Name of the Course : Advanced Data Structures and Algorithms in Java 9

Level : Easy

Tool Stack : Java9 and Junit5

Problem Statement : Provide a code solution to sort the array in ascending order using Quick Sort.

Description : Create three classes one Array class with array[], beg, and end fields and with a parameterized constructor and another QuickSort class with two static methods

1. static int[] quickSort(Array array1), which accepts Array object and returns loc as an int value.
2. static int[] quickSort(Array array1), which accepts Array object and returns array[] as an int[] array.

and one MainClass with one static method

1. public static void main method, for reading the size of the array and int array1[] values from input devices and call the quickSort method to do Quick Sort.

Code:

**package** yaksha;

**public** **class** Array {

**private** **int** array[];

**private** **int** beg;

**private** **int** end;

**public** Array(**int**[] array, **int** beg, **int** end) {

**super**();

**this**.array = array;

**this**.beg = beg;

**this**.end = end;

}

**public** **int**[] getArray() {

**return** array;

}

**public** **void** setArray(**int**[] array) {

**this**.array = array;

}

**public** **int** getBeg() {

**return** beg;

}

**public** **void** setBeg(**int** beg) {

**this**.beg = beg;

}

**public** **int** getEnd() {

**return** end;

}

**public** **void** setEnd(**int** end) {

**this**.end = end;

}

}

**package** yaksha;

**public** **class** QuickSort {

**public** **static** **int** partition(**int** a[], **int** beg, **int** end) {

**int** left, right, temp, loc, flag;

loc = left = beg;

right = end;

flag = 0;

**while** (flag != 1) {

**while** ((a[loc] <= a[right]) && (loc != right))

right--;

**if** (loc == right)

flag = 1;

**else** **if** (a[loc] > a[right]) {

temp = a[loc];

a[loc] = a[right];

a[right] = temp;

loc = right;

}

**if** (flag != 1) {

**while** ((a[loc] >= a[left]) && (loc != left))

left++;

**if** (loc == left)

flag = 1;

**else** **if** (a[loc] < a[left]) {

temp = a[loc];

a[loc] = a[left];

a[left] = temp;

loc = left;

}

}

}

**return** loc;

}

**static** **int**[] quickSort(Array array1) {

**int** array[] = array1.getArray();

**int** beg = array1.getBeg();

**int** end = array1.getEnd();

**int** loc;

**if** (beg < end) {

loc = *partition*(array, beg, end);

Array array2 = **new** Array(array, beg, loc - 1);

*quickSort*(array2);

Array array3 = **new** Array(array, loc + 1, end);

*quickSort*(array3);

}

**return** array;

}

}

**package** yaksha;

**import** java.util.Scanner;

**public** **class** MainClass {

**public** **static** **void** main(String[] args) {

**int** i;

Scanner scanner = **new** Scanner(System.***in***);

**int**[] array1;

**int** sizeOfArray;

System.***out***.println("Enter size of the Array :");

sizeOfArray = scanner.nextInt();

array1 = **new** **int**[sizeOfArray];

System.***out***.println("Enter numbers :");

**for** (i = 0; i < array1.length; i++) {

array1[i] = scanner.nextInt();

}

Array array = **new** Array(array1, 0, array1.length - 1);

**int** result[] = QuickSort.*quickSort*(array);

System.***out***.println("\n The sorted array is: \n");

**for** (i = 0; i < 10; i++) {

System.***out***.println(result[i]);

}

scanner.close();

}

}

Junit Testing

**package** yaksha;

**import** **static** yaksha.TestUtils.*businessTestFile*;

**import** **static** yaksha.TestUtils.*currentTest*;

**import** **static** yaksha.TestUtils.*yakshaAssert*;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** org.junit.jupiter.api.Test;

**public** **class** TestMainClass {

@Test

**public** **void** testExceptionConditon() **throws** Exception {

TestUtils.*yakshaAssert*(TestUtils.*currentTest*(), **true**, TestUtils.*boundaryTestFile*);

}

@Test

**public** **void** testBoundaryCondition() **throws** Exception {

TestUtils.*yakshaAssert*(TestUtils.*currentTest*(), **true**, TestUtils.*exceptionTestFile*);

}

@Test

**void** testQuickSort() **throws** Exception {

Integer expectedResult[] = { 23, 29, 31, 34, 45, 65, 67, 89, 90, 101 };

List<Integer> expectedResultList = Arrays.*asList*(expectedResult);

**int**[] arr = { 90, 29, 101, 45, 65, 23, 67, 89, 34, 31 };

// Input : 90 29 101 45 65 23 67 89 34 31

Array array = **new** Array(arr, 0, arr.length - 1);

**int** result[] = QuickSort.*quickSort*(array);

List<Integer> resultList = **new** ArrayList<Integer>(result.length);

**for** (**int** i : result) {

resultList.add(i);

}

*yakshaAssert*(*currentTest*(), (expectedResultList.containsAll(resultList) ? "true" : "false"), *businessTestFile*);

}

}

**package** yaksha;

**import** java.io.File;

**import** java.io.FileWriter;

**import** java.io.IOException;

// boiler-plate code

**public** **class** TestUtils {

**public** **static** File *businessTestFile*;

**public** **static** File *boundaryTestFile*;

**public** **static** File *exceptionTestFile*;

**static** {

*businessTestFile* = **new** File("./output\_revised.txt");

*businessTestFile*.delete();

*boundaryTestFile* = **new** File("./output\_boundary\_revised.txt");

*boundaryTestFile*.delete();

*exceptionTestFile* = **new** File("./output\_exception\_revised.txt");

*exceptionTestFile*.delete();

}

**public** **static** **void** yakshaAssert(String testName, Object result, File file) **throws** IOException {

System.***out***.println("\n" + testName + "=" + result);

FileWriter writer = **new** FileWriter(file, **true**);

writer.append("\n" + testName + "=" + result);

writer.flush();

writer.close();

}

**public** **static** String currentTest() {

**return** Thread.*currentThread*().getStackTrace()[2].getMethodName();

}

}

pom.xml :

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>iiht.yaksha.qs</groupId>

<artifactId>QuickSortADSnAlgJ9EQ1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>QuickSortADSnAlgJ9EQ1</name>

<description>QuickSortADSnAlgJ9EQ1</description>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.9</maven.compiler.source>

<maven.compiler.target>${maven.compiler.source}</maven.compiler.target>

<junit.jupiter.version>5.5.2</junit.jupiter.version>

<junit.platform.version>1.5.2</junit.platform.version>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.12</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit.jupiter.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-runner</artifactId>

<version>${junit.platform.version}</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

</plugins>

</build>

</project>

Test Data1

Enter size of the Array :

10

Enter numbers :

90 29 101 45 65 23 67 89 34 31

The sorted array is:

23

29

31

34

45

65

67

89

90

101

Learning outcome: Participant could able to learn how to use array, loop, conditional statement and implementation of Quick Sort.