Static Constructors

An instance constructor is run before an instance is used, and a *static constructor* is run once before a class is used:

```
class Forest
{
  static Forest()
  { /* ... */ }
}
```

This constructor is run when either one of these events occurs:

> Before an object is made from the type.
>
> Before a static member is accessed.

In other words, if this was the first line in `Main()`, a static constructor for `Forest` would be run:

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

It would also be run if this was the first line in `Main()`:

```
Forest.Define();
```

Typically we use static constructors to set values to static fields and properties.

A static constructor does not accept an access modifier.

☑Instructions

1.

In the previous exercises our `treeFacts` and `forestsCreated` fields were never given values! We'll fix that.

First, create a static constructor for `Forest`.

Hint

This example `Recipe` class has a single static constructor:

```
class Recipe
{
  static Recipe()
  { }
}
```

2.
In the body of the static constructor, set the `treeFacts` field to this string:

```
"Forests provide a diversity of ecosystem services including:\r\n  aiding in
regulating climate.\r\n  purifying water.\r\n  mitigating natural hazards such as
floods.\n"
```

Hint

Make sure to set the field `treeFacts`, NOT the property `TreeFacts`. The property has no set method.

3.
In the body of the static constructor, set the `ForestsCreated` property to 0.
4.
In **Program.cs**, call `Forest.PrintTreeFacts()` to check that the `TreeFacts` property was set.