# Constructors

**The goal for this exercise** is to understand how to call one constructor from another, thus avoiding duplicated code.

We’re going to look at how to create several, overloaded constructors (on a base class, or a derived class – it doesn’t matter), without needlessly duplicating code. The way you have one constructor call another within the same class is similar to calling a constructor on the base class, but uses the work **this** instead:

public class Base

{

 int x;

 public Base(int newX)

 {

 x = newX;

 }

 public Base() **: this(0)** // jump to the other constructor

 {

 }

}

**What you need to do for this exercise:**

Within the provided starter project, there is already a class named Inheritance\_Constructors. You should create a class named Saw near it. Create 1 subclass of Saw – an ElectricSaw. Each Saw object has a certain sharpness (rated on a scale of 1 through 10, whole numbers only), and the ElectricSaw also has a power cord of a certain length (measured in feet). As always, the instances variables for these classes should be declared private (since we don’t have a compelling reason to make them anything else)

Within the Saw class, create two constructors: a default constructor, and a constructor that takes a sharpness argument. Have the default constructor call over to the non-default constructor, and pass in the value of 5 (i.e., if we’re not told otherwise, we will assume that the Saw object is of middling sharpness).

Within the ElectricSaw class, create three constructors: a default constructor, a constructor which takes a value for the sharpness of the saw, and a constructor which takes a value for the sharpness of the saw along with and a value for the length of the cord. Have the ElectricSaw‘s default constructor, and the ElectricSaw‘s single-parameter constructor, call the ElectricSaw constructor that takes two arguments. Make sure that the two argument ElectricSaw constructor correctly calls the appropriate constructor in the base class.

If you need to use any particular values, you should use a sharpness of 6 (we’ll assume that ElectricSaws are slightly sharper than non-electric saws), and a cord length of 20 feet.

 Additionally, you should define a method named Print() on both classes, which you’ll use to verify that everything is working ok. Create enough instances, of each class, so that you’ve tested each and every constructor at least once – put this code in Inheritance\_Constructors.RunExercise().