import java.util.Random;

public class Sorting {

public void selectionSort(int[] array)

{

int length = array.length;

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

{

int min = i;

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

if (array[j] < array[min])

min = j;

int temp = array[min];

array[min] = array[i];

array[i] = temp;

}

}

public void insertionSort(int[] array)

{

int length = array.length;

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

int marker = array[i];

int j = i - 1;

while (j >= 0 && array[j] > marker) {

array[j + 1] = array[j];

j = j - 1;

}

array[j + 1] = marker;

}

}

void merge(int[] array, int l, int m, int r)

{

int leftSize = m - l + 1, rightSize = r - m;

int[] left= new int[leftSize];

int[] right = new int[rightSize];

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

left[i] = array[l + i];

for (int j = 0; j < rightSize ; j++)

right[j] = array[m + 1 + j];

int i = 0, j = 0;

int k = l;

while (i < leftSize && j < rightSize) {

if (left[i] <= right[j]) {

array[k] = left[i];

i++;

}

else {

array[k] = right[j];

j++;

}

k++;

}

while (i < leftSize) {

array[k] = left[i];

i++;

k++;

}

while (j < rightSize) {

array[k] = right[j];

j++;

k++;

}

}

public void mergeSort(int arr[], int l, int r)

{

if (l < r) {

int m = l + (r - l) / 2;

mergeSort(arr, l, m);

mergeSort(arr, m + 1, r);

merge(arr, l, m, r);

}

}

public static void quickSwap(int[] arr, int i, int j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

public static int partition(int[] arr, int low, int high)

{

int pivot = arr[high];

int i = (low - 1);

for(int j = low; j <= high - 1; j++)

{

if (arr[j] < pivot)

{

i++;

quickSwap(arr, i, j);

}

}

quickSwap(arr, i + 1, high);

return (i + 1);

}

public void quickSort(int[] arr, int low, int high)

{

if (low < high)

{

int partition = partition(arr, low, high);

quickSort(arr, low, partition - 1);

quickSort(arr, partition + 1, high);

}

}

public void printArray(int[] array)

{

int n = array.length;

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

System.out.print(array[i]+" ");

System.out.println();

}

public void shuffleArray(int[] array) {

Random rand = new Random();

for (int i = 0; i < 1100; i++) {

int number = rand.nextInt(1000);

array[i] = number;

}

}

public void almostSortedArray(int[] array)

{

Random rand = new Random();

int random = rand.nextInt(1000);

for (int i = 0; i<1100; i++)

{

int number = i;

array[i] = number;

}

int temp = array[random];

array[random] = array[random-1];

array[random-1] = temp;

}

}

import java.util.Scanner;

public class Test {

public static void main(String args[]) {

Sorting test = new Sorting();

int[] randomSorted = new int[1100];

int[] almostSorted = new int[1100];

Scanner scan = new Scanner(System.in);

int ans = 0;

while (ans <= 4)

{

System.out.println("Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort");

ans = scan.nextInt();

if(ans == 1)

{

test.shuffleArray(randomSorted);

test.printArray(randomSorted);

System.out.println("Please enter the sorting algorithm you would like to use with the random array: \n" + "1. Selection Sort\n" + "2. Insertion Sort\n" + "3. Merge Sort\n" + "4. Quick Sort");

ans = scan.nextInt();

if(ans == 1) {

long startTime = System.nanoTime();

test.selectionSort(randomSorted);

long endTime = System.nanoTime();

long duration = (endTime-startTime);

System.out.println("Selection Sort took " + duration + " nanoseconds");

test.printArray(randomSorted);

}

else if(ans == 2) {

long startTime = System.nanoTime();

test.insertionSort(randomSorted);

long endTime = System.nanoTime();

long duration = (endTime-startTime);

System.out.println("Insertion Sort took " + duration + " nanoseconds");

test.printArray(randomSorted);

}

else if(ans == 3) {

long startTime = System.nanoTime();

test.mergeSort(randomSorted, 0, randomSorted.length-1);

long endTime = System.nanoTime();

long duration = (endTime-startTime);

System.out.println("Merge Sort took " + duration + " nanoseconds");

test.printArray(randomSorted);

