Fields

We need to associate different pieces of data, like a size and name, to each Forest object. In C#, these pieces of data are called *fields*. Fields are one type of class *member*, which is the general term for the building blocks of a class.

Create fields like this:

```
class Forest {
  public string name;
  public int trees;
}
```

This might look similar to defining a variable. It is! Each field is a variable and it will have a different value for each object.

With the code above, we haven't set the value of either field, so each has a default value. In this case string s default to null, int s to 0, and bool s to false. You can find the default values for more types in Microsoft's default values table.

It is common practice to names field with lowercase (name instead of Name). This makes fields easy to recognize later on.

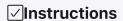
Don't worry about public yet: it's explained later in this lesson.

Once we create a Forest instance, we can access and edit each field with dot notation:

```
Forest f = new Forest();
f.name = "Amazon";
Console.WriteLine(f.name); // Prints "Amazon"

Forest f2 = new Forest();
f2.name = "Congo";
Console.WriteLine(f2.name); // Prints "Congo"
```

Each instance has a name field, but the value may differ across instances.



1.

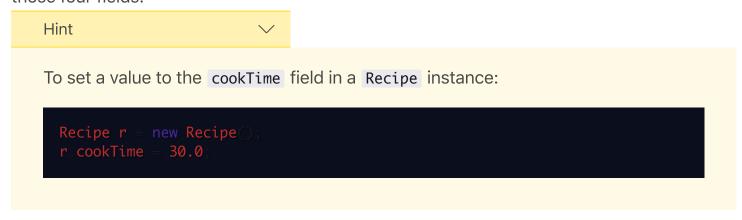
In Forest.cs, add name, trees, age, and biome fields to the Forest class.

Hint ~

As another example, this code declares a cookTime field in the Recipe class:

class Recipe
{
 public double cookTime;
}

2. In **Program.cs** in Main(), a Forest object has already been instantiated. Set values to those four fields.



3. In Main(), print the name field to the console.