Constructors

In each of the examples so far, we created a new `Forest` object and set the property values one by one. It would be nice if we could write a method that's run every time an object is created to set those values at once.

C# has a special type of method, called a *constructor*, that does just that. It looks like a method, but there is no return type listed and the method name is the name of its enclosing class:

```
class Forest
{
  public Forest()
  {
  }
}
```

We can add code in the constructor to set values to fields:

```
class Forest
{
  public int Area;

  public Forest(int area)
  {
    Area = area;
  }
}
```

This constructor method is used whenever we instantiate an object with the `new` keyword:

```
 // Constructor is called here
Forest f = new Forest(400);
```

But we've been instantiating new objects all day! Why did it work before we defined a constructor?

If no constructor is defined in a class, one is automatically created for us. It takes no parameters, so it's called a *parameterless constructor*. That's why we have been able to instantiate new objects without errors:

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

☑Instructions

1.

Define a constructor for the `Forest` class. It should have two parameters:

- `name`, which sets the `Name` property
- `biome`, which sets the `Biome` property

It should also set the value of `Age` to 0.

Hint

This is the first line of the method definition:

```
public Forest(string name, string biome)
```

2.

The code in **Program.cs** has been commented out. Un-comment it and run it.

You should see an error:

```
error CS7036: There is no argument given that corresponds to the required formal
parameter 'name' of 'Forest.Forest(string, string)'
```

This error occurs because you are using the parameterless constructor `Forest()` in **Program.cs**. This no longer works because a constructor `Forest(string, string)` has been defined.

3.

Call the new constructor in `Main()` to create a `Forest` object with the name “Congo” and biome “Tropical”.

Delete the lines:

```
f.Name = "Congo";
```

and

```
f.Biome = "Desert";
```

They’re no longer useful because those properties are now set in the constructor!

Hint

Here’s an example of the `Recipe` constructor with two string arguments:

```
Recipe r = new Recipe("Ratatouille", "A+");
```