**TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES**

**QUEZON CITY**

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

**CS 201 - Data Structures and Algorithms**

|  |  |
| --- | --- |
| **Name: Aristotle Buenaventura** | **Date: September 18, 2021** |
| **Program/Section: IT21S1** | **Instructor: Ms. Rosmina Joy M. Cabauatan** |
| **Assessment Task: Exercise 3 - Sort** | |

**Code**

**Array\_Element Class**

package exercise3\_Buenaventura;

import javax.swing.JOptionPane;

public class Array\_Element extends ArrayList\_Element{

public static void main(String args[]) {

// Getting the Size of the Array

int size= 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];

System.out.println("The number of Elements: " + size);

// for loop for accepting values

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

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

}

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

// Displaying the initial list of Array

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

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

if(i<size-1) {

System.out.print(", ");

}

}

System.out.println();

//calling the method

arrayListElement(array\_Element);

}

}

**ArrayList\_Element Class**

package exercise3\_Buenaventura;

import java.util.ArrayList;

import javax.swing.JOptionPane;

public class ArrayList\_Element {

public static void arrayListElement(int[] arrayElement) {

// Declaration and Instantiation of ArrayList

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

// Passing the elements of Array to ArrayList

for(int num: arrayElement) {

arrayList\_Element.add(num);

}

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

// Displaying the ArrayList

for(int i =0; i<arrayList\_Element.size();i++) {

System.out.print(arrayList\_Element.get(i));

if(i<arrayList\_Element.size()-1) {

System.out.print(", ");

}

}

// Calling the method with ArrayList as the argument

Merge.merge\_sort(arrayList\_Element);

Quick.quickSort(arrayList\_Element);

Bucket.bucketSort(arrayList\_Element);

Radix.bucketSort(arrayList\_Element);

}

}

**Sorting\_Algorithms Class**

package exercise3\_Buenaventura;

import java.util.ArrayList;

import java.util.Arrays;

// Merge Sort

class Merge{

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

// Declaration and Instantiation of ArrayList

ArrayList<Integer> arrays = arrayList\_Element;

// calling the method

Merge.sort\_part(arrays, 0, arrays.size()-1);

System.out.println();

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

// calling the method

Print.printable(arrays);

}

private static void sort\_part(ArrayList<Integer> arrays, int lower, int upper) {

if (lower == upper){

return;

}else{

// get mid point for division of array

int middle = (lower + upper)/2;

sort\_part(arrays, lower, middle);

sort\_part(arrays, middle+1, upper);

merge\_part(arrays, lower, middle, upper);

}

}

private static void merge\_part(ArrayList<Integer> intArr, int lower, int middle, int upper) {

/\*\* Create two temp arrays pertaining to two halves that

are being merged and add elements to them \*/

int subArrayOneLength = middle - lower + 1;

int subArrayTwoLength = upper - middle;

int[] temp1 = new int[subArrayOneLength];

int[] temp2 = new int[subArrayTwoLength];

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

temp1[i] = intArr.get(lower + i);

}

for(int j = 0; j < subArrayTwoLength; j++){

temp2[j] = intArr.get(middle + 1 + j);

}

int i =0;

int j = 0;

// merging process, merge two temp arrays

while((i < subArrayOneLength) && (j < subArrayTwoLength)){

if(temp1[i] < temp2[j]){

intArr.set(lower, temp1[i++]);

}else{

intArr.set(lower, temp2[j++]);

}

lower++;

}

// If there are more elements

while(i < subArrayOneLength){

intArr.set(lower++, temp1[i++]);

}

while(j < subArrayTwoLength){

intArr.set(lower, temp2[j++]);

}

}

}

// Quick Sort

class Quick{

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

// Declaration and Instantiation of ArrayList

ArrayList<Integer> arrays = arrayList\_Element;

// Calling the method

quickSorted(arrays, 0, arrays.size()-1);

System.out.println();

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

// Calling the method

Print.printable(arrays);

}

private static void quickSorted(ArrayList<Integer> arrays, int low, int high) {

// recursive

int loc;

if(low<high)

{

loc = partition\_part(arrays, low, high);

quickSorted(arrays, low, loc-1);

quickSorted(arrays, loc+1, high);

}

}

private static int partition\_part(ArrayList<Integer> arrays, int beg, int end) {

// partioning of the arrayList

int left, right, temp, location, pivot;

location = left = beg;

right = end;

pivot = 0;

while(pivot != 1){

while((arrays.get(location) <= arrays.get(right)) && (location!=right))

right--;

if(location==right)

pivot =1;

else if(arrays.get(location)>arrays.get(right))

