Весь код написанный нами, либо написанный уже кем-то до нас в виде библиотек, существуют в виде определенной структуры. Эту структуру можно представить в виде вложенных друг в друга "коробок".

1. Самая большая "коробка": Пространство имен (`namespace`)

В любой программе на C# вы увидите строку `namespace МояПерваяПрограмма;` . Пространство имен помогает избежать путаницы.

Представьте, что это адрес дома на улице. В мире может быть много людей с именем "Алексей", но "Алексей с улицы Цветочной, дом 5" это уже конкретный человек. Так же и здесь: два разных программиста могут создать класс с одинаковым именем `User` , но если они находятся в разных пространствах имен (`OnlineStore.User` и `Game.User`), то компилятор их не перепутает.

- **Зачем нужно?** Для организации кода и предотвращения конфликтов имен.

2. "Коробка" поменьше: Класс (`class`)

Внутри пространства имен всегда есть как минимум один класс. В простейшей консольной программе он часто называется `Program` .

```
class Program
{
    // Тут будет наш код
}
```

Класс - основной строительный блок в C#. На данном этапе думайте о нем просто как о **логическом контейнере для нашего кода**. Это чертеж, который описывает, что наша программа умеет делать. Все, что мы хотим выполнить, должно находиться внутри какого-нибудь класса.

3. "Точка входа": Метод `Main`

Внутри класса `Program` есть часть, с которой начинается работа программы. Это метод `Main` .

```
class Program
{
    static void Main(string[] args)
    {
        // Программа начинает выполняться отсюда!
    }
}
```

`Main` — это точка входа в программу. Когда вы запускаете приложение, система ищет именно этот метод, чтобы понять, с чего начать выполнение. Все, что вы напишете между фигурными скобками `{}` метода `Main`, будет исполняться по порядку, строчка за строчкой.

Главный вопрос: А почему мы пишем `Console.WriteLine()`?

Мы разобрались, что наш код лежит в классе `Program`, в методе `Main`. Теперь рассмотрим что такое `Console` и `Math`?

`Console` и `Math` это "готовые модели" (библиотеки), которые уже встроены в язык C#. Это **классы**, созданные разработчиками Microsoft, чтобы мы не реализовывали низкоуровневую логику вывода чего-то на экран, а использовали уже готовый инструмент.

- `Console` - класс, который отвечает за работу с консолью (тем самым черным окном, куда выводится текст).
- `Math` - класс, который содержит готовые математические функции.

Существуют другие библиотеки и классы, мы будем сталкиваться с ними по ходу работы.

Для чего нужна точка

Точка `.` в программировании — это **оператор доступа**. Она позволяет нам заглянуть внутрь класса и использовать то, что там лежит.

Когда мы пишем `Console.WriteLine("Привет, мир!")`, мы говорим компилятору:

1. "Найди мне класс с названием `Console`".
2. "Внутри этого класса найди инструмент (метод) с названием `WriteLine`".
3. "Используй этот инструмент и передай ему текст `Привет, мир!` для вывода на экран".

То же самое и с `Math.PI` или `Math.Sqrt(9)`:

1. "Найди класс `Math`".
2. "Внутри него возьми готовую функцию `Sqrt` (квадратный корень)".
3. "Выполни эту функцию для числа `9`".

Ключевое слово `static`

Вы, наверное, заметили слово `static` перед методом `Main`. По этому же принципу работают `Console` и `Math`.

На этом этапе представьте, что есть два типа инструментов:

1. **Инструмент для конкретного объекта**.
2. **Инструмент класса**.

Так вот, `Console` и `Math` — это как раз такие "общедоступные" классы. Все их методы (как `WriteLine` или `Sqrt`) помечены как `static` (статические). Это означает, что для их вызова **не нужно создавать отдельный экземпляр класса**. Они принадлежат самому классу, как бы "лежат в общей коробке", и доступны любому, кто знает имя этой коробки.

Итог для запоминания:

- **Класс** — это "коробка с инструментами" (методами) и данными.
- `Console` и `Math` — это готовые, встроенные в C# классы.
- **Точка** `.` позволяет "залезть" в класс и взять оттуда нужный инструмент.
- Мы пишем `Console.WriteLine`, потому что `WriteLine` — это статический (общедоступный) метод, который лежит внутри класса `Console`. Нам не нужно создавать "свою личную" консоль, мы пользуемся одной, общей на всю программу.