```java
package sort;

abstract class SortableDataDelegation<T> implements SortableData<T>
 {

    private SortableData<? extends T> data;

    public SortableDataDelegation(SortableData<? extends T> data) {
        this.data = data;
    }

    public int size() {
        return data.size();
    }

    public void swap(int i, int j) {
        data.swap(i, j);
    }

    public T get(int i) {
        return data.get(i);
    }

    public int compare(int i, int j) {
        return data.compare(i, j);
    }

}
```

SortableDataWithTracer.java

```java
package sort;

public class SortableDataWithTracer<T> extends
 SortableDataDelegation<T> implements SortableData<T> {

    public SortableDataWithTracer(SortableData<? extends T> data) {
        super(data);
    }

    public void swap(int i, int j) {
        System.out.println("Swapping " + i + " " + j);
        super.swap(i, j);
    }

    public int compare(int i, int j) {
        System.out.println("Comparing " + i + " " + j);
        return super.compare(i, j);
    }
}
```

SortableDataWithStatistics.java

```
 1 package sort;
 2
 3 public class SortableDataWithStatistics<T> extends
   SortableDataDelegation<T>
 4         implements SortableData<T> {
 5
 6     int nswap;
 7
 8     int ncompare;
 9
10     public SortableDataWithStatistics(SortableData<? extends T>
   data) {
11         super(data);
12         nswap = ncompare = 0;
13     }
14
15     public void swap(int i, int j) {
16         nswap++;
17         super.swap(i, j);
18     }
19
20     public int compare(int i, int j) {
21         ncompare++;
22         return super.compare(i, j);
23     }
24
25     public int getSwapStat() {
26         return nswap;
27     }
28
29     public int getCompareStat() {
30         return ncompare;
31     }
32 }
```

```java
import java.util.ArrayList;
import java.util.List;
import sort.InsertionSort;
import sort.QuickSort;
import sort.Sort;
import sort.SortableComparableData;
import sort.SortableData;
import sort.SortableDataWithStatistics;
import sort.SortableDataWithTracer;
import sort.SwapableList;

public class TestSort {

    public static void sortAndPrint(SortableData<?> sd, Sort sort)
  {
        sort.doSort(sd);
        for (int i = 0; i < sd.size(); ++i) {
            System.out.print(sd.get(i) + " ");
        }
        System.out.println();
    }

    public static <T> void initList(List<T> l, T[] a) {
        l.clear();
        for (T t : a) {
            l.add(t);
        }
    }

    public static void main(String[] args) {
        Sort isort = new InsertionSort();
        Sort qsort = new QuickSort();
        List<String> ls = new ArrayList<String>();

        initList(ls, args);
        SortableDataWithStatistics<String> sdws = new
  SortableDataWithStatistics<String>(
                new SortableDataWithTracer<String>(
                        new SortableComparableData<String>(
                                new SwapableList<String>(ls))));
        sortAndPrint(sdws, isort);
        System.out.println("Stats: " + sdws.getSwapStat() + " swap,
  "
                + sdws.getCompareStat() + " compare");

        initList(ls, args);
        sdws = new SortableDataWithStatistics<String>(
                new SortableDataWithTracer<String>(
                        new SortableComparableData<String>(
                                new SwapableList<String>(ls))));
        sortAndPrint(sdws, qsort);
        System.out.println("Stats: " + sdws.getSwapStat() + " swap,
  "
                + sdws.getCompareStat() + " compare");
    }

}
```

```java
package sort;

import java.util.Comparator;
import java.util.List;

public class Sorts {

    public static final QuickSort QUICKSORT = new QuickSort();

    public static final InsertionSort INSERTION_SORT = new InsertionSort();

    private static void sort(Sort s, SortableData<?> data) {
        s.doSort(data);
    }

    public static <T extends Comparable<? super T>> void quickSort(T[] array) {
        sort(QUICKSORT, new SortableComparableData<T>(new SwapableArray<T>(array)));
    }

    public static <T> void quickSort(T[] array, Comparator<? super T> comparator) {
        sort(QUICKSORT, new SortableDataWithComparator<T>(new SwapableArray<T>(array),
                comparator));
    }

    public static <T extends Comparable<? super T>> void quickSort(List<T> l) {
        sort(QUICKSORT, new SortableComparableData<T>(new SwapableList<T>(l)));
    }

    public static <T> void quickSort(List<T> l, Comparator<? super T> comparator) {
        sort(QUICKSORT, new SortableDataWithComparator<T>(new SwapableList<T>(l),
                comparator));
    }

    public static <T extends Comparable<? super T>> void insertionSort(T[] array) {
        sort(INSERTION_SORT, new SortableComparableData<T>(new SwapableArray<T>(array)));
    }

    public static <T> void insertionSort(T[] array,
            Comparator<? super T> comparator) {
        sort(INSERTION_SORT, new SortableDataWithComparator<T>(new SwapableArray<T>(array),
                comparator));
    }

    public static <T extends Comparable<? super T>> void insertionSort(List<T> l) {
```

```
46          sort(INSERTION_SORT, new SortableComparableData<T>(new
  SwapableList<T>(l)));
47      }
48
49      public static <T> void insertionSort(List<T> l,
50              Comparator<? super T> comparator) {
51          sort(INSERTION_SORT, new SortableDataWithComparator<T>(new
  SwapableList<T>(l),
52                  comparator));
53      }
54
55  }
```