Java Object Class and the toString( ) Method

The Java **Object** class resides at the top of the class hierarchy tree in the Java development environment. This means every class in the Java system is a descendent of the **Object** class. (This is an illustration of inheritance.) The **Object** class defines the basic state and behavior for all objects, such as

1. Being able to compare oneself (the current object) to another object – an **equals( )** method
2. Returning the object's class – the **getClass( )** method
3. Returning a string representation of the object - the **toString( )** method

So, there is a default **toString( )** method defined in the **Object** class and thus is available to be used by every class you create. In other words, the classes you create inherit characteristics and methods from the **Object** class, including the **toString( )** method.

The to**String( )** method of an object, including objects instantiated from classes you create, is automatically called when you

1. Pass an object to a **print( )** or **println( )** method
2. Concatenate an object to a string

Unfortunately, the default **toString( )** method from the Object class is VERY generic and often inadequate for the classes you create. Therefore, it recommended that all classes you create include a **toString( )** method to override the **Object toString( )** method. This allows you to control how your object will be converted to a string for output (and concatenation).

The **toString( )** method is VERY handy for debugging your new classes as it will display the data currently held in your instance variables (if you create the **toString( )** methods to display all instance data).

Let's see an example of the **Object toString( )** method and a class **toString( )** method.

1. Open the **CarDriver.java** file and the **Customer.java** files that were previously loaded in Blackboard with the **CarExample.zip** file. These were discussed in the Chapter 5 videos.

2. Look at the **toString( )** method in the **Customer.java** file. This method returns a string in a pleasing output format. The string includes all instance data (name, parts, and labor).

3. Look at the **CarDriver.java** file. Scroll to the end of the **main( )** method. You will see a section on using the **toString( )** methods. Two **println( )** methods are printing **Henry.toString( )** and **Julie.toString( ).** In other words, a string has been passed from the **toString( )** method in the CarDriver class to the **println( )** method in the driver program to print something for the Henry and Julie objects. The appearance of this string is controlled by the **toString( )** method in the **Customer** class.

4. Compile the two programs and run **CarDriver**. You will see the formatted output. The last lines of output are from the toString( ) method.

5. Change the following lines of code in CarDriver

```
System.out.println(Henry.toString());
System.out.println(Julie.toString());
```

   to

```
System.out.println(Henry);
System.out.println(Julie);
```

6. Compile and run **CarDriver** again.

   You get the same results because the **toString( )** method is automatically called when you pass an object to the **print( )** or **println( )** methods (such as Henry and Julie in the last 2 statements above).

7. Now, go to the **Customer** class and comment out the entire **toString( )** method. Compile the **Customer** class. We no longer have an active **toString( )** method in the Customer class.

   BUT -- You will still have access to a **toString( )** method in the **Object** class. The **Customer** class inherits this method and can be used.

8. Run the **CarDriver** program and look at the output from the **toString( )** method from the **Object** class. It's not pretty. In fact, it's not very readable.

This illustrates why it is important to create a **toString( )** method for the classes you create. The **toString( )** method from the **Customer** class is better as it overrode the **toString( )** method from the **Object** class using a much more pleasing output format.