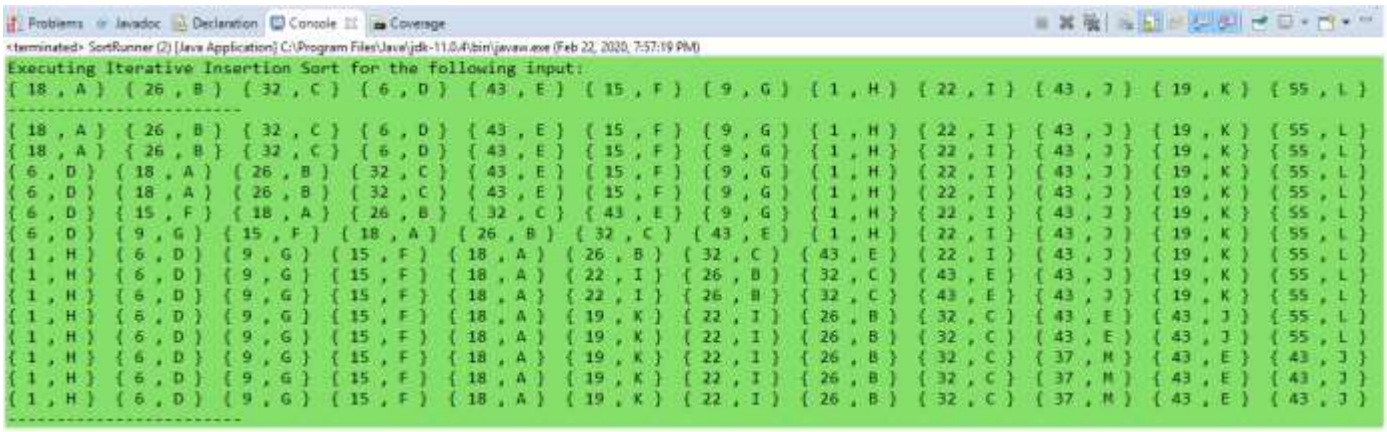
Assignment-01
CSE 373
Design and Analysis of Algorithms
Section-10

Spring2020
North South University

Submitted To: Shaikh Shawon Arefin Shimon (SAS3)

Name	: Md. Rifat Hossain
Student ID	: 1621373
Email Address	: Rifat.hossain05@northsouth.edu
Submission Date	: 22 Feb 2020



IterativeInsertionSort.java

```
1 package com.nsu.cse373.spring2020.ID1621373;
2
3 public class IterativeInsertionSort {
4 public static <E extends Comparable<E>> void
iterativeInsertionSort(E[]
inputArray){
5 System.out.println("Executing Iterative Insertion Sort for the
following
input:");
6 SortHelper.print(inputArray,inputArray.length);
7 System.out.println("------------------------");
12 sortInternal(inputArray, inputArray.length);
13 System.out.println("------------------------");
14 }
15
21 private static <E extends Comparable<E>> void sortInternal(E[]
inputArray
22 , int size){
23
24 size = inputArray.length;
25 for (int i = 1; i < size; ++i) {
26 E key = inputArray[i];
27 int j = i - 1;
28
29
30 while (j >= 0 && inputArray[j].compareTo(key)>0) {
31 inputArray[j + 1] = inputArray[j];
32 j = j - 1;
33 }
34 inputArray[j + 1] = key;
```

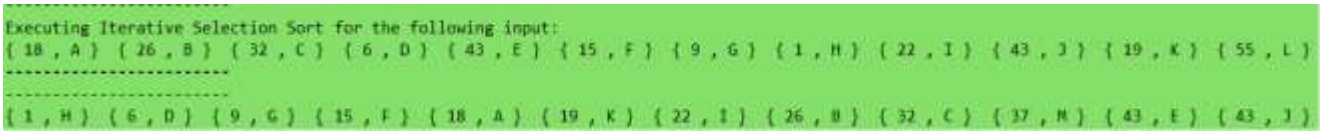
```
35 SortHelper.print(inputArray,size);
36 }
44 }
45
IterativeInsertionSort.java
46 }
```

```
Executing HeapSort for the following input:
{ 18 , A } { 26 , B } { 32 , C } { 6 , D } { 43 , E } { 15 , F } { 9 , G } { 1 , H } { 22 , I } { 43 , J } { 19 , K } { 55 , L }
----------------------
{ 1 , H } { 6 , D } { 9 , G } { 15 , F } { 18 , A } { 19 , K } { 22 , I } { 26 , B } { 32 , C } { 37 , M } { 43 , L } { 43 , J }
----------------------
```

