Comparing Objects in Java

There are two ways to compare or order instances of Java objects.

1. Implement the Comparable (*java.lang.Comparable.java*) interface. You will have to implement the *compareTo()* method of this interface. The String class, for example, implements this interface. Once this method is implemented, the natural order of comparison will use this method. If you call the *sort* method of the Collection Framework on a Collection of objects with no additional arguments, this comparison will be used. Here is one example:

```
/** A simple example of using the Comparable interface **/
public class A implements Comparable<A> {
   int i;
   String s;
   public A(int i, String s){
         this.i=i;
         this.s=s;
   }

   /* here we define a compareTo() method based on the String field */
   public int compareTo(A a){
         return this.s.compareTo(a.s); //We are using the built-in lexical String comparison.
   }

   public static void main(String[] args){
         A c1= new A(new Integer(args[0]), args[1]);
         A c2= new A(new Integer(args[2]), args[3]);
         System.out.println(c1.compareTo(c2));
   }
}
```

Sample output:

```
$ java A 1 "hello" 2 "hello"
0
$ java A 1 "hello" 2 "ello"
3
$ java A 2 "ello" 1 "hello"
-3
```

2. Define one or more classes that implement the *Comparator* interface (*java.util.Comparator.java*). Use this for more complex comparisons than what can be done with *compareTo()* method of the *Comparable* interface. You can define multiple Comparator implementations for the same object, and pass instances of these classes as arguments to the *Collections.sort* method in the Collection Framework to achieve different types of comparisons and sorting.

Note: *Collections.sort*, despite the name, sorts only Lists, not all possible Collection objects.

Here is a complete example that uses multiple Comparator classes to sort objects in a Collection according to different criteria:

```
import java.util.*;

class Student implements Comparable<Student>{
   int Id;
   String Name;
   int Major;
   public Student(int i, String n, int m){
         Id=i;
         Name=n;
         Major=m;
   }

   public String toString(){
         return "Name->"+ Name + "  ID->" + Id + "  Major->"+Major;
   }

   public boolean equals(Student x){
         return (this.Id==x.Id && this.Name.equals(x.Name));
   }

   /**
         Here we define how a Collections.sort call without a comparator argument
         should work  The comparison is based on the Name string
   **/
   public int compareTo(Student s){
         return this.Name.compareTo(s.Name); //We're using compareTo() method of Sting
class
   }

}


/** This is a Comparator based on Student ID **/
class StudentComparator1 implements Comparator<Student> {
   public int compare(Student s1, Student s2){
         return s1.Id-s2.Id;
   }
}

/** This is a Comparator based on Student Majors **/
class StudentComparator2 implements Comparator<Student> {
   public int compare(Student s1, Student s2){
         return s1.Major-s2.Major;
   }
}

/** Test **/

class TestStudent {

  /** Print this List **/
   public static void printList(List L){
         Iterator it=L.iterator();
         while(it.hasNext()){
```

```
            System.out.println(it.next());
        }
    }

    public static void main(String[] args){
        Student s1=new Student(10, "FITZGIBBON RICHARD ", 2);
        Student s2=new Student(12, "CRAMER HARRY", 2);
        Student s3=new Student(13, "OVNAND CHESTER", 4);
        Student s4=new Student(14, "BUIS DALE", 1);
        Student s5=new Student(15, "NEWTON WILLIAM", 3);
        Student s6=new Student(16, "MULLINS ROGER", 2);
        Student s7=new Student(18, "ALEXANDER GEORGE", 4);
        Student s8=new Student(20, "FLOURNOY MAURICE", 3);
        Student s9=new Student(23, "STEPHAN RICHARD", 1);
        Student s10=new Student(24, "CRESS TOM", 3);
        Student s11=new Student(26, "MAGEE RALPH", 2);
        Student s12=new Student(25, "MATTESON GLENN", 4);
        Student s13=new Student(26, "SAMPSON LESLIE", 1);
        Student s14=new Student(28, "WEITKAMP EDGAR", 3);
        Student s15=new Student(31, "WESTON OSCAR", 4);
        Student s16=new Student(32, "BANKOWSKI ALFONS", 3);
        Student s17=new Student(33, "GARSIDE FREDERICK", 2);
        Student s18=new Student(34, "FELAND THEODORE", 1);
        Student s19=new Student(35, "BISCHOFF JOHN", 2);
        Student s20=new Student(36, "BIBER GERALD", 3);
        Student[]
s={s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20};
        ArrayList<Student> aL=new ArrayList<Student>();
        for (int i=0; i<20;i++){
              aL.add(s[i]);
          }
          System.out.println("......Original ArrayList......");
          printList(aL);
          System.out.println("......Sorted according to Name......");
          Collections.sort(aL);
          printList(aL);
          System.out.println("......Sorted according to Id.......");
          StudentComparator1 cmp1=new StudentComparator1();
          Collections.sort(aL, cmp1);
          printList(aL);
          System.out.println("......Sorted according to Major.......");
          StudentComparator2 cmp2=new StudentComparator2();
          Collections.sort(aL, cmp2);
          printList(aL);

          System.out.println(".....Creating a Linked List.....");

        LinkedList<Student> LL=new LinkedList<Student>();
        for (int i=0; i<20;i++){
              LL.add(s[i]);
          }
          printList(LL);
          System.out.println(".....Reversing the Linked List.....");
          Collections.reverse(LL);
          printList(LL);
    }
}
```