}

else if(ans == 4) {

long startTime = System.nanoTime();

test.quickSort(randomSorted, 0, randomSorted.length-1);

long endTime = System.nanoTime();

long duration = (endTime-startTime);

System.out.println("Quick Sort took " + duration + " nanoseconds");

test.printArray(randomSorted);

}

else{

System.out.println("You are exiting the program, please run it again");

}

}

else if(ans == 2)

{

test.almostSortedArray(almostSorted);

test.printArray(almostSorted);

System.out.println("Please enter the sorting algorithm you would like to use with the almost sorted array: \n" + "1. Selection Sort\n" + "2. Insertion Sort\n" + "3. Merge Sort\n" + "4. Quick Sort");

ans = scan.nextInt();

if(ans == 1) {

long startTime = System.nanoTime();

test.selectionSort(almostSorted);

long endTime = System.nanoTime();

long duration = (endTime-startTime);

System.out.println("Selection Sort took " + duration + " nanoseconds");

test.printArray(almostSorted);

}

else if(ans == 2) {

long startTime = System.nanoTime();

test.insertionSort(almostSorted);

long endTime = System.nanoTime();

long duration = (endTime-startTime);

System.out.println("Insertion Sort took " + duration + " nanoseconds");

test.printArray(almostSorted);

}

else if(ans == 3) {

long startTime = System.nanoTime();

test.mergeSort(almostSorted, 0, almostSorted.length - 1);

long endTime = System.nanoTime();

long duration = (endTime-startTime);

System.out.println("Merge Sort took " + duration + " nanoseconds");

test.printArray(almostSorted);

}

else if(ans == 4) {

long startTime = System.nanoTime();

test.quickSort(almostSorted, 0, almostSorted.length - 1);

long endTime = System.nanoTime();

long duration = (endTime-startTime);

System.out.println("Quick Sort took " + duration + " nanoseconds");

test.printArray(almostSorted);

}

else {

System.out.println("You are exiting the program, please run it again");

}

}

else

{

System.out.println("You are exiting the program, please run it again");

}

}

}

}

NOTE: In the version I run, it has the array printed before and after it is sorted. To save paper, I took this out of the output for the in-person submission because it came out to 26 pages. If you would like to see what it looks like, please see the online version that was submitted on Blackboard.

Output:

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

1

Please enter the sorting algorithm you would like to use with the random array:

1. Selection Sort

2. Insertion Sort

3. Merge Sort

4. Quick Sort

1

Selection Sort took 5112998 nanoseconds

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

1

Please enter the sorting algorithm you would like to use with the random array:

1. Selection Sort

2. Insertion Sort

3. Merge Sort

4. Quick Sort

2

Insertion Sort took 3881021 nanoseconds

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

1

Please enter the sorting algorithm you would like to use with the random array:

1. Selection Sort

2. Insertion Sort

3. Merge Sort

4. Quick Sort

3

Merge Sort took 937469 nanoseconds

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

1

Please enter the sorting algorithm you would like to use with the random array:

1. Selection Sort

2. Insertion Sort

3. Merge Sort

4. Quick Sort

4

Quick Sort took 1510311 nanoseconds

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

2

Please enter the sorting algorithm you would like to use with the almost sorted array:

1. Selection Sort

2. Insertion Sort

3. Merge Sort

4. Quick Sort

1

Selection Sort took 1509906 nanoseconds

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

2

Please enter the sorting algorithm you would like to use with the almost sorted array:

1. Selection Sort

2. Insertion Sort

3. Merge Sort

4. Quick Sort

2

Insertion Sort took 12471 nanoseconds

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

2

Please enter the sorting algorithm you would like to use with the almost sorted array:

1. Selection Sort

2. Insertion Sort

3. Merge Sort

4. Quick Sort

3

Merge Sort took 261626 nanoseconds

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

2

Please enter the sorting algorithm you would like to use with the almost sorted array:

1. Selection Sort

2. Insertion Sort

3. Merge Sort

4. Quick Sort

4

Quick Sort took 7834801 nanoseconds

Please enter 1 to select the randomized array and 2 to choose the almost sorted array to sort

5

You are exiting the program, please run it again