HeapSort.java

```
1 package com.nsu.cse373.spring2020.ID1621373;
2 public class HeapSort {
3 public static <E extends Comparable<E>> void heapSort(E[]
inputArray){
4 System.out.println("Executing HeapSort for the following input:");
5 SortHelper.print(inputArray,inputArray.length);
6 System.out.println("------------------------");
7 sort(inputArray);
8 SortHelper.print(inputArray,inputArray.length);
9 System.out.println("------------------------");
10 }
11 private static<E extends Comparable<E>> void
12 sort (E []inputArray) {
13 int len=inputArray.length;
14 for(int i=len/2 -1; i>= 0; i--)
15 heapify(inputArray,len,i);
16 for(int i=len-1; i>=0; i--) {
17 E temp= inputArray[0];
18 inputArray[0]=inputArray[i];
19 inputArray[i]=temp;
20 heapify(inputArray,i,0);
21 }
22 }
23 public static <E extends Comparable<E>> void
24 heapify(E []inputArray, int n, int i) {
25 int largest=i;
26 int l=2*i+1;
27 int r=2*i+2;
28 if(l<n && inputArray[l].compareTo(inputArray[largest])>0) {
29 largest=l;
30 }
31 if(r<n && inputArray[r].compareTo(inputArray[largest])>0) {
32 largest=r;
```

```
33 }
34 if(largest != i) {
35 E swap=inputArray[i];
36 inputArray[i]=inputArray[largest];
37 inputArray[largest]=swap;
38
39 heapify(inputArray,n, largest);
40 }
41 }
42 }
43
```



IterativeSelectionSort.java

```
1 package com.nsu.cse373.spring2020.ID1621373;
2
3 public class IterativeSelectionSort {
4 public static <E extends Comparable<E>> void
iterativeSelectionnSort(E[]
inputArray){
5 System.out.println("Executing Iterative Selection Sort for the
following
input:");
6 SortHelper.print(inputArray,inputArray.length);
7 System.out.println("------------------------");
8 /*
9 Call your internal sorting method here
10 sortInternal(inputArray, inputArray.length);
11 */
12 sortInternal(inputArray,inputArray.length);
13 System.out.println("------------------------");
14 SortHelper.print(inputArray,inputArray.length);
15 }
16
17 /*
18 * You are allowed to change the function signature to whatever you
want
19 * but it must take generic type input to be able to sort any array
20 * containing data that can be compared. Look at BubbleSort class
for
21 */
```

```
22 private static <E extends Comparable<E>> void sortInternal(E[]
inputArray
23 , int size){
24
25 for(int i=0; i<size-1; i++) {
26 int min_indx=i;
27 for(int j=i+1; j<size; j++) {
28 if( inputArray[min_indx].compareTo(inputArray[j]) > 0 )
29 min_indx=j;
30
31 }
32 E temp=inputArray[min_indx];
33 inputArray[min_indx]=inputArray[i];
34 inputArray[i]=temp;
35
36 }
37
38 }
39 }
```

```
Executing MergeSort for the following input:
{ 18 , A } { 26 , B } { 32 , C } { 6 , D } { 43 , E } { 15 , F } { 9 , G } { 1 , H } { 22 , I } { 43 , J } { 19 , K } { 55 , L }
----------------------
{ 1 , H } { 6 , D } { 9 , G } { 15 , F } { 18 , A } { 19 , K } { 22 , I } { 26 , B } { 32 , C } { 37 , M } { 43 , E } { 43 , J }
----------------------
```

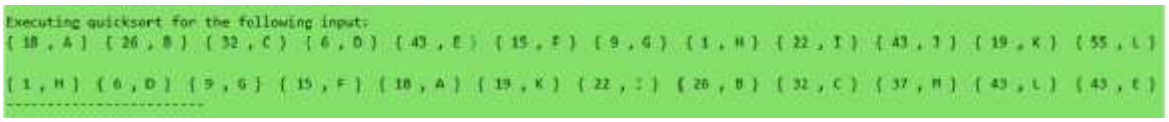
MergeSort.java

```
1 package com.nsu.cse373.spring2020.ID1621373;
2
3 public class MergeSort {
4
5 private static <E extends Comparable <E>> void
6 merge(E arr[], int left, int middle, int right){
7 int sizeOne= middle-left+1;
8 int sizeTwo= right- middle;
9 E[] tempOne= (E []) new Comparable[sizeOne];
10 E[] tempTwo= (E[]) new Comparable[sizeTwo];
11
12 for(int i=0; i< sizeOne; ++i) {
13 tempOne[i]=arr[left+i];
14 }
15 for(int j=0; j<sizeTwo; ++j) {
16 tempTwo[j]= arr[middle+1+j];
17 }
18 int i=0;
19 int j=0;
20 int k=left;
```

```
21 while(i < sizeOne && j < sizeTwo) {
22 if((tempOne[i].compareTo(tempTwo[j])< 0) ||
23 (tempOne[i].compareTo(tempTwo[j])==0)) {
24 arr[k]=tempOne[i];
25 i++;
26 }
27
28 else {
29 arr[k]= tempTwo[j];
30 j++;
31 }
32 k++;
33 }
34 while(i <sizeOne) {
35 arr[k]=tempOne[i];
36 i++;
37 k++;
38 }
39 while(j < sizeTwo) {
40 arr[k]= tempTwo[j];
41 j++;
42 k++;
43 }
44
45 }
46 private static <E extends Comparable <E>> void
47 sort(E arr[], int left, int right) {
```

```
48 if(left < right) {
49 int middle= (left + right)/2 ;
50
51 sort(arr, left , middle);
52 sort(arr, middle+1, right);
53
54
55 merge(arr, left, middle, right);
56 }
57 }
58
59 public static <E extends Comparable<E>> void mergeSort(E[]
inputArray){
60 System.out.println("Executing MergeSort for the following input:");
61 SortHelper.print(inputArray,inputArray.length);
62 System.out.println("------------------------");
63 sort(inputArray, 0, inputArray.length-1);
```

```
64 SortHelper.print(inputArray,inputArray.length);
65 System.out.println("------------------------");
66 }
67
68
69 }
```