Output:

```
$ java TestStudent
......Original ArrayList......
Name->FITZGIBBON RICHARD   ID->10  Major->2
Name->CRAMER HARRY  ID->12  Major->2
Name->OVNAND CHESTER  ID->13  Major->4
Name->BUIS DALE  ID->14  Major->1
Name->NEWTON WILLIAM  ID->15  Major->3
Name->MULLINS ROGER  ID->16  Major->2
Name->ALEXANDER GEORGE  ID->18  Major->4
Name->FLOURNOY MAURICE  ID->20  Major->3
Name->STEPHAN RICHARD  ID->23  Major->1
Name->CRESS TOM  ID->24  Major->3
Name->MAGEE RALPH  ID->26  Major->2
Name->MATTESON GLENN  ID->25  Major->4
Name->SAMPSON LESLIE  ID->26  Major->1
Name->WEITKAMP EDGAR  ID->28  Major->3
Name->WESTON OSCAR  ID->31  Major->4
Name->BANKOWSKI ALFONS  ID->32  Major->3
Name->GARSIDE FREDERICK  ID->33  Major->2
Name->FELAND THEODORE  ID->34  Major->1
Name->BISCHOFF JOHN  ID->35  Major->2
Name->BIBER GERALD  ID->36  Major->3
......Sorted according to Name......
Name->ALEXANDER GEORGE  ID->18  Major->4
Name->BANKOWSKI ALFONS  ID->32  Major->3
Name->BIBER GERALD  ID->36  Major->3
Name->BISCHOFF JOHN  ID->35  Major->2
Name->BUIS DALE  ID->14  Major->1
Name->CRAMER HARRY  ID->12  Major->2
Name->CRESS TOM  ID->24  Major->3
Name->FELAND THEODORE  ID->34  Major->1
Name->FITZGIBBON RICHARD   ID->10  Major->2
Name->FLOURNOY MAURICE  ID->20  Major->3
Name->GARSIDE FREDERICK  ID->33  Major->2
Name->MAGEE RALPH  ID->26  Major->2
Name->MATTESON GLENN  ID->25  Major->4
Name->MULLINS ROGER  ID->16  Major->2
Name->NEWTON WILLIAM  ID->15  Major->3
Name->OVNAND CHESTER  ID->13  Major->4
Name->SAMPSON LESLIE  ID->26  Major->1
Name->STEPHAN RICHARD  ID->23  Major->1
Name->WEITKAMP EDGAR  ID->28  Major->3
Name->WESTON OSCAR  ID->31  Major->4
......Sorted according to Id.......
Name->FITZGIBBON RICHARD   ID->10  Major->2
Name->CRAMER HARRY  ID->12  Major->2
Name->OVNAND CHESTER  ID->13  Major->4
Name->BUIS DALE  ID->14  Major->1
Name->NEWTON WILLIAM  ID->15  Major->3
Name->MULLINS ROGER  ID->16  Major->2
Name->ALEXANDER GEORGE  ID->18  Major->4
Name->FLOURNOY MAURICE  ID->20  Major->3
Name->STEPHAN RICHARD  ID->23  Major->1
Name->CRESS TOM  ID->24  Major->3
Name->MATTESON GLENN  ID->25  Major->4
```

```
Name->MAGEE RALPH  ID->26  Major->2
Name->SAMPSON LESLIE  ID->26  Major->1
Name->WEITKAMP EDGAR  ID->28  Major->3
Name->WESTON OSCAR  ID->31  Major->4
Name->BANKOWSKI ALFONS  ID->32  Major->3
Name->GARSIDE FREDERICK  ID->33  Major->2
Name->FELAND THEODORE  ID->34  Major->1
Name->BISCHOFF JOHN  ID->35  Major->2
Name->BIBER GERALD  ID->36  Major->3
