**Interfaces**

**What is an Interface?**

* An **Interface** is a class-like construct that contains only constants, and abstract methods
* In many ways, an interface is like an abstract, but the intent of this structure is to specify a common behaviour for objects that implement the interface
* Interfaces are a way of allowing an object to have multiple abstract parents
  + A Class is not limited in the number of interfaces it can implement
* Collection classes and their associated helper classes do rely in some cases on an interface being specified in order for them to work correctly

**Interface Syntax**

*public interface InterfaceName*

*{*

*constant declarations;*

*abstract method signatures;*

*}*

*public interface whimisTrained*

*{*

*public Boolean isTrained();*

*}*

**Interfaces are a Special Class**

* An interface is compiled into separate bytecode files just like a regular Java class
* You **cannot** create an instance of an interface using the **new** operator
  + However, similar to abstract classes, you can have interface references to the classes that implement the interface
    - Example: you can have an array of interface references

**The Comparable Interface**

* This is a system defined interface that requires any class that implements this interface to have a **compareto** method associated with it
  + Numbers and strings automatically have this interface implemented
* The **compareto** method returns an integer based upon the difference between 1 object comparing itself against another object
  + *objOne.compareTo(obTwo)*
  + If objOne is greater than obTwo
    - A positive value will be returned
  + If objOne is less than obTwo
    - A negative value will be returned
  + If objOne is equal to obTwo
    - 0 will be returned

**Comparable**

* The **comparable** interface can be used as an object reference, or a generic
* If a class implements Comparable
  + Then an object reference will be passed to it
* If a class implements Comparable<T>
  + Then a reference of type **T** will be passed to it
* **These will be on the Final**

**The Collections Class**

* A built-in class with a number of static methods associated with it
* This can be used to sort a particular collection in the way that you specify
* Can also be used to search for certain elements in a list
* Provides the same functionality as the **Arrays** class except that it works with **Collections**

**Comparator Interface**

* The second most common interface that is used is the **comparator** interface
* Any class that implements this interface needs the following methods
  + **compare** method
  + **equals** method
* **Compare** takes two object references and returns an integer based on the relationship between them

|  |  |
| --- | --- |
| Comparison | Return Value |
| Object 1 > Object 2 | Positive Value |
| Object 1 < Object 2 | Negative Value |
| Object 1 == Object 2 | 0 |