Interfaces and Abstract Classes

Look at the `interface`

```
package java.lang;
/**/
public interface Comparable<T> {
  int compareTo(T o);
}
```

Source code from java.lang

method declaration. This method must be defined in a class

The abstract class Currency implements the `Interface`

```
public abstract class Currency implements Comparable<Currency> {
  /**/
  public Currency () { }
  /**/
  public abstract double getUSDvalue ();
  /**/
  public int compareTo ( Currency c ) {
    /**/
    double dif=100.0*(getUSDvalue()-c.getUSDvalue());
    return (int)(dif+0.5);
  }
}
```

method declaration, The method must be declared in a non-abstract class

round to the nearest int

* (1) An interface cannot be instantiated directly, but a variable may be declared as Comparable<Currency> var;
* (2) An abstract class cannot be instantiated directly, but a variable may be declared as Currency C;

The class Peso extends Currency

Peso is a sub-class of Currency

```
public class Peso extends Currency {
  /**/
  private static double VALUE = 0.053; // in USD as of Mar 27 2017
  private double amount;
  /*
   *  pesoAmount must be positive
   */
  public Peso ( double pesoAmount ) {
    amount=pesoAmount;
  }
  /*
   *  abstract in Currency
   */
  public double getUSDvalue() {
    return amount*VALUE;
  }
  /**/
  public static String toString ( double usdAmount ) {
    /**/
    double pesoAmount=usdAmount/VALUE;
    Peso peso=new Peso(pesoAmount);
    return peso.toString();
  }
  /**/
  public String toString() {
    /**/
    int numWhole=(int)Math.log10(amount)+1;
    String wholeS;
    if ( numWhole == 0 )   wholeS = "4";
    else                   wholeS = Integer.toString(numWhole+3);
    String formatS = "%" + wholeS + ".2f";
    return "Mex$" + String.format(formatS,amount);
  }
}
```

* Declaring the same object in different ways is called "polymorphism"

Certain java classes require the use of the Comparable interface.
For example,

```
import java.util.ArrayList;
import java.util.Collections;

ArrayList<Currency> currencyList;
Collections.<Currency>sort(currencyList); // requires Comparable<Currency>
```

will sort `currencyList` into ascending order.

other notes:

(a) One may write interfaces that are not already in java

(b) Non-abstract classes may also be sub-classed (extends)

Example: Class PesoTest

```
public class PesoTest {
  /**/
  public static void main ( String [] arg ) {
    /**/
    Currency p1=new Peso(10.00);
    Currency p2=new Peso(22.00);
    /**/
    System.out.println();
    System.out.println( "p1 = " + p1.toString() );
    System.out.println( "p2 = " + p2.toString() );
    /**/
    System.out.println();
    System.out.println( "p1 = $" + String.format("%4.2f",p1.getUSDvalue()) );
    System.out.println( "p2 = $" + String.format("%4.2f",p2.getUSDvalue()) );
    /**/
    System.out.println();
    System.out.println("$1.00 = " + Peso.toString(1.00));
    /**/
    System.out.println();
    if ( p1.compareTo(p2) < 0 ) System.out.println("p1 < p2" );
    else                        System.out.println("p1 >= p2");
  }
  /**/
  public PesoTest () { }
}
```

produces the output

```
p1 = Mex$10.00
p2 = Mex$22.00

p1 = $0.53
p2 = $1.17

$1.00 = Mex$18.87

p1 < p2
```