**ArrayUtil**

import java.util.Random;  
  
public class ArrayUtil {  
  
 public static void displayArrayContent(Object [] data)  
 {  
 System.*out*.println(*getString*(data));  
 }  
  
 private static String getString(Object [] data)  
 {  
 String resultString = new String("[ ");  
  
 for(int i = 0; i< data.length; i++) {  
 resultString = resultString + data[i].toString() + " ";  
 }  
 resultString = resultString + "]";  
  
 return resultString;  
 }  
  
 public static Integer[] generateRandomArray(int size)  
 {  
 Integer resultArray[] = new Integer[size];  
 Random generator = new Random();  
  
 for(int i = 0; i< size; i++) {  
 int value = generator.nextInt(size);  
 resultArray[i] = value;  
 }  
 return resultArray;  
 }  
  
 public static Integer[] duplicateArray(Object [] orig)  
 {  
 int size = orig.length;  
 Integer resultArray[] = new Integer[size];  
  
 for(int i = 0; i< size; i++) {  
 resultArray[i] = (Integer) orig[i];  
 }  
 return resultArray;  
 }  
}

**SwapsTest**

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
  
public class SwapsTest {  
  
 public static void main(String args[]){  
  
 PrintWriter pw = null;  
  
 try {  
 pw = new PrintWriter(new File("SwapsData.csv"));  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 StringBuilder builder = new StringBuilder();  
 String ColumnNamesList = "Array Length,Bubble Swaps,Selection Swaps,Shell Swaps";  
 builder.append(ColumnNamesList +"\n");  
  
 Integer initialData[];  
 Integer bubbleSortData[];  
 Integer selectionSortData[];  
 Integer shellSortData[];  
 Integer ordered[] = new Integer[100];  
 Integer reverse[] = new Integer[100];  
  
  
 long bubbleSwaps = 0;  
 long selectionSwaps = 0;  
 long shellSwaps = 0;  
 int bubbleMin = 0;  
 int selectionMin = 0;  
 int shellMin = 0;  
 int shellMax = 0;  
 int bubbleMax = 0;  
 int selectionMax = 0;  
  
 for(int i = 0; i<=99; i++){  
  
 ordered[i] = i;  
 reverse[i] = 100 -i;  
  
 }  
  
 for(int j = 0; j <= 99; j++) {  
  
// initialData = ArrayUtil.generateRandomArray(100);  
// bubbleSortData = ArrayUtil.duplicateArray(initialData);  
// bubbleSortData = ordered;  
 bubbleSortData = reverse;  
  
 int returnVal = BubbleSortArray.*bubbleSort*(bubbleSortData);  
  
 bubbleSwaps = bubbleSwaps + returnVal;  
  
 if(j == 0) {  
  
 bubbleMin = returnVal;  
  
 }  
  
 if(returnVal >= bubbleMax){  
  
 bubbleMax = returnVal;  
  
 } else if(returnVal <= bubbleMin){  
  
 bubbleMin = returnVal;  
  
 }  
  
 }  
 bubbleSwaps = bubbleSwaps / 100;  
  
 for(int j = 0; j <= 99; j++) {  
// initialData = ArrayUtil.generateRandomArray(100);  
// selectionSortData = ArrayUtil.duplicateArray(initialData);  
// selectionSortData = ordered;  
 selectionSortData = reverse;  
  
 int returnVal = SelectionSortArray.*selectionSort*(selectionSortData);  
  
 selectionSwaps = selectionSwaps + returnVal;  
  
 if(j == 0) {  
  
 selectionMin = returnVal;  
  
 }  
  
 if(returnVal >= selectionMax){  
  
 selectionMax = returnVal;  
  
 } else if(returnVal <= selectionMin){  
  
 selectionMin = returnVal;  
  
 }  
  
 }  
 selectionSwaps = selectionSwaps / 100;  
  
 for(int j = 0; j <= 99; j++) {  
  
// initialData = ArrayUtil.generateRandomArray(100);  
// shellSortData = ArrayUtil.duplicateArray(initialData);  
// shellSortData = ordered;  
 shellSortData = reverse;  
  
 int returnVal = ShellSortArray.*shellSort*(shellSortData);  
  
 shellSwaps = shellSwaps + returnVal;  
  
 if(j == 0) {  
  
 shellMin = returnVal;  
  
 }  
  
 if(returnVal >= shellMax){  
  
 shellMax = returnVal;  
  
 } else if(returnVal <= shellMin){  
  
 shellMin = returnVal;  
  
 }  
  
 }  
 shellSwaps = shellSwaps / 100;  
  
