Static Methods

You already know how to create an instance method, like:

```
class Forest
{
  private string definition;
  public void Define()
  {
    Console.WriteLine(definition);
  }
}
```

This behavior (printing a general definition) isn't specific to any one instance — it applies to the class itself, so it should be made static.

To make a static method, just add `static` after the access modifier (`public` or `private`).

```
class Forest
{
  private static string definition;
  public static void Define()
  {
    Console.WriteLine(definition);
  }
}
```

Notice that we made added `static` to both the field `definition` and method `Define()`.

This is because a static method can only access other static members. It cannot access instance members:

```
class Forest
{
  private string definition;
  public static void Define()
  {
    // Throws error because definition is not static
    Console.WriteLine(definition);
    }
  }
```

Otherwise, static methods work like any other method.

☑**Instructions**

1.
Earlier we mentioned storing an explanation of forests in general. We'll use a field and property to define the explanation. Define a private static string field named `treeFacts`.

2.
Define a public static property named `TreeFacts` with just a getter (no setter).

3.
Define a public static method name `PrintTreeFacts()` that writes the value of `TreeFacts` to the console.

Note that `TreeFacts` is never assigned a value: we'll resolve that in the next exercise.

Hint

Here's the first line of the method:

```
public static void PrintTreeFacts()
```