Classes and Object-Oriented Programming

If you are taking this course, you have probably used object-oriented programming (OOP) previously in another language. If it's been a while since you've used OOP, OOP is a style of coding that collects related data (object *attributes*) and functions (object *methods*) together to form a single data structure, called an *object*. This allows that collection of attributes and methods to be used repeatedly in your program without code repetition.

In C++ the attributes and methods that make up an object are specified in a code *class*, and each object in the program is an *instance* of that class.

This concept is intended to provide you with the basic syntax for writing classes in C++. In this Foundations course, you will not need to write your own classes for the project, but you will be modifying existing classes in the code. You will be writing your own classes in the next course of this Nanodegree: Object-Oriented Programming.

↑ Menu Expand

Inheritance

It is possible for a class to use methods and attributes from another class using class *inheritance*. For example, if you wanted to make a `Sedan` class with additional attributes or methods not found in the generic `Car` class, you could create a `Sedan` class that inherited from the `Car` by using the colon notation:

```
class Sedan : public Car {
    // Sedan class declarations/definitions here.
};
```

By doing this, each `Sedan` class instance will have access to any of the *public* methods and attributes of `Car`. In the code above, these are `IncrementDistance()` and `PrintCarData()`. You can add additional features to the `Sedan` class as well. In the example above, `Car` is often referred to as the *parent* class, and `Sedan` as the *child* or *derived* class.

A full discussion of inheritance is beyond the scope of this course, but you will encounter it briefly in the project code later. In the project code, the classes are set up to inherit from existing classes of an open source code project. You won't need to use inheritance otherwise, but keep in mind that your classes can use all of the public methods and attributes of their parent class.