 System.*out*.println("Bubble Swaps: " + bubbleSwaps);  
 System.*out*.println("Bubble Min: " + bubbleMin);  
 System.*out*.println("Bubble Max: " + bubbleMax);  
 System.*out*.println();  
  
 System.*out*.println("Selection Swaps: " + selectionSwaps);  
 System.*out*.println("Selection Min: " + selectionMin);  
 System.*out*.println("Selection Max: " + selectionMax);  
 System.*out*.println();  
  
 System.*out*.println("Shell Swaps: " + shellSwaps);  
 System.*out*.println("Shell Min: " + shellMin);  
 System.*out*.println("Shell Max: " + shellMax);  
 System.*out*.println();  
  
 builder.append(bubbleSwaps + "," + selectionSwaps + "," + shellSwaps + "\n");  
  
 pw.write(builder.toString());  
 pw.close();  
  
 }  
  
}

**ComparisonsTest**

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
  
public class ComparisonsTest {  
  
 public static void main(String args[]){  
  
 final int MAX\_ARR\_LEN = 5000;  
  
 PrintWriter pw = null;  
  
 try {  
 pw = new PrintWriter(new File("SortData.csv"));  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 StringBuilder builder = new StringBuilder();  
 String ColumnNamesList = "Array Length,Bubble Comparisons,Selection Comparisons,Shell Comparisons\n";  
 builder.append(ColumnNamesList);  
  
 Integer initialData[];  
 Integer bubbleSortData[];  
 Integer selectionSortData[];  
 Integer shellSortData[];  
  
 int bubbleTot = 0;  
 int selectionTot = 0;  
 int shellTot = 0;  
  
 int loopCounter = 0;  
 for(int i = 25; i <= MAX\_ARR\_LEN; i += 25){  
  
 loopCounter++;  
  
 initialData = ArrayUtil.*generateRandomArray*(i);  
 bubbleSortData = ArrayUtil.*duplicateArray*(initialData);  
 selectionSortData = ArrayUtil.*duplicateArray*(initialData);  
 shellSortData = ArrayUtil.*duplicateArray*(initialData);  
  
 System.*out*.println("Test No.: " + loopCounter);  
 System.*out*.println("Array Length: " + i);  
 System.*out*.println();  
  
 int bubbleComparisons = BubbleSortArray.*bubbleSort*(bubbleSortData);  
  
 int selectionComparisons = SelectionSortArray.*selectionSort*(selectionSortData);  
  
 int shellComparisons = ShellSortArray.*shellSort*(shellSortData);  
  
 System.*out*.println("Bubble Comparisons: " + bubbleComparisons);  
 System.*out*.println("Selection Comparisons: " + selectionComparisons);  
 System.*out*.println("Shell Comparisons: " + shellComparisons);  
 System.*out*.println();  
  
 builder.append(i + "," + bubbleComparisons + "," + selectionComparisons + "," + shellComparisons + "," + "\n");  
  
 }  
  
// Averages  
 int bubbleAvg = bubbleTot / loopCounter;  
 int selectionAvg = selectionTot / loopCounter;  
 int shellAvg = shellTot / loopCounter;  
 System.*out*.println("Bubble Average: " + bubbleAvg);  
 System.*out*.println("Selection Average: " + selectionAvg);  
 System.*out*.println("Shell Average: " + shellAvg);  
  
 pw.write(builder.toString());  
 pw.close();  
  
 }  
  
}

**TimeTest**

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
  
public class TimeTest {  
  
 public static void main(String args[]){  
  
 PrintWriter pw = null;  
  
 try {  
 pw = new PrintWriter(new File("TimeData.csv"));  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 StringBuilder builder = new StringBuilder();  
 String ColumnNamesList = "Array Length,Bubble Time,Selection Time,Shell Time";  
 builder.append(ColumnNamesList +"\n");  
  
 Integer initialData[];  
 Integer bubbleSortData[];  
 Integer selectionSortData[];  
 Integer shellSortData[];  
  
 int bubbleTot = 0;  
 int selectionTot = 0;  
 int shellTot = 0;  
  
 long bubbleTime = 0;  
 long selectionTime = 0;  
 long shellTime = 0;  
  
 long startTime;  
 long endTime;  
  
 int loopCounter = 0;  
  
 for(int i = 25; i <= 5000; i += 25){  
  
 loopCounter++;  
  
 initialData = ArrayUtil.*generateRandomArray*(i);  
 bubbleSortData = ArrayUtil.*duplicateArray*(initialData);  
 selectionSortData = ArrayUtil.*duplicateArray*(initialData);  
 shellSortData = ArrayUtil.*duplicateArray*(initialData);  
  
