20

Данный пример показывает, что ссылки `oneCircle` и `threeCircle` ссылаются на один объект.

А ссылка на окружность с центром (0, 0) и радиусом 1 потеряна. Но сборщик мусора удалит все объекты и даже те, ссылки на которые были потеряны.

