

Review

Congrats! You are now ready to use `static` throughout your classes:

In general, *static* means “associated with the class, not an instance”.

A static member is always accessed by the class name, rather than the instance name, like `Forest.Area`.

A static method cannot access non-static members.

A static constructor is run once per type, not per instance. It is invoked before the type is instantiated or a static member is accessed.

Either of these would trigger the static constructor of `Forest`:

```
public static void Main() {  
    Forest f = new Forest();  
}
```

or

```
public static void Main() {  
    Forest.Define();  
}
```

A static class cannot be instantiated. Its members are accessed by the class name, like `Math.PI`.

☒ Instructions

1.

From **Program.cs**, print the number of forests created.

Hint



Use the static `ForestsCreated` property.

2.

Instantiate two `Forest` objects.

Hint



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

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

3.

Print the number of forests created again. Before moving on, make sure you can explain how this value was changed.