**TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES**

**QUEZON CITY**

**COLLEGE OF INFORMATION TECHNOLOGY EDUCATION (CITE)**

**CS 201 - Data Structures and Algorithms**

|  |  |
| --- | --- |
| **Name: Aristotle Buenaventura** | **Date: September 21, 2021** |
| **Program/Section: IT21S1** | **Instructor: Ms. Rosmina Joy M. Cabauatan** |
| **Assignment 3. | Project Conceptualization and Development** | |

**SOURCE CODE**

**Array\_Element Class**

package \_3\_assignment\_3;

import javax.swing.JOptionPane;

public class Array\_Element extends ArrayList\_Element{

public static void arrayElement() {

byte size=(byte) Integer.parseInt(JOptionPane.showInputDialog(null, "Enter the Number of Arrays: ", "Array Elements",JOptionPane.INFORMATION\_MESSAGE));

// Declaration and instantiation of Array

int array\_Element[] = new int[size];

// for loop for accepting values

for(int i=0; i < array\_Element.length; i++) {

array\_Element[i] = Integer.parseInt(JOptionPane.showInputDialog("Enter ID number: "));

}

System.out.print("The list of Array Elements: " );

for(int i=0;i<size;i++) {

System.out.print(array\_Element[i]);

if(i<size-1) {

System.out.print(", ");

}

}

System.out.println();

arrayListElement(array\_Element);

}

}

**ArrayList\_Element Class**

package \_3\_assignment\_3;

import java.util.ArrayList;

public class ArrayList\_Element extends Heap\_Sort{

public static void arrayListElement(int[] arrayElement) {

// Declaration and Instantiation of ArrayList

ArrayList<Integer> arrayList\_Element =new ArrayList<Integer>(arrayElement.length);

for(int num: arrayElement) {

arrayList\_Element.add(num);

}

System.out.print("The list of ArrayList Elements: " );

for(int arrayListPrint: arrayList\_Element) {

System.out.print(arrayListPrint + " ");

}

Stack\_Element.stack(arrayList\_Element);

heap\_Sort(arrayList\_Element);

}

}

**Stack\_Element Class**

package \_3\_assignment\_3;

import java.util.ArrayList;

import java.util.Stack;

public class Stack\_Element {

public static void stack(ArrayList<Integer> arrayList\_Element) {

Stack <Integer> stack = new Stack<>();

System.out.println("\n\nPush elements from ArrayList to Stack.\n");

for(int arrayListPrint: arrayList\_Element) {

stack.push(arrayListPrint);

System.out.println("\t" + arrayListPrint + " has been added to the stack.");

}

}

}

**Heap\_Sort Class**

package \_3\_assignment\_3;

import java.util.ArrayList;

public class Heap\_Sort {

public static void heap\_Sort(ArrayList<Integer> arrayList\_Element) {

ArrayList<Integer> arr = arrayList\_Element;

int i;

heapSort(arr, arr.size());

System.out.println();

System.out.print("Sorted using Heap Sort Algorithm: ");

for(i =0; i<arr.size();i ++) {

System.out.print(arr.get(i));

if(i<arr.size()-1) {

System.out.print(", ");

}

}

}

static void heapSort(ArrayList<Integer> arr, int size) {

int i;

for (i = size / 2 - 1; i >= 0; i--)

heapify(arr, size, i);

for (i=size-1; i>=0; i--)

{

int temp = arr.get(0);

arr.set(0, arr.get(i));

arr.set(i, temp);

heapify(arr, i, 0);

}

}

private static void heapify(ArrayList<Integer> arr, int size, int i) {

int largest = i;

int left = 2\*i + 1;

int right = 2\*i + 2;

if (left < size && arr.get(left) >arr.get(largest))

largest = left;

if (right < size && arr.get(right) > arr.get(largest))

largest = right;

if (largest != i)

{

int temp = arr.get(i);

arr.set(i, arr.get(largest));

arr.set(largest, temp);

heapify(arr, size, largest);

}

}

}

**Runner Class**

package \_3\_assignment\_3;

import javax.swing.JOptionPane;

public class Runner extends Array\_Element {

public static void main(String args[]) {

try {

arrayElement();

} catch(Exception e) {

JOptionPane.showMessageDialog(null, "Oops, something went wrong. Please try again.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

**JUSTIFICATION OF WHY WE SELECT HEAP SORT ALGORITHM:**

The group decided to use a heap sort algorithm since it is well-known for its efficiency in sorting a large number of items to be sorted. Furthermore, due to its simplicity and consistency, it performs excellently in the best, average, and worst cases, and it is considered as an improved version of the selection sort with the use of the binary search tree. Furthermore, heap sort only takes n logn or logarithmic time compared to other sorting algorithm that take n^2 or quadratic time.

A Heap Sort divides its input into a sorted region, and it iteratively shrinks the unsorted region by extracting the largest element from it and inserting it into the sorted region.

**OUTPUT**

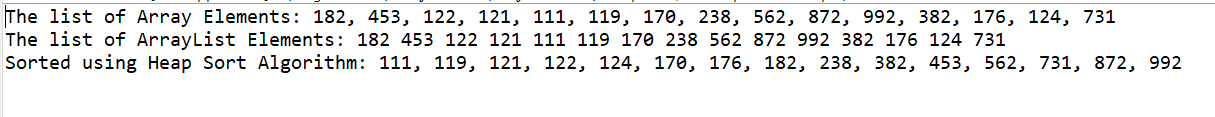


Figure 1. Output

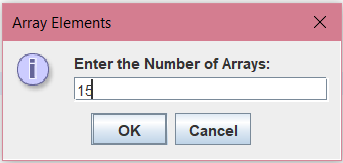
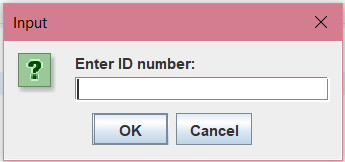


Figure 2. Enter size of the Array

Figure 3. To populate the Array

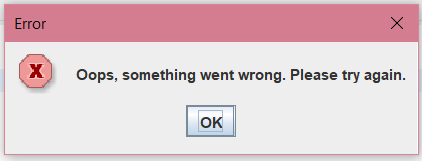


Figure 4. Error Output

Honor Pledge: “I accept responsibility for my role in ensuring the integrity of the work submitted by the group in which I participated.”