

CSE 565: Software Verification, Validation and Testing

Design of Experiments Assignment

Assignment – 1

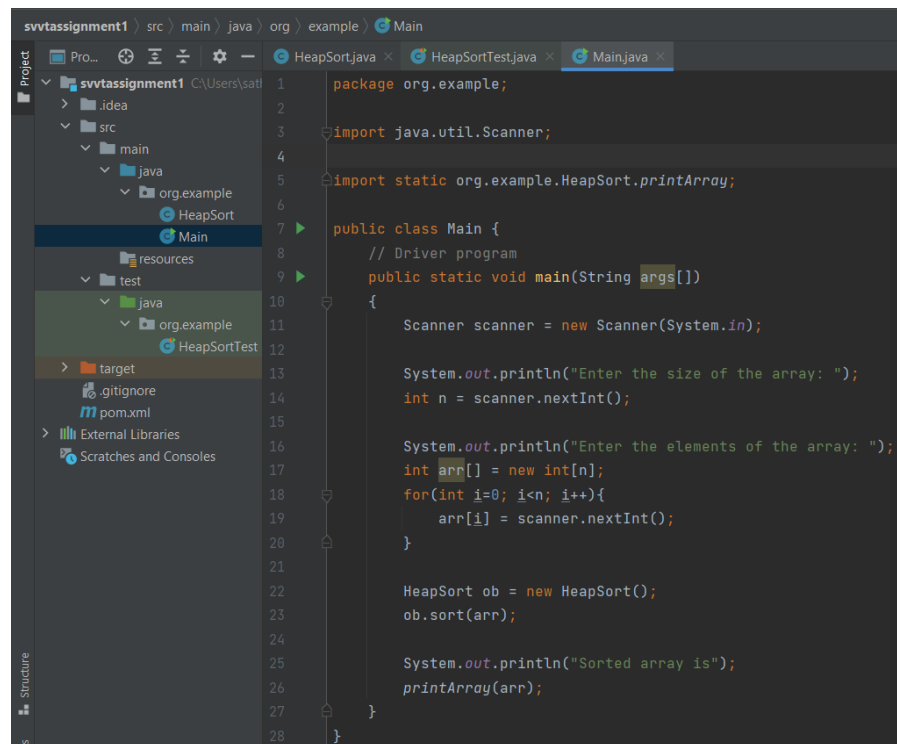
1. Development of an algorithm

This assignment follows the Maven Project Structure where the main source code resides under **src/main/java** and the test code resides under the folder **src/test/java**. The heap sort algorithm for this assignment is downloaded from [here](#). This has been modified to take array and its size as an input. The IDE here being used is IntelliJ.

There are two main source code files here. One is Main.java which is the driver code and HeapSort.java which contains the algorithm.

The below are screenshots of these files.

Main.java



```
1 package org.example;
2
3 import java.util.Scanner;
4
5 import static org.example.HeapSort.printArray;
6
7 public class Main {
8     // Driver program
9     public static void main(String args[])
10    {
11        Scanner scanner = new Scanner(System.in);
12
13        System.out.println("Enter the size of the array: ");
14        int n = scanner.nextInt();
15
16        System.out.println("Enter the elements of the array: ");
17        int arr[] = new int[n];
18        for(int i=0; i<n; i++){
19            arr[i] = scanner.nextInt();
20        }
21
22        HeapSort ob = new HeapSort();
23        ob.sort(arr);
24
25        System.out.println("Sorted array is");
26        printArray(arr);
27    }
28 }
```

