Making Classes

C# provides built-in data types, like `string`: each instance of the `string` type has its own values and functionality.

```
string phrase = "zoinks!";
Console.WriteLine(phrase.Length);
Console.WriteLine(phrase.IndexOf("k"));
```

In this case `phrase` is an *instance* of the `string` type. Every `string` has a `Length` property and `IndexOf()` method, but the resulting values are different for each instance.

A *class* represents a custom data type. In C#, the class defines the kinds of information and methods included in a custom type.

You can then make *instances* of that class (above, `phrase` was an instance of `string`). There may be many instances of the same class, all with unique values.

To begin defining a class, C# uses this structure:

```
class Forest {
}
```

The code for a class is usually put into a file of its own, named with the name of the class. In this case it's **Forest.cs**. This keeps our code organized and easy to debug.

In other parts of code, like `Main()` in **Program.cs**, we can use the class. We make instances, or *objects*, of the `Forest` class with the `new` keyword:

```
Forest f = new Forest();
```

We could say `f` is an instance of the `Forest` class, or `f` is of type `Forest`.

The process of creating an instance is called *instantiation*. Today we *instantiate* a class; yesterday they *instantiated* a class, and so on.

☑Instructions

1.

We will define our class in **Forest.cs** and work with that class in the `Main()` method in **Program.cs**.

Within the namespace `BasicClasses`, build an empty `Forest` class in **Forest.cs**.

Hint

Follow the example shown in the narrative. Make sure you are defining your class in **Forest.cs**.

2.

In `Main()` in **Program.cs** make a new instance of the `Forest` class and store the result in a variable `f`.

Hint

To create a new instance of the `Recipe` class:

```
Recipe r = new Recipe();
```

Make sure to do this in `Main()`!