{

temp = arrays.get(location);

//arrays[location] = arrays[right];

arrays.set(location, arrays.get(right));

//arrays[right] = temp;

arrays.set(right, temp);

location = right;

}

if(pivot!=1)

{

while((arrays.get(location) >= arrays.get(right)) && (location!=left))

left++;

if(location==left)

pivot =1;

else if(arrays.get(location) <arrays.get(left))

{

temp = arrays.get(location);

//arrays[location] = arrays[left];

arrays.set(location, arrays.get(left));

//arrays[left] = temp;

arrays.set(left, temp);

location = left;

}

}

}

return location;

}

}

// Bucket Sort

class Bucket{

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

// Declaration and Instantiation of ArrayList

ArrayList<Integer> arrays = arrayList\_Element;

int max\_value=maxValue(arrays);

// Calling the method

bucket\_sort(arrays,max\_value);

System.out.println();

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

// Calling the method

Print.printable(arrays);

}

public static int[] bucket\_sort(ArrayList<Integer> arrays, int max\_value)

{

int[] bucket = new int[max\_value + 1];

int[] sorted\_arr = new int[arrays.size()];

for (int i= 0; i <arrays.size(); i++)

bucket[arrays.get(i)]++;

int pos = 0;

for (int i = 0; i < bucket.length; i++)

for (int j = 0; j < bucket[i]; j++)

sorted\_arr[pos++] = i;

return sorted\_arr;

}

static int maxValue(ArrayList<Integer> arrays)

{

int max\_value = 0;

for (int i = 0; i < arrays.size(); i++)

if (arrays.get(i) > max\_value)

max\_value = arrays.get(i);

return max\_value;

}

}

// Radix Sort

class Radix {

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

// Declaration and Instantiation of ArrayList

ArrayList<Integer> arrays = arrayList\_Element;

radixsort(arrays, arrays.size());

System.out.println();

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

Print.printable(arrays);

}

private static void radixsort(ArrayList<Integer> arrays, int n) {

// Find the maximum number to know number of digits

int m = getMax(arrays, n);

// Do counting sort for every digit. Note that

// instead of passing digit number, exp is passed.

// exp is 10^i where i is current digit number

for (int exp = 1; m / exp > 0; exp \*= 10)

countSort\_part(arrays, n, exp);

}

private static void countSort\_part(ArrayList<Integer> arrays, int n, int exp) {

int output[] = new int[n]; // output array

int i;

int count[] = new int[10];

Arrays.fill(count, 0);

// Store count of occurrences in count[]

for (i = 0; i < n; i++)

count[(arrays.get(i) / exp) % 10]++;

// Change count[i] so that count[i] now contains

// actual position of this digit in output[]

for (i = 1; i < 10; i++)

count[i] += count[i - 1];

// Build the output array

for (i = n - 1; i >= 0; i--) {

output[count[(arrays.get(i) / exp) % 10] - 1] = arrays.get(i);

count[(arrays.get(i) / exp) % 10]--;

}

// Copy the output array to arr[], so that arr[] now

// contains sorted numbers according to current digit

for (i = 0; i < n; i++) {

arrays.set(i, output[i]);

}

}

private static int getMax(ArrayList<Integer> arrays, int n) {

int mx = arrays.get(0);

for (int i = 1; i < n; i++)

if (arrays.get(i) > mx)

mx = arrays.get(i);

return mx;

}

}

class Print{

public static void printable(ArrayList<Integer> arrays) {

// Printing the results

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

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

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

System.out.print(", ");

}

}

}

} // End of the Program

**Output**

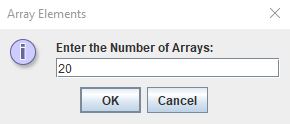


Figure 1. Enter the Size

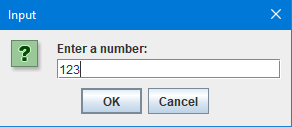


Figure 2. Enter the Elements

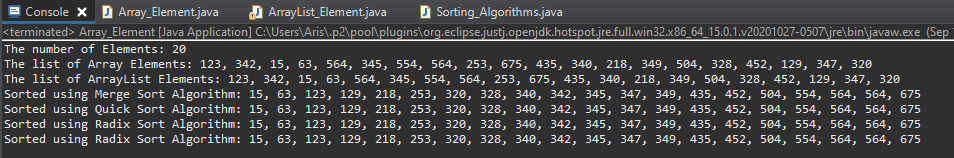


Figure 3. The Final Output