HeapSort.java

```
1 package org.example;
2
3 // Java program for implementation of Heap Sort
4 public class HeapSort {
5     12 usages
6     public void sort(int arr[])
7     {
8         if(arr == null){
9             return;
10        }
11        int n = arr.length;
12
13        // Build heap (rearrange array)
14        for (int i = n / 2 - 1; i >= 0; i--)
15            heapify(arr, n, i);
16
17        // One by one extract an element from heap
18        for (int i = n - 1; i >= 0; i--) {
19            // Move current root to end
20            int temp = arr[0];
21            arr[0] = arr[i];
22            arr[i] = temp;
23
24            // call max heapify on the reduced heap
25            heapify(arr, i, 0);
26        }
27    }
28
29    // To heapify a subtree rooted with node i which is
30    // an index in arr[]. n is size of heap
31    3 usages
32    void heapify(int arr[], int n, int i)
33    {
34        int largest = i; // Initialize largest as root
35        int l = 2 * i + 1; // left = 2*i + 1
36        int r = 2 * i + 2; // right = 2*i + 2
37
38        // If left child is larger than root
39        if (l < n && arr[l] > arr[largest])
40            largest = l;
41
42        // If right child is larger than largest so far
43        if (r < n && arr[r] > arr[largest])
44            largest = r;
45
46        // If largest is not root
47        if (largest != i) {
48            int swap = arr[i];
49            arr[i] = arr[largest];
50            arr[largest] = swap;
51
52            // Recursively heapify the affected sub-tree
53            heapify(arr, n, largest);
54        }
55    }
56
57    /* A utility function to print array of size n */
58    2 usages
59    @ static void printArray(int arr[])
60    {
61        int n = arr.length;
62        for (int i = 0; i < n; ++i)
63            System.out.print(arr[i] + " ");
64        System.out.println();
65    }
66 }
```

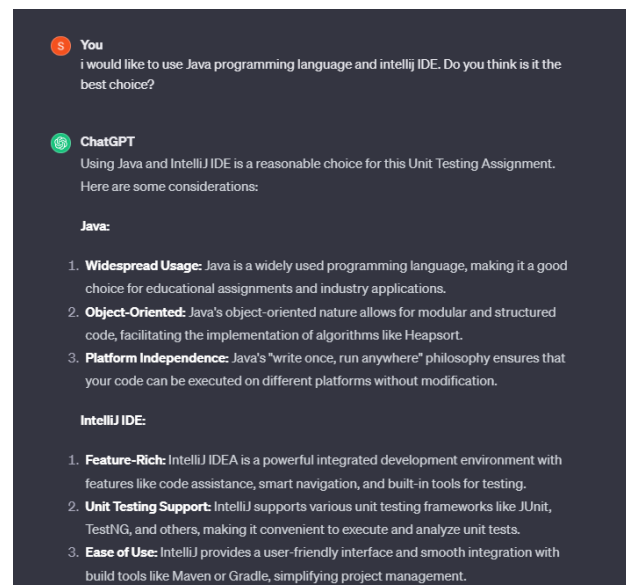
2. Explanation of Unit Testing Framework and prompt generation

In java, there are several unit testing frameworks available. For this assignment, I have used “**JUnit Testing Framework**” for writing unit test cases. It has several features such as Annotations, Assertions and parameterized tests. It is widely supported and well adopted and it can be integrated with many IDEs.

In a maven project, the libraries are added as a dependency in the pom.xml. The maven will take care to download the respective jar and make it available while building the project. Below, junit is added as dependency in pom.xml

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

The generative AI tool used here is “Chatgpt”. The below are the screenshots of the prompts tried and used to create unit level testing.



S You
what is the unit testing framework available in java?

ChatGPT

In Java, one of the most widely used unit testing frameworks is JUnit. JUnit is a powerful and popular framework for writing and running unit tests. It provides annotations to identify test methods, assertions for testing expected results, and various features for organizing and running tests.

