
KEEPING IT
CLASSY
WITH
C#

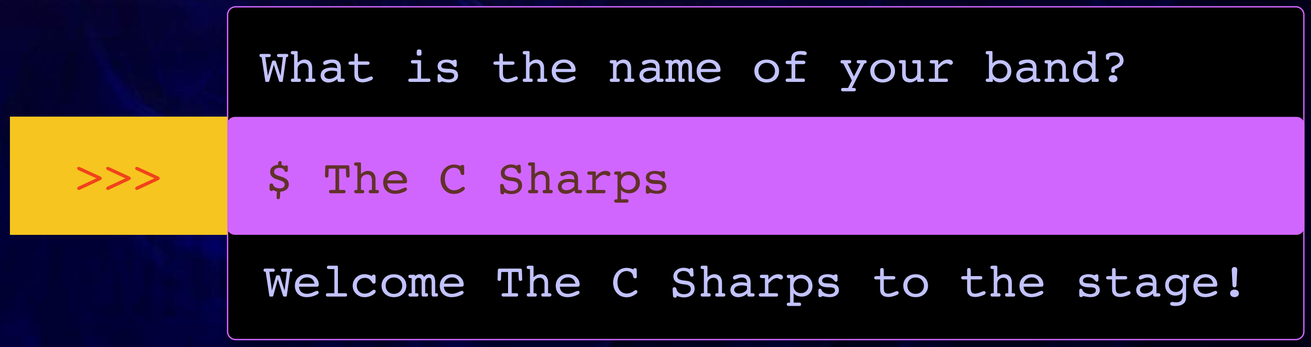

Level 1
Methods

What Are We Going to Make?

Using C# we're going to create an application for storing and retrieving band information.

What our application will do:

- Store information about a band and its musicians
- Announce the band
- Announce the musicians



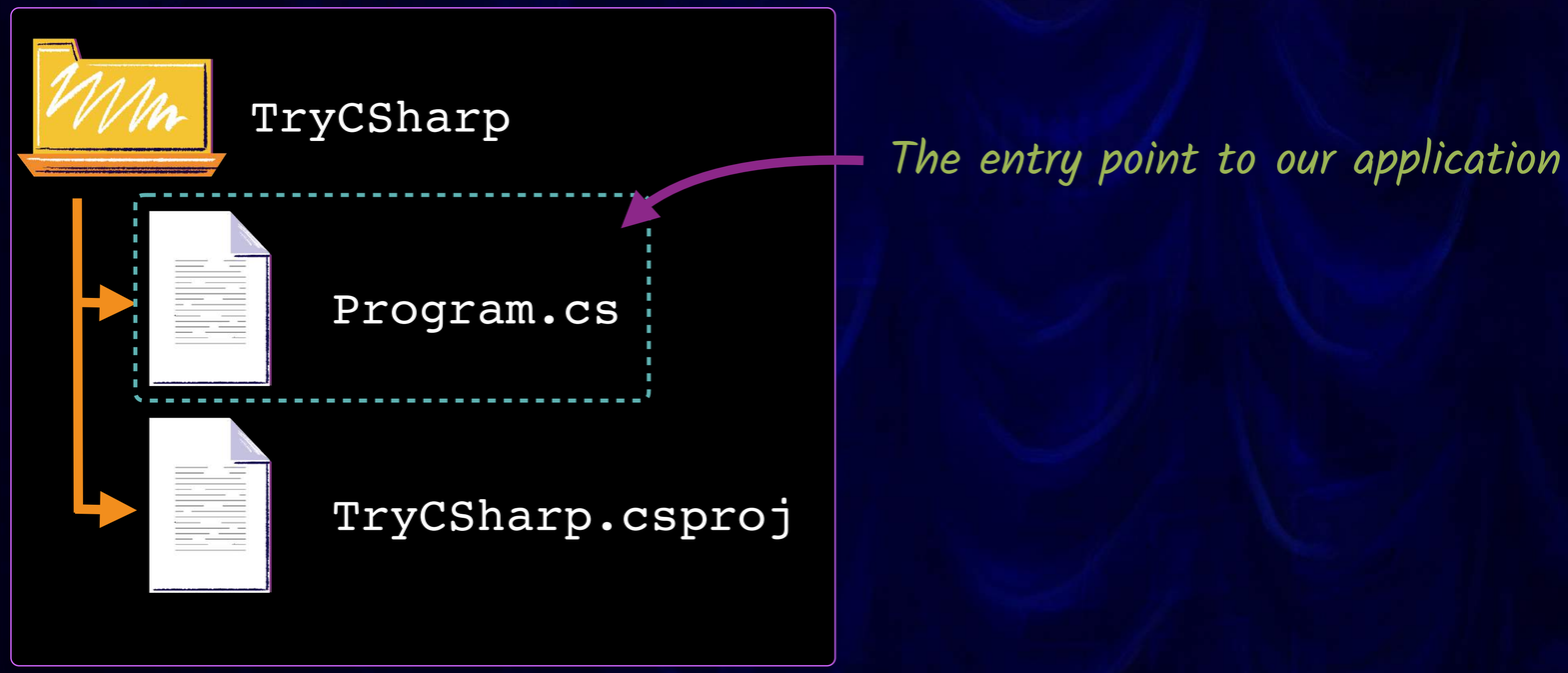
In this level:

