Introduction to Static

At this point, you may recall:

> A custom data type is defined by a *class*.
>
> An *instance* of a class is called an *object*. Multiple, unique objects can be instantiated from one class.
>
> This process of bundling related data and methods into a type is called *encapsulation*, and it makes code easier to organize and reuse.

What if we needed to do something related to the type itself, not instances of that type? For example, where do we store the count of all `Forest` objects, or an explanation of forests in general?

To keep with the rules of encapsulation, these shouldn't be associated with one instance (because this information is related to the `Forest` class, not a single `Forest` instance).

These facts and behaviors should be associated with the class itself! We call these types of members *static*.

This lesson will cover the meaning of *static*, how to apply it to different types of class members, and its typical uses cases.

☑Instructions

The `Forest` class is defined in **Forest.cs**. Make sure you're familiar with it before moving on! We'll be adding to this class throughout the lesson.

Testing this kind of code takes longer, so some checkpoints in this lesson will need more time to run. It's worth the wait!