Here are some key features of JUnit:

1. **Annotations:** JUnit uses annotations to mark methods as test methods, setup methods, and teardown methods. Annotations such as `@Test`, `@Before`, and `@After` are commonly used.
2. **Assertions:** JUnit provides a set of assertion methods to check if certain conditions are true. For example, `assertEquals`, `assertTrue`, `assertFalse`, etc.
3. **Test Runners:** JUnit supports test runners that help in organizing and executing tests. The standard test runner is `JUnitCore`, and it can be used from the command line or within an IDE.
4. **Parameterized Tests:** JUnit supports parameterized tests, allowing you to run the same test with different sets of parameters.
5. **Assertions for Exception Testing:** JUnit allows you to test if a specific exception is thrown during the execution of a test.

S You
Write some unit test cases for heap sort algorithm in Java?

ChatGPT

Certainly! Here are some basic unit test cases for the Heap Sort algorithm in Java using JUnit. These tests cover scenarios like sorting an already sorted array, sorting in reverse order, and handling an array with duplicate elements.

```
java Copy code

import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class HeapSortTest {

    @Test
    public void testHeapSort() {
        // Scenario 1: Basic test
        int[] arr1 = {12, 11, 13, 5, 6, 7};
```

S You
give me test cases for handling null inputs and large arrays

ChatGPT

Certainly! Here are additional test cases that cover handling null inputs and large arrays for the Heap Sort algorithm in Java:

```
java Copy code

import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNull;

public class HeapSortTest {

    @Test
    public void testHeapSort() {
        // Existing test cases...

        // Scenario 6: Handling null input
        int[] arr6 = null;
        HeapSort.sort(arr6);
        assertNull(arr6); // Sorting null should not throw an exception
```

3. Explanation of test cases by AI tool

The below is the screenshot of unit test cases generated by Chatgpt. There are 5 different scenarios being tested out here.

Scenario 1: Basic test

This is a basic test where an array of random elements is passed as an argument to the `HeapSort.sort()` function. The function should sort the array. The test case asserts the array passed with the sorted array. The test case will be passed when both the arrays are matched.

Scenario 2: Sorting an already sorted array

In this scenario, we will pass an already sorted array and test if the `HeapSort.sort()` function return the same array as is or not. The array will be asserted with the same sorted array.

Scenario 3: Sorting in reverse order

In this scenario, we will pass a strictly descending order array and test if the `HeapSort.sort()` returns the array in ascending order or not. The array will be asserted over an array of same elements in ascending order.

Scenario 4: Array with duplicate Elements

In this scenario, we will pass an array that contains duplicate elements and test if the algorithm sorts the algorithm properly or not. It should preserve the duplicate elements in adjacent locations and this will be asserted.

Scenario 5: Empty Array

In this scenario, an empty array will be passed to the `HeapSort.sort()` function. The test verifies if the function returns empty array without any error. The empty array will be asserted.

There are other two scenarios which are later asked by specifically mentioning in the prompt. They are null input and large arrays scenario.

Scenario 6: Null Input

This is a special case scenario where a null input is passed. The test verifies if the algorithm returns simply without performing anything or not when it encounters a null input.

Scenario 7: Large Array (10000 elements)