- Collect the band's name
- Announce the band

KEEPING IT CLASSY WITH C#

Creating Our New Application

All C# **console applications contain a** csproj **file and a** Program.cs **file.**



KEEPING IT CLASSY WITH C#

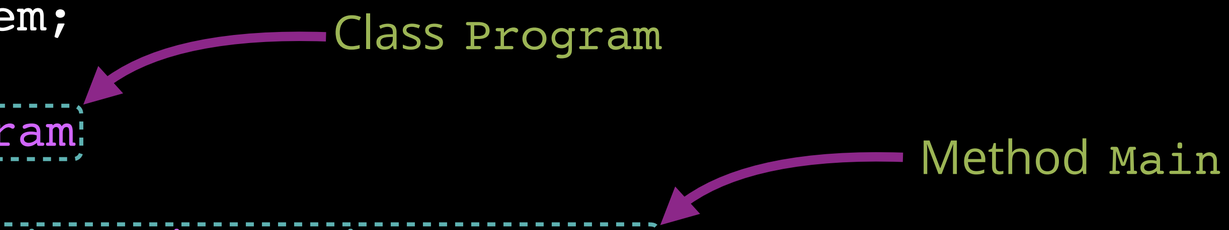
The Program.cs File

The generated Program.cs **file contains a** Program **class and** Main **method.**

Program.cs

```
using System;

class Program
{
  static void Main(string[] args)
  {
    Console.WriteLine("Hello World!");
  }
}
```



*Methods allow us to break our code into **reusable functional blocks**. We should create a separate method to announce our band... but how do we do that?*

