Interface Wisdom

Interface types are groups of related methods with empty bodies.

```
public interface Stampedable{
     void stampede();
     double calculateRiskOfStampede();
     void killMufasa();
}
```

By themselves, interfaces to do nothing. It is impossible to instantiate an instance of an interface type.

```
// Can't do this!
Stampedable giantHerdOfBuffalo = new Stampedable();
```

Interfaces are able to serve as apparent types for any class which implements them.

```
public WildebeestHerd implements Stampedable{
     public void stampede(){
          System.out.println("Run everybody, run!");
     }
     public double calculateRiskOfStampede(){
          return numFortranBooks / numHyenas;
     }
     public void killMufasa(){
          System.out.println("Long live the king.");
     }
}

// Treat the Wildebeest Herd as a Stampedable
WildebeestHerd herd = new WildebeestHerd();
Stampedable stampedableThing = herd;
```

They are useful for algorithms that process objects of different classes.

```
// Suppose the HyenaPack class defined a causeStampede
// method like so:
public void causeStampede(Stampedable s);

// We could call it like so:
hyenaPack.causeStampede(herd);
```

Allowed Members

Instance Methods	Class Methods	Instance Fields	Class Fields
Yes	No	No	Yes

Other Rules

In Interface Definitions...

1. All members are public by default.
2. Methods cannot be given implementations (bodies).
3. Interfaces are not instantiable.

Classes which implement an interface...

4. Must provide bodies for all methods declared in the interface.

Comparable Quick Reference

The old fashioned way:

```
public class Soup implements Comparable{
     private int spiciness;

     /**
      * Compares this object with the specified object for
      * order. Returns a negative integer, zero, or a
      * positive integer as this object is less than, equal
      * to, or greater than the specified object.
     */
     public int compareTo(Object other){
          Soup otherSoup = (Soup) other;
          return spiciness - otherSoup.spiciness;
     }
}
```

The better way with generics:

```
public class Soup implements Comparable<Soup>{
     private int spiciness;

     public int compareTo(Soup otherSoup){
          return spiciness - otherSoup.spiciness;
     }
}
```