LESSON 36 NOTES

Specifics on Interfaces

- ***No body of code within methods***
- ***Methods simply contain the method header, followed by a semicolon***
- ***All methods have public visibility***
- ***Interfaces CANNOT BE INSTANTIATED!!***
- ***A class that implements one or more interfaces MUST provide a definition (method body) of each method in each interface that is implemented***

Format of an Interface

```
public interface IClassName
{
    //constants
    //method headers (followed by semicolon)
}
```

Implementing an Interface (the class header)

```
public class ClassName implements
                IClassName1, IClassName2, ...
```

From Sun's Java Tutorials:

There are a number of situations in software engineering when it is important for disparate groups of programmers to agree to a "contract" that spells out how their software interacts. Each group should be able to write their code without any knowledge of how the other group's code is written. Generally speaking, *interfaces* are such contracts.

For example, imagine a futuristic society where computer-controlled robotic cars transport passengers through city streets without a human operator. Automobile manufacturers write software (Java, of course) that operates the automobile—stop, start, accelerate, turn left, and so forth. Another industrial group, electronic guidance instrument manufacturers, make computer systems that receive GPS (Global Positioning System) position data and wireless transmission of traffic conditions and use that information to drive the car.

The auto manufacturers must publish an industry-standard interface that spells out in detail what methods can be invoked to make the car move (any car, from any manufacturer). The guidance manufacturers can then write software that invokes the methods described in the interface to command the car. Neither industrial group needs to know *how* the other group's software is implemented. In fact, each group considers its software highly proprietary and reserves the right to modify it at any time, as long as it continues to adhere to the published interface.

The Comparable interface: ***Can be used to compare two objects***

- ***Only contains one method:*** `compareTo()`

- ***EXAMPLE OF USAGE:***
  - `if (obj1.compareTo(obj2) < 0)`
    - `System.out.println ("obj1 is less than obj2");`

- ***The value returned from*** `compareTo` ***should be:***
  - ***negative if*** `obj1` **<** `obj2`**,**
  - ***0 if*** `obj1` **=** `obj2`
  - ***positive if*** `obj1` **>** `obj2`

  - ***IT IS UP TO THE PROGRAMMER TO DETERMINE WHAT MAKES ONE OBJECT LESS THAN ANOTHER***