

```
// CS 3723. Example showing how to pass a function as
// a parameter in Java. The function is embedded in an
// interface, and then different implementations of the
// interface are used.

// Sorter.java: interface holds a generic sort routine
public interface Sorter {
    public void sort(int[] a);
}

// BubbleSort.java: implements sort using a bubblesort
public class BubbleSort implements Sorter {
    // sort: really simple bubblesort algorithm
    public void sort(int[] a) {
        System.out.println("Bubble Sort:");
        int size = a.length;
        for (int dum = 0; dum < size-1; dum++)
            for (int j = 0; j < size-1; j++)
                if (a[j] > a[j+1]) {
                    int temp = a[j];
                    a[j] = a[j+1];
                    a[j+1] = temp;
                }
    }
}

// SelectSort.java: implements sort using a selectsort
public class SelectSort implements Sorter {
    // sort: fairly reasonable selectsort, except for the recursion
    public void sort(int[] a) {
        System.out.println("Select Sort:");
        int n = a.length - 1;
        int maxi;
        while (n > 0) {
            maxi = maxInd(n, a);
            int temp = a[maxi];
            a[maxi] = a[n];
            a[n] = temp;
            n--;
        }

        // maxInt: recursive function just for demonstration purposes
        private int maxInd(int m, int[] a) {
            int maxi;
            if (m == 0) return m;
            maxi = maxInd(m-1, a);
            if (a[m] > a[maxi]) return m;
            return maxi;
        }
    }

    // MultiSort.java: Demonstration of passing a function as a parameter
    public class MultiSort {
        int a[]; // array of ints to be sorted
        Sorter s[]; // array of references to objects holding sort functions

        // MultiSort: constructor to allocate stuff
        public MultiSort() {
            a = new int[10]; // actual array of 10 ints
            s = new Sorter[2]; // actual array of 2 refs to objects
            s[0] = new BubbleSort(); // object holding a bubblesort function
            s[1] = new SelectSort(); // object holding a selectsort function
        }
    }
}

```

```
// DoSorts: invoke the two sort routines in order
public void DoSorts() {
    for (int i = 0; i < s.length; i++)
        execSort(a, s[i]);
}

// execSort: actually sort, using any sort function
private void execSort(int[] a, Sorter someSort) {
    initArr(a);
    someSort.sort(a);
    printArr(a);
}

// initArr: initialize parameter to hold random ints
private void initArr(int[] a) {
    for (int i = 0; i < a.length; i++)
        a[i] = (int)(1000.0*Math.random());
}

// printArr: print parameter on one line
private void printArr(int[] a) {
    for (int i = 0; i < a.length; i++)
        System.out.print(a[i] + " ");
    System.out.println();
}

// main: just create the class and start the sorting
public static void main(String[] args) {
    MultiSort ms = new MultiSort();
    ms.DoSorts();
}

ten60% javac Sorter.java
ten60% javac BubbleSort.java
ten60% javac SelectSort.java
ten60% javac MultiSort.java
ten60% java MultiSort

Bubble Sort:
29 35 48 355 376 656 692 702 732 868
Select Sort:
40 251 310 345 505 515 925 967 972 987

```