Overloading Constructors

Just like other methods, constructors can be overloaded. For example, we may want to define an additional constructor that takes one argument:

```
public Forest(int area, string country)
{
  this.Area = area;
  this.Country = country;
 }

public Forest(int area)
{
  this.Area = area;
  this.Country = "Unknown";
}
```

The first constructor provides values for both fields, and the second gives a default value when the country is not provided. Now you can create a `Forest` instance in two ways:

```
Forest f = new Forest(800, "Africa");
Forest f2 = new Forest(400);
```

Notice how we've written duplicate code for our second constructor: `this.Area = area;`. Later on, if we need to adjust the constructor, we'll need to find every copy of the code and make the exact same change. That means more work and chances for errors.

We have two options to resolve this. In either case we will remove the duplicated code:

> Use default arguments. This is useful if you are using C# 4.0 or later (which is fairly common) and the only difference between constructors is default values.

```
public Forest(int area, string country = "Unknown")
{
  this.Area = area;
  this.Country = country;
}
```

2. Use `: this()`, which refers to another constructor in the same class. This is useful for old C# programs (before 4.0) and when your second constructor has additional functionality. This example has an additional functionality of announcing the default value.

```
public Forest(int area, string country)
{
  this.Area = area;
  this.Country = country;
}

public Forest(int area) : this(country, "Unknown")
{
  Console.WriteLine("Country property not specified. Value defaulted to
'Unknown'.");
}
```

Remember that `this.Area` refers to the current instance of a class. When we use `this()` like a method, it refers to another constructor in the current class. In this example, the second constructor calls `this()` — which refers to the first `Forest()` constructor — AND it prints information to the console.

☑Instructions

1.
Define a second constructor for the `Forest` class:

It should take one parameter, `name`.

It should use `: this()` with the `name` variable as the first argument and `"Unknown"` as the second.

It should also print a warning to the console about the defaulted value.

Hint

Here's the example used in the narrative:

```
public Forest(int area) : this(area, "Unknown")
{
  Console.WriteLine("Country property not specified. Value defaulted to
'Unknown'.");
}
```

2.
In `Main()`, call the second constructor to create a `Forest` object named “Rendlesham”.
3.
Below the constructor call, print the `Biome` property of this new instance.

When you run the code, you should see the warning message and “Unknown” printed to the console. Why are these two things printed?