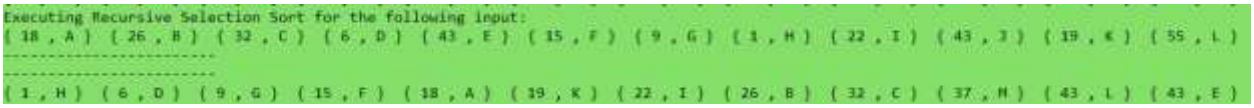
QuickSort.java

```
1 package com.nsu.cse373.spring2020.ID1621373;
2 public class QuickSort {
3 public static <E extends Comparable<E>> void quickSort(E[]
inputArray){
4 System.out.println("Executing quick for the following input:");
5 SortHelper.print(inputArray,inputArray.length);
6 System.out.println("------------------------");
7 sort(inputArray,0, inputArray.length-1);
8 SortHelper.print(inputArray,inputArray.length);
9 System.out.println("------------------------");
10 }
11 private static <E extends Comparable<E> > int
12 partition(E arr[],int low, int high) {
13 E pivot= arr[high];
14 int i= (low-1);
15 for(int j=low; j<high; j++) {
16 if(arr[j].compareTo(pivot)<0) {
17 ++i;
18 E temp= arr[i];
19 arr[i]=arr[j];
20 arr[j]=temp;
21 }
22 }
23 E temp=arr[i+1];
24 arr[i+1]=arr[high];
25 arr[high]=temp;
26 return (i+1);
27 }
28
29 private static <E extends Comparable<E> > void sort(
30 E arr[], int low , int high) {
31 if(low < high) {
32 int pi= partition (arr, low, high);
```

```
33 sort(arr, low, pi-1);
34 sort(arr, pi+1, high);
35
36 }
37 }
38 }
39
```

RecursiveInsertionSort.java

```
1 package com.nsu.cse373.spring2020.ID1621373;
2
3 public class RecursiveInsertionSort {
4 public static <E extends Comparable<E>> void
recursiveInsertionSort(E[]
inputArray){
5 System.out.println("Executing Recursive Insertion Sort for the
following
input:");
6 SortHelper.print(inputArray,inputArray.length);
7 System.out.println("------------------------");
8 sortInternal(inputArray,inputArray.length);
9 System.out.println("------------------------");
10 SortHelper.print(inputArray,inputArray.length);
11
12 }
13
14
15 private static <E extends Comparable<E>> void sortInternal(E[]
inputArray,
int size){
16 if(size <= 1)
17 return;
18 sortInternal(inputArray,size-1);
19 E last=inputArray[size-1];
20 int j=size-2;
21 while(j>=0 && inputArray[j].compareTo(last) > 0 ) {
22 inputArray[j+1]=inputArray[j];
23 j--;
24 }
25 inputArray[j+1]=last;
26
27 }
28 }
```



RecursiveSelectionSort.java

```
1 package com.nsu.cse373.spring2020.ID1621373;
2
3 public class RecursiveSelectionSort {
4
5 public static <E extends Comparable<E>> void
recursiveSelectionSort(E[]
inputArray){
6 System.out.println("Executing Recursive Selection Sort for the
following
input:");
7 SortHelper.print(inputArray,inputArray.length);
8 System.out.println("------------------------");
9
10 sortInternal(inputArray,inputArray.length,0);
11 System.out.println("------------------------");
12 SortHelper.print(inputArray,inputArray.length);
13 }
14
15 public static <E extends Comparable<E>> int minIndex(E
inputArray[], int i,
int j)
16 {
17 if (i == j)
18 return i;
19
20 // Find minimum of remaining elements
21 int k = minIndex(inputArray, i + 1, j);
22
23 // Returning minimum of current and remaining.
24
25 if(inputArray[i].compareTo(inputArray[k]) < 0)
26 return i;
27 else
28 return k;
29
30 }
31
32
33 public static <E extends Comparable<E>> void sortInternal(E[]
inputArray, int
size, int index){
```

```
34
35 if(index==size)
36 return;
37 int k= minIndex(inputArray,index,size-1 );
38 if (k != index){
39 // swap
40 E temp = inputArray[k];
41 inputArray[k] = inputArray[index];
42 inputArray[index] = temp;
43 }
```

```
44
45 // Recursively calling selection sort function
46 sortInternal(inputArray, size, index+1);
47
48
49
50 }
51 }
```