In this scenario, a large array of 10000 elements is created in reverse order and this is passed to the `HeapSort.sort()` function. The test verifies if the algorithm sorts the array in ascending order or not and asserts if each element is less than or equal to next element.

```
java Copy code

import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class HeapSortTest {

    @Test
    public void testHeapSort() {
        // Scenario 1: Basic test
        int[] arr1 = {12, 11, 13, 6, 6, 7};
        HeapSort.sort(arr1);
        assertEquals(new int[]{6, 6, 7, 11, 12, 13}, arr1);

        // Scenario 2: Sorting an already sorted array
        int[] arr2 = {1, 2, 3, 4, 5};
        HeapSort.sort(arr2);
        assertEquals(new int[]{1, 2, 3, 4, 5}, arr2);

        // Scenario 3: Sorting in reverse order
        int[] arr3 = {5, 4, 3, 2, 1};
        HeapSort.sort(arr3);
        assertEquals(new int[]{1, 2, 3, 4, 5}, arr3);

        // Scenario 4: Array with duplicate elements
        int[] arr4 = {5, 2, 3, 2, 1, 4, 1};
        HeapSort.sort(arr4);
        assertEquals(new int[]{1, 1, 2, 2, 3, 4, 5}, arr4);

        // Scenario 5: Empty array
        int[] arr5 = {};
        HeapSort.sort(arr5);
        assertEquals(new int[], arr5);
    }
}
```

```
java Copy code

import org.junit.Test;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNull;

public class HeapSortTest {

    @Test
    public void testHeapSort() {
        // Existing test cases...

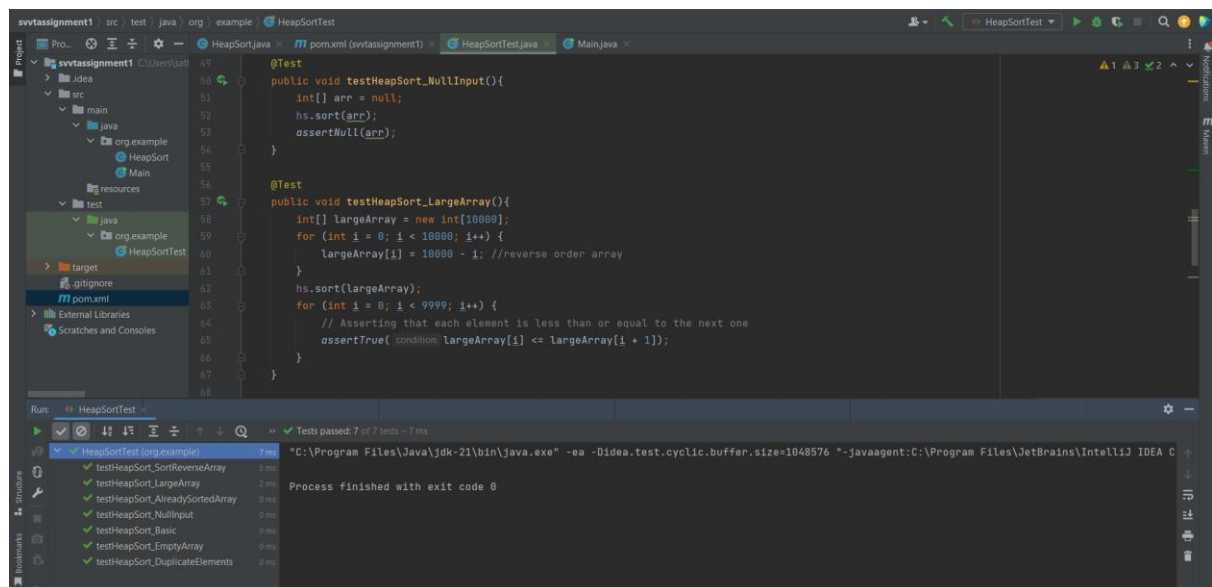
        // Scenario 6: Handling null input
        int[] arr6 = null;
        HeapSort.sort(arr6);
        assertNull(arr6); // Sorting null should not throw an exception

        // Scenario 7: Large array (10000 elements)
        int[] largeArray = new int[10000];
        for (int i = 0; i < 10000; i++) {
            largeArray[i] = 10000 - i; // Creating a reversed order array
        }
        HeapSort.sort(largeArray);
        for (int i = 0; i < 9999; i++) {
            // Asserting that each element is less than or equal to the
            // next element
            assertTrue(largeArray[i] <= largeArray[i + 1]);
        }
    }
}
```

4. Report out of test case execution

All the test generated by chatgpt have been accommodated under HeapSortTest.java file. Each test passes an array to be sorted to the HeapSort.sort() function and asserts the results to the expected results. If results are matched with expected, then the test case is passed otherwise it fails.