.....Sorted according to Major.....
Name->BUIS DALE  ID->14  Major->1
Name->STEPHAN RICHARD  ID->23  Major->1
Name->SAMPSON LESLIE  ID->26  Major->1
Name->FELAND THEODORE  ID->34  Major->1
Name->FITZGIBBON RICHARD   ID->10  Major->2
Name->CRAMER HARRY  ID->12  Major->2
Name->MULLINS ROGER  ID->16  Major->2
Name->MAGEE RALPH  ID->26  Major->2
Name->GARSIDE FREDERICK  ID->33  Major->2
Name->BISCHOFF JOHN  ID->35  Major->2
Name->NEWTON WILLIAM  ID->15  Major->3
Name->FLOURNOY MAURICE  ID->20  Major->3
Name->CRESS TOM  ID->24  Major->3
Name->WEITKAMP EDGAR  ID->28  Major->3
Name->BANKOWSKI ALFONS  ID->32  Major->3
Name->BIBER GERALD  ID->36  Major->3
Name->OVNAND CHESTER  ID->13  Major->4
Name->ALEXANDER GEORGE  ID->18  Major->4
Name->MATTESON GLENN  ID->25  Major->4
Name->WESTON OSCAR  ID->31  Major->4
.....Creating a Linked List.....
Name->FITZGIBBON RICHARD   ID->10  Major->2
Name->CRAMER HARRY  ID->12  Major->2
Name->OVNAND CHESTER  ID->13  Major->4
Name->BUIS DALE  ID->14  Major->1
Name->NEWTON WILLIAM  ID->15  Major->3
Name->MULLINS ROGER  ID->16  Major->2
Name->ALEXANDER GEORGE  ID->18  Major->4
Name->FLOURNOY MAURICE  ID->20  Major->3
Name->STEPHAN RICHARD  ID->23  Major->1
Name->CRESS TOM  ID->24  Major->3
Name->MAGEE RALPH  ID->26  Major->2
Name->MATTESON GLENN  ID->25  Major->4
Name->SAMPSON LESLIE  ID->26  Major->1
Name->WEITKAMP EDGAR  ID->28  Major->3
Name->WESTON OSCAR  ID->31  Major->4
Name->BANKOWSKI ALFONS  ID->32  Major->3
Name->GARSIDE FREDERICK  ID->33  Major->2
Name->FELAND THEODORE  ID->34  Major->1
Name->BISCHOFF JOHN  ID->35  Major->2
Name->BIBER GERALD  ID->36  Major->3
.....Reversing the Linked List.....
Name->BIBER GERALD  ID->36  Major->3
Name->BISCHOFF JOHN  ID->35  Major->2
Name->FELAND THEODORE  ID->34  Major->1
Name->GARSIDE FREDERICK  ID->33  Major->2
Name->BANKOWSKI ALFONS  ID->32  Major->3
Name->WESTON OSCAR  ID->31  Major->4
```

```
Name->WEITKAMP EDGAR  ID->28  Major->3
Name->SAMPSON LESLIE  ID->26  Major->1
Name->MATTESON GLENN  ID->25  Major->4
Name->MAGEE RALPH  ID->26  Major->2
Name->CRESS TOM  ID->24  Major->3
Name->STEPHAN RICHARD  ID->23  Major->1
Name->FLOURNOY MAURICE  ID->20  Major->3
Name->ALEXANDER GEORGE  ID->18  Major->4
Name->MULLINS ROGER  ID->16  Major->2
Name->NEWTON WILLIAM  ID->15  Major->3
Name->BUIS DALE  ID->14  Major->1
Name->OVNAND CHESTER  ID->13  Major->4
Name->CRAMER HARRY  ID->12  Major->2
Name->FITZGIBBON RICHARD   ID->10  Major->2
$
```

