Static and Non-Static Methods

In this lesson, we will discuss static and non-static methods.

Static and Non-Static Methods

Methods can be of static or non-static, also known as instance methods, type. Let's have a brief discussion about these.

Static Methods

Static methods, just like static variables, can be called from another class without instantiating/creating an object of the class. These methods can be called using the class' name. Let's check this out in our code:

```
class VendingMachine {

  static string manufacturer = "Vendy Inc.";

  public static void PrintManufacturer() { //static method
    Console.WriteLine("The manufacturer of machine is {0}", manufacturer);
  }

}

class Demo {

  public static void Main(string[] args) { //Note that the Main() is also a static method
      //calling the static method using the class name
      VendingMachine.PrintManufacturer();

  }

}
```

> Static methods cannot access non-static variables directly, i.e., without creating an instance of a class.

Normally static methods are declared when we don't need multiple instances of a particular feature. For example, there can be only one `Main()` method in a C# program because it serves as an entry point to that program. When the compiler executes a program it looks for a static method named `Main()` because it knows it has to run this method directly without the need to create an instance of a class. If needed, we can actually have multiple `Main()` methods in a program but then we'll have to tell the compiler which `Main()` method should be used as the entry point.

Non-Static Methods

Non-static methods of a class can only be called with reference to a specific instance of that class. Since the non-static methods are referenced using a specific object, we can say that they can be used to perform an object-specific operation. Non-static methods are also referred to as instance methods.

We have been working with instance methods until now. Let's take a deeper look into this with the help of a code.

```csharp
class VendingMachine {

  static string manufacturer = "Vendy Inc.";
  private int _capacity = 100;

  public void PrintSpecification() { //instance method
    Console.WriteLine("The {0}'s machine has {1}  products capacity", manufacturer, _capacity
  }

}

class Demo {

  public static void Main(string[] args) {
      VendingMachine vendingMachine = new VendingMachine();
      //calling the instance method using the class object
      vendingMachine.PrintSpecification();

  }

}
```

The non-static methods can access both static and non-static fields.

In the next lesson, we will learn about the `this` reference variable.