# Objects As Parameters

**The goal for this exercise** is to define the Television object, which you will be using in several, subsequent exercises.

For all the methods that you implement (in this course (not just this exercise, but in this course, as you go forwards)), you should remember that since method is public, anyone, anywhere can call the method. Even people whom you never thought would call this method. Therefore, you need to make sure that all the parameters to this method are ‘safe’ before using them. In particular, you need to check that the **any** object references are valid, and not equal to the special **null** value. If any object reference parameters are null, you can simply return early.

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

1. Implement the Television class. Each Television has a brand name (a string), a price (use decimal), and a size (measured diagonally in inches). Give the class a constructor, and getter/setter methods for each of the three data attributes.
   1. Note that price is actually a decimal, not a double, value.
   2. When a Television object is created using the default (‘no-argument’) constructor, the price and size should be zero, and the brand should be the empty string (“”), NOT the value null.
   3. Remember to validate the input to the setter methods. If you’re given a negative value for price or size, reject the new value and keep the current value, instead. Similarly, if the SetBrand method is given a null value, it should reject this null value and keep the current value, instead.
   4. You also need to implement a Print method, which should print out a message in the following format:  
      Brand:Sony  
      Price:1000.17  
      Size:10.5
   5. The class definition is already partially provided for you in the starter solution.
2. In preparation for future exercises, you should make sure that you can create a Television object, give it some values, and then pass that object to the method. A great place to put this code is the RunExercise method of the Television\_Definition class.
3. Other than implementing this class so that it works with future exercises, there are no particular requirements for this exercise. The only reason this might be listed as "Hand-In" is to make it clear that you need to do this work.