**SortingAlgorithms.java**

class SortingAlgorithms{

static void bubbleSort(int[] size)

{

int flag =0;

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

flag=0;

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

if(size[j]>size[j+1]) {

flag=1;

size[j] = size[j]+size[j+1];

size[j+1] = size[j]-size[j+1];

size[j] = size[j]-size[j+1];

}

//loopCount++;

}

if(flag==0)

break;

}

}

static void selectionSort(int[] size)

{

int min=0;

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

min=i;

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

if(size[min]>size[j])

min=j;

if(i!=min) {

size[i] = size[i]+size[min];

size[min] = size[i]-size[min];

size[i] = size[i]-size[min];

}

}

}

static void quickSortMain(int[] size)

{

quickSort(0,size.length-1,size);

}

public static void quickSort(int start, int end, int[] size) {

if(start<end){

int pIndex = partition(start,end,size);

quickSort(start,pIndex-1,size);

quickSort(pIndex+1,end,size);

}

}

public static int partition(int start, int end,int[] size){

int pivot =size[end];

int pIndex=start, temp=0;

for(int i=start;i<=end;i++) {

if(size[i]<=pivot){

temp = size[i];

size[i] = size[pIndex];

size[pIndex] = temp;

pIndex++;

}

}

return pIndex-1;

}

public static void insertionSort(int[] arr){

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

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

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

insert(arr,j,i);

}

}

}

private static void insert(int[] arr, int j, int i){

int temp = arr[i];

for(int k=i-1; k>=j; k--)

arr[k+1] = arr[k];

arr[j] = temp;

}

}

**Calculator.java**

class Calculator{

static int[] generateRandomArray(int[] arr)

{

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

{

int n = (int)(Math.random()\*100);

arr[i] = n;

}

return arr;

}

static int[] generateSortedArray(int[] arr)

{

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

arr[i] = i+1;

return arr;

}

static int[] generateReverseArray(int[] arr)

{

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

arr[j] = i;

return arr;

}

}

**Test.java**

import java.util.\*;

class Test{

public static void main(String[] args)

{

long startTime;

long endTime ;

long totalTime;

int size=0;

String main=("\*\*SORTING LOGORITHM BENCHMARK APP\*\*");

int count =main.length();

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

{

System.out.print("\*");

}

System.out.println();

System.out.println(main);

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

{

System.out.print("\*");

}

System.out.println();

System.out.println("\nEnter the Array size");

while(true){

try{

Scanner sc = new Scanner(System.in);

size = sc.nextInt();

break;

}catch(InputMismatchException misEx){

System.out.println("Invalid INput!!\n Please Enter a Number");

continue;

}

}

//end of array size

int[] arr = new int[size];

while(true)

{

int num;

try{

System.out.println("Choose Complexity");

System.out.println("\n1) Best Case");

System.out.println("\n2) Random Case");

System.out.println("\n3) Worst Case");

Scanner sc = new Scanner(System.in);

num = sc.nextInt();

}catch(InputMismatchException misEx){

System.out.println("Please Enter a valid Input -- "+misEx);

continue;

}

//System.out.println();

switch(num)

{

case 1 : startTime = System.currentTimeMillis();

arr = Calculator.generateRandomArray(arr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

String c1 = ("Sorted Array of " + size + " created for Best case ");

int count8 =c1.length();

for(int i=0;i<count8;i++)

{

System.out.print("\*");

}

System.out.println();

System.out.println(c1);

for(int i=0;i<count8;i++)

{

System.out.print("\*");

}

break;

case 2: startTime = System.currentTimeMillis();

arr = Calculator.generateSortedArray(arr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

String c2 =("Random array of " + size + " created for average case");

int count1 =c2.length();

for(int i=0;i<count1;i++){

System.out.print("\*");

}

System.out.println();

System.out.println(c2);

for(int i=0;i<count1;i++){

System.out.print("\*");

}

break;

case 3: startTime = System.currentTimeMillis();

arr = Calculator.generateReverseArray(arr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

String c3 =("Array of " + size + " created for Worst case");

int count2 =c3.length();

for(int i=0;i<count2;i++){

System.out.print("\*");

}

System.out.println();

System.out.println(c3);

for(int i=0;i<count2;i++){

System.out.print("\*");

}

break;

default:System.out.println("Invalid Input");

continue;

}break;

}

//End of Complexity

int[] cloneArr = arr.clone();

while(true)

{

int option;

try{

System.out.println("\nChoose the algorithm");

System.out.println("1) BubbleSort");

System.out.println("2) SelectionSort");

System.out.println("3) QuickSort");

System.out.println("4) Insertion Sort");

System.out.println("5) Benchmark All");

System.out.println("6) Start Over");

System.out.println("7) Exit");

Scanner sc = new Scanner(System.in);

option = sc.nextInt();

}catch(InputMismatchException inMisEx){

System.out.println("Invalid input. Please enter a number of the menu item --"+inMisEx);

continue;

}

switch(option)

{

case 1 :

startTime = System.currentTimeMillis();

SortingAlgorithms.bubbleSort(cloneArr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

System.out.println("\n\nBubbleSort completed in "+totalTime +" milliseconds");

break;

case 2: startTime = System.currentTimeMillis();

SortingAlgorithms.selectionSort(cloneArr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

System.out.println("\nSelectionSort completed in "+totalTime +" milliseconds");

break;

case 3: startTime = System.currentTimeMillis();

SortingAlgorithms.quickSortMain(cloneArr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

System.out.println("\nQuickSort completed in "+totalTime +" milliseconds");

break;

case 4: startTime = System.currentTimeMillis();

SortingAlgorithms.insertionSort(cloneArr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

System.out.println("\nInsertion Sort completed in "+totalTime +" milliseconds");

break;

case 5 :

startTime = System.currentTimeMillis();

SortingAlgorithms.bubbleSort(cloneArr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

System.out.println("\nBubbleSort completed in "+totalTime +" milliseconds");

startTime = System.currentTimeMillis();

SortingAlgorithms.selectionSort(cloneArr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

System.out.println("\nSelectionSort completed in "+totalTime +" milliseconds");

startTime = System.currentTimeMillis();

SortingAlgorithms.quickSortMain(cloneArr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

System.out.println("\nQuickSort completed in "+totalTime +" milliseconds");

startTime = System.currentTimeMillis();

SortingAlgorithms.insertionSort(cloneArr);

endTime = System.currentTimeMillis();

totalTime = endTime - startTime;

System.out.println("\nInsertion Sort completed in "+totalTime +" milliseconds");

break;

case 6 :

main(args);

break;

case 7 :

System.exit(0);

break;

default:

System.out.println("Invalid Input");

continue;

}

}

}

}