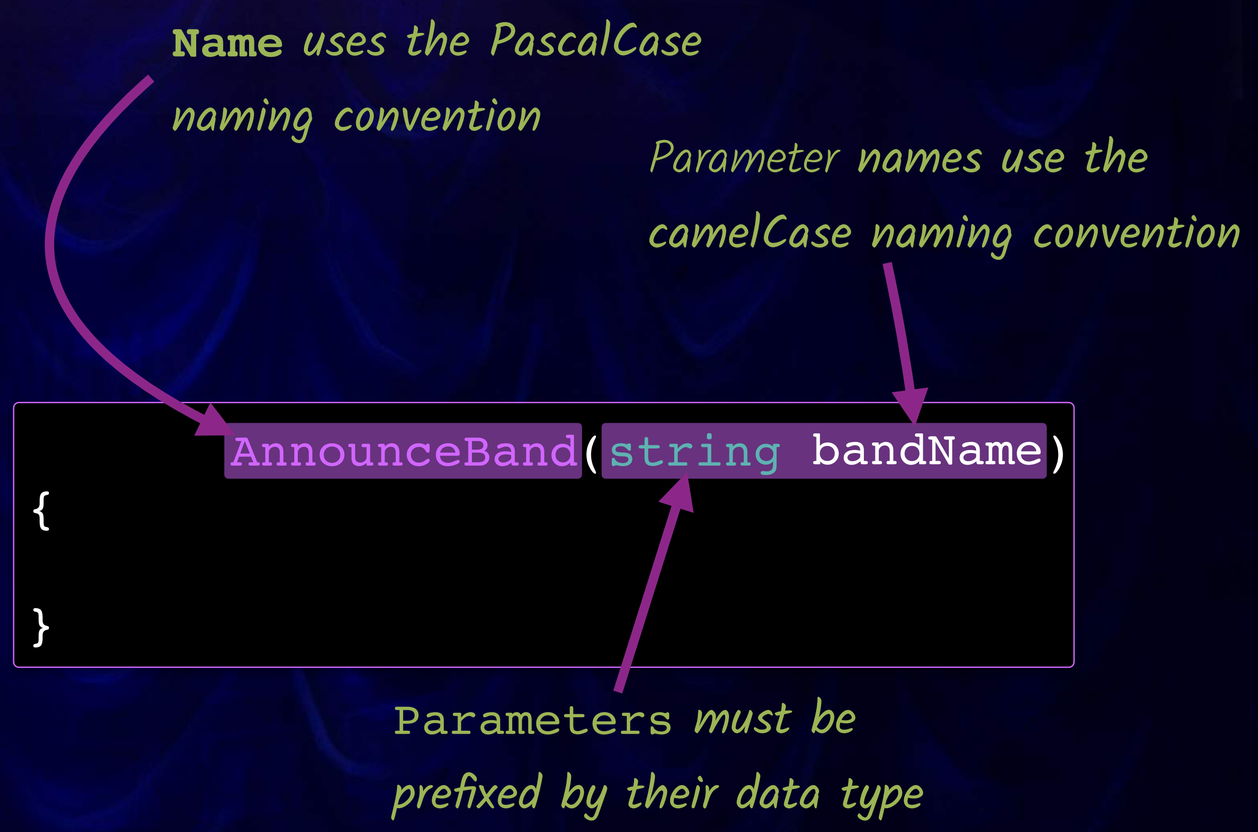
KEEPING IT CLASSY WITH C#

Method Name and Parameters

Methods **require a name and parameters.**

Name - The name we'll use to call the method throughout our program.

Parameters - values passed into the method.



This covers what our method takes in as parameters, but what about the type of data it returns?

KEEPING IT CLASSY WITH C#

Method Return Type and Statement

Methods **require a return type and return statement.**

Return type - The data type we expect to get back from the `method`.

```
string AnnounceBand(string bandName)
{
    return "Welcome " + bandName;
}
```

Return statement - Passes a value to the caller of the `method`.

But how do we call our `method`*?*

Calling a Method

To call a method**, we use its** Name **and replace the** parameters **with the values to be sent to the method.**

```
string AnnounceBand(string bandName)
{
  return "Welcome " + bandName;
}
```

coming from previous slide with magic move

`AnnounceBand` *prepends* *"Welcome "* *to the provided* `bandName`

Call the method using its `Name` *followed by the parameter values*

```
string announcement = AnnounceBand("The C Sharps");
Console.WriteLine(announcement);
```

animate this code box first

```
Welcome The C Sharps
```

KEEPING IT CLASSY WITH C#

Method Calls Can Be Used as Arguments

When a method is used as an argument, the data returned by the method will be used as the argument.

```
string AnnounceBand(string bandName)
{
  return "Welcome " + bandName;
}
```

We can use `AnnounceBand` *as an argument to* `Console.WriteLine`

```
Console.WriteLine(AnnounceBand("The C Sharps"));
```

→ `Welcome The C Sharps`

We should just move the `Console.WriteLine` *inside our* `AnnounceBand` *method.*

KEEPING IT CLASSY WITH C#

Method Error

AnnounceBand **is throwing an error because its return type is a** string**, but isn't returning anything.**

*We are writing to the console, but NOT returning a **string** from the method*

```
string AnnounceBand(string bandName)
{
  Console.WriteLine("Welcome " + bandName);
}
```

ERROR: Not all code paths return a value

How do we write a method that will not return anything?