 System.*out*.println("Test No.: " + loopCounter);  
 System.*out*.println("Array Length: " + i);  
 System.*out*.println();  
  
 for(int j = 0; j <= 10; j++) {  
 startTime = System.*nanoTime*();  
 BubbleSortArray.*bubbleSort*(bubbleSortData);  
 endTime = System.*nanoTime*();  
 bubbleTime = bubbleTime + (endTime - startTime);  
 }  
 bubbleTime = bubbleTime / 10;  
  
 for(int j = 0; j <= 10; j++) {  
 startTime = System.*nanoTime*();  
 SelectionSortArray.*selectionSort*(selectionSortData);  
 endTime = System.*nanoTime*();  
 selectionTime = endTime - startTime;  
 }  
 selectionTime = selectionTime / 10;  
  
 for(int j = 0; j <= 10; j++) {  
 startTime = System.*nanoTime*();  
 ShellSortArray.*shellSort*(shellSortData);  
 endTime = System.*nanoTime*();  
 shellTime = endTime - startTime;  
 }  
 shellTime = shellTime / 10;  
  
 System.*out*.println("Bubble Time: " + bubbleTime);  
 System.*out*.println("Selection Time: " + selectionTime);  
 System.*out*.println("Shell Time: " + shellTime);  
 System.*out*.println();  
  
 builder.append(i + "," + bubbleTime + "," + selectionTime + "," + shellTime + "\n");  
  
 }  
  
 pw.write(builder.toString());  
 pw.close();  
  
 }  
  
}

**BubbleSortArray**

public class BubbleSortArray {  
  
 public static int bubbleSort(Integer[] array){  
  
 int numberOfComparisons = 0;  
 int numberOfSwaps = 0;  
  
 int arrayLength = array.length;  
 for(int i = 0; i <= arrayLength; i++){  
  
 for(int j = 1; j <= (arrayLength -1); j++) {  
  
 int left = array[j-1];  
 int right = array[j];  
  
 numberOfComparisons++;  
 if (array[j-1] > array[j]) {  
  
 numberOfSwaps++;  
 Integer temp = array[j];  
 array[j] = array[j-1];  
 array[j-1] = temp;  
  
 }  
  
 }  
  
 }  
  
// return numberOfComparisons;  
 return numberOfSwaps;  
  
 }  
  
}

**SelectionSortArray**

public class SelectionSortArray {  
  
 public static int selectionSort(Integer[] array){  
  
 int numberOfComparisons = 0;  
 int numberOfSwaps = 0;  
  
 int arrayLength = array.length;  
 for(int i = 0; i <= arrayLength -1; i++){  
  
 int first = i;  
 int indexOfCurrentMinimum = first;  
 int smallest;  
  
 for(int j = first; j <= arrayLength -1; j++){  
  
 numberOfComparisons++;  
 if(array[j] <= array[indexOfCurrentMinimum]){  
  
 indexOfCurrentMinimum = j;  
  
 }  
  
 }  
  
 smallest = indexOfCurrentMinimum;  
  
 if(array[smallest] < array[i]) {  
 *swap*(array, i, smallest);  
 numberOfSwaps++;  
 }  
  
  
 }  
  
// return numberOfComparisons;  
 return numberOfSwaps;  
  
 }  
  
 private static void swap(Integer[] array, int startIndex, int smallestIndex){  
  
 int temp = array[startIndex];  
 array[startIndex] = array[smallestIndex];  
 array[smallestIndex] = temp;  
  
 }  
  
}

**ShellSortArray**

public class ShellSortArray {  
  
 public static int shellSort(Integer[] array){  
  
 int arrayLength = array.length;  
 int start = 0;  
 int end = arrayLength -1;  
 int interval = ((end - start) + 1)/2;  
 int numberOfComparisons = 0;  
 int numberOfSwaps = 0;  
  
 while(interval > 0) {  
  
 for (int i = start; i < (start + interval); i++) {  
  
 int index;  
  
 for (int j = start + interval; j <= end; j += interval) {  
  
 int nextToInsert = array[j];  
  
 index = j - interval;  
  
 numberOfComparisons++;  
  
 while ((index >= start) && (nextToInsert < array[index])){  
  
 numberOfSwaps++;  
  
 array[index + interval] = array[index];  
  
 index = index - interval;  
  
 }  
  
 array[index + interval] = nextToInsert;  
 }  
  
 }  
  
 interval = interval/2;  
  
 }  
  
// return numberOfComparisons;  
 return numberOfSwaps;  
  
 }  
  
}