Comparing Objects in Java - Another example

In this example, a class A has two fields, an int i and a String s. The "natural" comparison of A Objects is defined to be based on i (this is just a matter of definition - there is nothing natural about it in a usual sense). The code below shows how to implement a Comparator to do comparisons based on the String field. A test program shows sorting a List using both criteria.
Notice how we parameterize many things with <A>.

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

class A implements Comparable<A>{
   public int i;
   public String s;

   /*Constructor */
   public A(int i, String s){
         this.i=i;
         this.s=s;
   }

   public String toString(){
         return "A(i="+i+", s="+s+")";
   }

   /* compareTo method */
   public int compareTo(A a){
            return (this.i-a.i);
         }
}

class testA{
   public static void main(String[] args){
         A a1=new A(10, "lorem");
         A a2=new A(5, "ipsum");
         A a3=new A(9, "dolor");
         A a4=new A(23, "amet");
         A a5=new A(20, "interdum");
```

```
        A a6=new A(16, "adipiscing");
        ArrayList<A> AL=new ArrayList<A>();
        AL.add(a1);
        AL.add(a2);
        AL.add(a3);
        AL.add(a4);
        AL.add(a5);
        AL.add(a6);
        System.out.println("----Original list -----");
        for (A x : AL){
           System.out.println(x);
        }
        System.out.println("----Sorting according to natural order (int field) -----");
        Collections.sort(AL);
        for (A b : AL){
           System.out.println(b);
        }
        System.out.println("----Sorting according to the String field -----");
        Collections.sort(AL, new cmp1());
        for (A b : AL){
           System.out.println(b);
        }
   }
}

class cmp1 implements Comparator<A> {
   public int compare(A a1, A a2){
           return (a1.s).compareTo(a2.s);
   }
}
```

OUTPUT:

```
$ java testA
----Original list -----
A(i=10, s=lorem)
A(i=5, s=ipsum)
A(i=9, s=dolor)
A(i=23, s=amet)
A(i=20, s=interdum)
A(i=16, s=adipiscing)
----Sorting according to natural order (int field) -----
A(i=5, s=ipsum)
A(i=9, s=dolor)
A(i=10, s=lorem)
A(i=16, s=adipiscing)
A(i=20, s=interdum)
A(i=23, s=amet)
----Sorting according to the String field -----
A(i=16, s=adipiscing)
A(i=23, s=amet)
A(i=9, s=dolor)
A(i=20, s=interdum)
A(i=5, s=ipsum)
A(i=10, s=lorem)
$
```