KEEPING IT CLASSY WITH C#

Return Type void When Nothing Returned

Methods that don't return anything should use the void return type.

Indicates no value will be returned by the method...

```
void AnnounceBand(string bandName)
{
  Console.WriteLine("Welcome " + bandName);
}
```

```
Welcome The C Sharps
```

...and return statement is not required

KEEPING IT CLASSY WITH C#

Expand Welcome Message

Announcing our band should be more inviting.

Change our Announce to "Welcome _____ to the stage!"

```
void AnnounceBand(string bandName)
{
  Console.WriteLine("Welcome " + bandName + " to the stage!");
}
```

```
Welcome The C Sharps to the stage!
```

KEEPING IT CLASSY WITH C#

Set Up Main Method to Use AnnounceBand

The Main method is now calling the AnnounceBand method and passing it the band name.

Program.cs

```
class Program
{
  static void Main(string[] args)
  {
    Console.WriteLine("What is the name of your band?");
    string name = Console.ReadLine();
    AnnounceBand(name);
  }

  void AnnounceBand(string bandName){…}
}
```

Let us review how the program will actually run this code.

KEEPING IT CLASSY WITH C#

Method Execution Step-By-Step

Execution starts on the Main **method then transfers to** AnnounceBand.

Program.cs

```
static void Main(string[] args)
{
1 Console.WriteLine("What is the name of your band?");
  string name = Console.ReadLine();
  AnnounceBand(name);
}

void AnnounceBand(string bandName)
{
  Console.WriteLine("Welcome " + bandName + " to the stage!");
}
```

KEEPING IT CLASSY WITH C#

Method Execution Step-By-Step

Execution starts on the Main **method then transfers to** AnnounceBand.

Program.cs

```
static void Main(string[] args)
{
1 Console.WriteLine("What is the name of your band?");
2 string name = Console.ReadLine();
  AnnounceBand(name);
}

void AnnounceBand(string bandName)
{
  Console.WriteLine("Welcome " + bandName + " to the stage!");
}
```

KEEPING IT CLASSY WITH C#

Method Execution Step-By-Step

Execution starts on the Main **method then transfers to** AnnounceBand.

Program.cs

```
static void Main(string[] args)
{
1 Console.WriteLine("What is the name of your band?");
2 string name = Console.ReadLine();
> AnnounceBand(name);
}
                    Main will transfer execution to AnnounceBand
void AnnounceBand(string bandName)
{
3 Console.WriteLine("Welcome " + bandName + " to the stage!");
}
```

Methods Run Synchronously

Once AnnounceBand **completes, it transfers execution back to the** Main **method.**

Program.cs

```
static void Main(string[] args)
{
1 Console.WriteLine("What is the name of your band?");
2 string name = Console.ReadLine();
> AnnounceBand(name);
                          4 Main is finished executing
}

void AnnounceBand(string bandName)
{
3 Console.WriteLine("Welcome " + bandName + " to the stage!");
}
```

AnnounceBand *will transfer execution back to* Main

Our Working Application

Our application now performs the functions we set out to accomplish in this level.

Current Features:

- Collects the band's name
- Announces the band

```
What is the name of your band?
>>>  $ The C Sharps
Welcome The C Sharps to the stage!
```

KEEPING IT CLASSY WITH C#

A Quick Recap on Methods

We divide our executable code into logical pieces using methods.

- Methods contain our classes' executable code
- Method names are case sensitive
- To call a method, use its Name followed by values we intend to pass into the method in parenthesis