**Q1**

package sorting;

import java.util.Scanner;

public class Assign1InsertionSort

{

// insertion Sort

public static void insertionSort(String[] arr)

{

for (int i = 1; i < arr.length; i++)

{

String key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j].compareTo(key) > 0)

{

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

public static void display(String arr[])

{

for(String i:arr)

{

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

}

}

public static void main(String[] args)

{

// creating input object

Scanner input = new Scanner(System.in);

// taking number of element inserting in array

System.out.print("Enter the size of array strings: ");

int n = input.nextInt();

input.nextLine(); // Consume the newline character

String[] strings = new String[n];

System.out.println("Enter the strings:");

// taking string inputs

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

{

strings[i] = input.nextLine();

}

// sorting by insertion sort

insertionSort(strings);

//

System.out.println("\nSorted Strings:");

display(strings);

input.close();

}

}

**Output**

**Enter the size of array strings: 5**

**Enter the strings:**

**AS**

**AR**

**AO**

**AC**

**AM**

**Sorted Strings:**

**AC AM AO AR AS**

**Q2**

package sorting;

import java.util.Random;

public class SortingPerformanceAnalyzer

{

// Generate random number using Random

public static int[] generateRandomArray(int size)

{

int[] arr = new int[size];

Random random = new Random();

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

{

arr[i] = random.nextInt(1000); // Generate random numbers between 0 to 999

}

return arr;

}

//analyze the selection sort

public static void selectionSort(int[] arr)

{

for(int i=0;i<arr.length-1;i++)

{

int minIndex=i;

for (int j=i+1;j<arr.length;j++)

{

if(arr[j]<arr[minIndex])

{

minIndex=j;

}

}

int temp=arr[minIndex];

arr[minIndex]=arr[i];

arr[i]=temp;

}

}

// analyze the bubble sort

public static void bubbleSort(int[] arr)

{

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

{

boolean loop=false;

for(int j=0;j<arr.length-i-1; j++)

{

if(arr[j]>arr[j+1])

{

int temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

loop=true;

}

}

// braking loop after iteration

if(!loop)

{

System.out.println("\n\nsorting complete after "+i+" loop iteration \n ");

break;

}

}

}

// analyze the insertion sort performance

public static void insertionSort(int[] arr)

{

for(int i=1;i<arr.length;i++)

{

int temp=arr[i];

int j=i-1;

while(j>=0 && arr[j]>temp)

{

arr[j+1]=arr[j];

j--;

}

arr[j+1]=temp;

}

}

public static void main(String[] args)

{

// genrate random 1000 integer numbers

int[] arr = generateRandomArray(1000);

// checking selection sort time

long startTimeSelectionSort = System.nanoTime();

selectionSort(arr.clone());

long selectionSortTime = System.nanoTime() - startTimeSelectionSort;

// checking Bubble sort time

long startTimeBubbleSort = System.nanoTime();

bubbleSort(arr.clone());

long bubbleSortTime = System.nanoTime() - startTimeBubbleSort;

//checking Insertion sort time

long startTimeInsertionSort = System.nanoTime();

insertionSort(arr.clone());

long insertionSortTime = System.nanoTime() - startTimeInsertionSort;

// printing all sorting times

System.out.println("Selection Sort Time: " + selectionSortTime + " ns");

System.out.println("Bubble Sort Time: " + bubbleSortTime + " ns");

System.out.println("Insertion Sort Time: " + insertionSortTime + " ns");

}

}

**OUTPUT**

**sorting complete after 956 loop iteration**

**Selection Sort Time: 4854400 ns**

**Bubble Sort Time: 21223400 ns**

**Insertion Sort Time: 6687500 ns**