Sorting Arrays and Collections

```
// Printout the command line args in sorted order:

import java.util.*;

public class sortArgs {
    public static void main(String args[]) {
       Arrays.sort(args);
       System.out.println(Arrays.asList(args));
    }
}
```

```
import java.util.*;

public class Sort {
   public static void main(String args[]) {
       List l = Arrays.asList(args);
       Collections.sort(l);
       System.out.println(l);
   }
}
```

Sorting with Comparators

Sort the array in reverse alphabetical order, ignoring case.

```
import java.util.*;

class Temp {
      public static void main (String[] args) {
         String [] word =
      {"One","Two","three","Four","five","Six","seven"};
         Arrays.sort(word,new myComparator());
         System.out.println(Arrays.asList(word));
       }

static class myComparator implements Comparator {
   public int compare (Object x, Object y) {
      return -String.CASE_INSENSITIVE_ORDER.compare(x,y);
    }
}
```

prints: **[three, seven, five, Two, Six, One, Four]**

```
import java.util.*;

class Temp implements Comparator {
      public static void main (String[] args) {
         String [] word = {"One","Two","three","Four","five","Six","seven"
         Arrays.sort(word,new Temp());
         System.out.println(Arrays.asList(word));
       }
      public int compare (Object x, Object y) {
         return -String.CASE_INSENSITIVE_ORDER.compare(x,y);
       }
}
```

Comparators versus Methods

```
import java.util.*;
import java.lang.reflect.*;

// Matthew Morgenstern, Nov 2000

class mySort {
   public static void main (String[] args) {
      // GET METHOD FROM myComparator OBJECT:
     Class cl = null;
      Method meth = null;
      try{
       cl = myComparator.class;
       meth = cl.getMethod("compare",
                           new Class[]{Object.class, Object.class} );
      } catch(Exception e) {e.printStackTrace();}
        // ClassNotFoundException, NoSuchMethodException
        System.out.println("\nMethod from myComparator:   ");
       System.out.println(meth);

         // FROM THE METHOD compare GET THE
         // COMPARATOR OBJECT INSTANCE:
         System.out.print("\nComparator Object Instance:  ");
         Class compClass = meth.getDeclaringClass();
         Comparator compObject = null;
         try {
             compObject = (Comparator)compClass.newInstance()
             System.out.println(compObject);
             } catch (Exception e) {e.printStackTrace();}
               // IllegalAccessException, InstantiationException
        System.out.print("compObject instanceOf Comparator --> " )
```

```
        System.out.print("which is different from "+
                "\n      (new Integer(5)) instanceof Comparable -->
        System.out.println((new Integer(5)) instanceof
Comparable);

. . . . . .
```

prints:

```
Method from myComparator:
public int myComparator.compare(java.lang.Object,java.lang.Object)

Comparator Object Instance:  myComparator@70e54ae1
compObject instanceOf Comparator --> true
which is different from
      (new Integer(5)) instanceof Comparable --> true
```

```
        // USE COLLECTIONS.SORT
        // with COMPARATOR OBJECT INSTANCE:
      String [] wordcopy = (String[])word.clone();
      List lst = Arrays.asList(args);
      Collections.sort(lst, compObject);
      System.out.println("\nCollections.sort of args in reverse Sorted order:  "
      System.out.println(Arrays.asList(word));

        // DEMONSTRATE EXCEPTION STACK TRACING:
        System.out.println("\nDemo of Exception Catch and reporting:");
        try {
            meth = cl.getMethod("compare", new Class[0]);
            }  catch (Exception e) {e.printStackTrace();}
        System.out.println("And program is still running, running, ....");
    }
}
```

prints:

```
Collections.sort of args in reverse Sorted order:
[three, seven, five, Two, Six, One, Four]

Demo of Exception Catch and reporting:

java.lang.NoSuchMethodException: compare
        at java.lang.Class.getMethod0(Native Method)
        at java.lang.Class.getMethod(Class.java:854)
        at mySort.main(comparators.java)

And program is still running, running, ....
```

Full Printout:

[three, seven, five, Two, Six, One, Four]

Method from myComparator:
public int
myComparator.compare(java.lang.Object,java.lang.Object)

Comparator Object Instance: myComparator@70e54ae1
compObject instanceOf Comparator --> true
which is different from
(new Integer(5)) instanceof Comparable --> true

Collections.sort of args in reverse Sorted order:
[three, seven, five, Two, Six, One, Four]

Demo of Exception Catch and reporting:
java.lang.NoSuchMethodException: compare
at java.lang.Class.getMethod0(Native Method)
at java.lang.Class.getMethod(Class.java:854)
at mySort.main(comparators.java)
And program is still running, running, ...

Press Enter to continue