Платформа .NET и язык программирования C#

Методы.
Основы рефакторинга.

Методы

Метод — это последовательность операторов C#, оформленных заголовком и расположенных внутри фигурных скобок. Метод выполняет задачу и возвращает контроль в точку вызова, может возвращать или не возвращать значение.

```
модификаторы возвр ИмяМетода (список параметров)
{
        … /операторы метода
}
```

модификаторы — пустой, public, private, static и т. п. (полный список — в справке)

возвр — это void (метод не возвращает значения) или тип возвращаемого значения.

Параметры методов

Список параметров идет через запятую. Каждый параметр может быть:

1. **Обязательный:** ref|out типПараметра имяПараметра
2. **Необязательный:** ref|out типПараметра имяПараметра = значениеПоУмолчанию
3. **Параметр, принимающий переменное количество аргументов:**
   params ref|out тип[] имяМассиваПараметров

Сначала указываются все обязательные параметры, потом все необязательные, потом параметр с переменным количеством аргументов.

Завершение работы метода

Метод завершает работу, когда выполнен последний оператор или после выполнения оператора return:

Досрочное завершение метода, не возвращающего значения, осуществляется оператором
return;

Метод, возвращающий значение всегда должен завершать работу оператором
return выражение;

Тип выражения должен совпадать с типом в заголовке метода или может быть приведен к нему неявно.

Перегрузки метода

Перегрузки метода — это несколько методов, имеющих одинаковое имя, но отличающихся друг от друга списком параметров (сигнатурой).

Различие в сигнатурах может быть как по количеству параметров, так и по типам параметров.

Документирование метода

Можно документировать метод прямо в коде программы (Visual Studio).

Для этого перед заголовком метода следует набрать строку комментария начинающегося с тройной косой черты и отредактировать появившийся шаблон подсказки:

/// …

Статические методы

Статический метод имеет модификатор static

Вызов статического метода
ИмяКласса.ИмяМетода (список аргументов)

Например:

```
int number = int.Parse("123456789");
double squareOfTriangle = 0,5 * a * b * Math.Sin(alpha);
Console.Writeline("Hello!");
```

Если статический метод вызывается внутри класса, где он определен, имя класса можно опустить.

Динамические методы

Динамический метод не имеет модификатора static

Вызов динамического метода
имяОбъекта.ИмяМетода(список аргументов)

Например:

```
var x = 0x91;
var y = x.ToString();
```

Рефакторинг

Рефакторинг — это изменение кода программы без изменения ее функциональности с целью

- улучшения читаемости кода,
- повышения ясности структуры кода и его эстетического восприятия,
- упрощения дальнейшей поддержки и изменений кода.

Один из основных принципов разработки — **code and refact**.

Рефакторинг

Правильное именование:

- начинайте имена локальных переменных с маленькой буквы, используя camelCase (для C#);
- начинайте имена методов, классов и констант с большой буквы, используя PascalCase (для C#);
- используйте содержательные имена;
- выбирайте имена на подходящем уровне абстракции;
- выбирайте недвусмысленные имена;
- избегайте кодирования в именах.

Рефакторинг

Методы

- метод должен выполнять **только одно действие**;
- название метода должно отражать **что он делает**,
  а не как он это делает;
- название метода желательно начинать с глагола в императиве;
- методы не должны иметь слишком много аргументов
  (идеально **0—3**);
- избегайте выходных аргументов (out);
- по количеству строк метод желательно должен быть
  не больше одного экрана;
- **избегайте флагов**;
- **удаляйте «мертвые» методы** и «мертвый код» в методах;
- избегайте повторяемости кода (принцип **DRY**);

Рефакторинг

Методы (продолжение)

- если метод **слишком громоздкий**, его целесообразно **разбить** на более простые методы;
- код метода должен быть написан **на одном уровне абстракции**.
  - методы лучше определять вблизи от места их использования;
  - локальные переменные лучше объявлять непосредственно перед первым использованием;
  - методы в классе следует располагать сверху вниз, вызываемые методы ниже вызывающих, чтобы весь код читался как журнальная статья;
  - не пишите несколько операторов в одной строке;
  - используйте пустые строки для визуального разделения кода метода на смысловые блоки.

Область видимости переменной

Это область кода программы, в пределах которой данная переменная доступна («видна»).

Место в коде программы, где объявлена переменная определяет ее область видимости. Грубо говоря, область видимости переменной — это контент, внутри которого она объявлена.

Тем самым область видимости может быть:

1. На уровне блока (внутри метода — локальная переменная)
2. На уровне метода (локальная переменная)
3. На уровне класса (поле)

Локальная переменная «перекрывает» статическое поле класса с тем же именем.