# Calling a Constructor from Another Constructor

In this lesson, you will learn how to call a constructor from another constructor.

We have previously learned about this reference variable. This is the second use case of the `this` reference variable.

We already know that a class can have multiple constructors. There can be a parameterized constructor which initializes all the fields or there can be another constructor which initializes some of the fields and so on. In a case where the code to initialize some of the fields is already written in another constructor, why not just call that other constructor.

In C#, we can call a constructor from another constructor. When you call a constructor from another constructor, you use the `this` keyword followed by `()` to refer to the constructor.

Let's see it in action:

```
class VendingMachine {

  private bool _onOff;
  private int _count;
  private int _capacity = 100;
  private int _moneyCollected;
  // A parameter-less constructor implemented
  public VendingMachine() {
    // Use of this keyword on the left side of = operator
    this._onOff = false;
    this._count = 0;
    this._moneyCollected = 0;

  }
  // A parameterized constructor implemented
  public VendingMachine(bool onOff , int count) {
    // Use of this keyword on the left side of = operator
    this._onOff = onOff;
    this._count = count;

  }

  public VendingMachine(bool onOff , int count, int moneyCollected)
    : this(onOff,count) // Calling the above parameterized constructor
  {
```

```
    this._moneyCollected = moneyCollected;

  }

  // A simple print function
  public void PrintFields(){

    Console.WriteLine("Is the machine turned on? {0}", this._onOff);
    Console.WriteLine("The count of products is: {0}", this._count);
    Console.WriteLine("The capacity of machine is: {0}", this._capacity);
    Console.WriteLine("The total money collected till now is: {0}\n", this._moneyCollected);
  }

}

class Demo {

  public static void Main(string[] args) {
    // Object created with parameterized constructor!
    var vendingMachine1 = new VendingMachine(true,50,10);
    // Object created with overloaded constructor!
    var vendingMachine2 = new VendingMachine(true,5);
    // Object created with parameter-less constructor!
    var vendingMachine3 = new VendingMachine();
    vendingMachine1.PrintFields();
    vendingMachine2.PrintFields();
    vendingMachine3.PrintFields();

  }

}
```

The `this` keyword followed by parentheses means that another constructor in the same C# class is being called. At line **24**, the constructor with `onOff` and `count` parameters is being called. The compiler automatically calls the constructor that has a matching argument list.

---

This concludes our discussion on the basic classes in C#. The next section deals with the concept of data hiding, which plays a pivotal role in implementing efficient classes. Before moving on, let's take a quick quiz to test our understanding.