All the above 7 scenarios are added as 7 different test cases in HeapSortTest.java. When we run HeapSortTest, all the 7 tests will be executed one after the other. This took a total time of 7ms. We can also see that Sorted Reverse Array and Large Array took maximum time rather than other test cases like null input, empty array. The below screenshot shows the results of all the 7 test cases discussed above.



5. Assessment and further improvement of test cases

Although chatgpt has generated wide range of test cases including Basic array, already sorted order, reverse order, duplicate elements, empty array. There are certain improvements that can be done in these test cases.

- It could have given more variations of unsorted arrays, including arrays with different lengths or containing duplicate elements.
- While testing already sorted array, it could have include array which is mostly sorted but contain a few unsorted elements which can test algorithm stability.
- It could've generated partially sorted descending order array or high number of duplicate elements widespread to test algorithm in mixed scenarios.

To further test the algorithm, Four new test cases covering different scenarios have been developed by me. They are:

Scenario 8: Single Element Array.

In this scenario, only a single element will be present in the array. The test asserts the array.

Scenario 9: All same elements

In this scenario, the array contains all same elements and the expects the function to return the array as is.

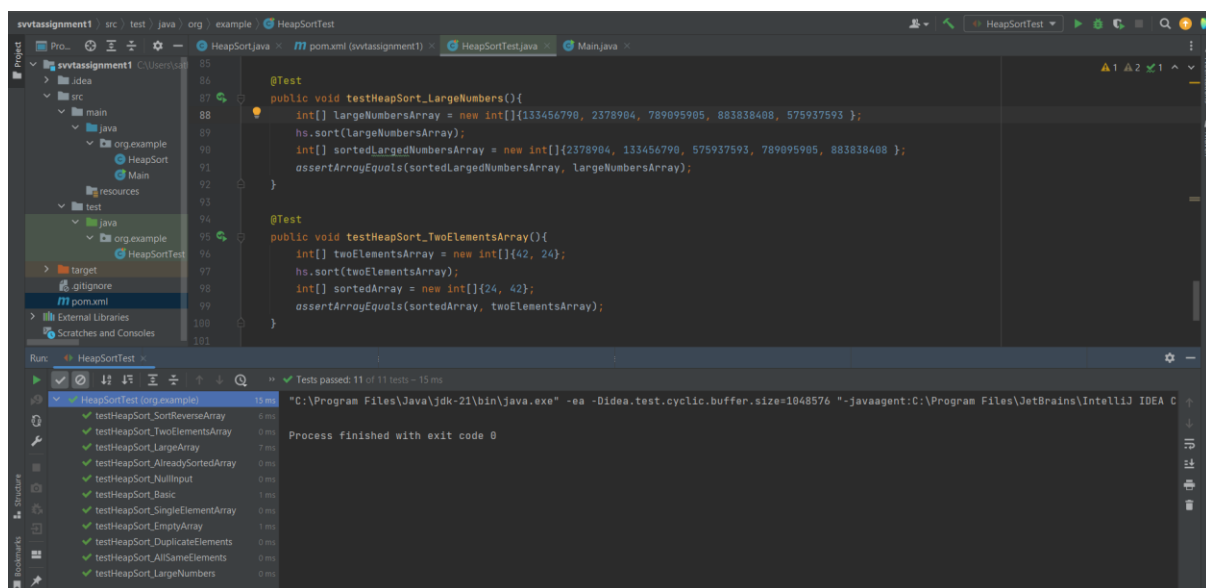
Scenario 10: Large Number Array

In this scenario, the array contains large integers such as 9-digit numbers. The test verifies if the algorithm sorts these large number correctly or not.

Scenario 11: Two Elements array

This is one of the basic test cases that is missed by chatgpt. In this case, only two elements are present in the array. The test asserts if the array is sorted in ascending order or not.

The below is the screenshot of all 11 test cases. We can see that 11 tests are passed and it took over 15ms of time.



6. An assessment of the generative AI tool

The generative AI tool used for this assignment is “Chatgpt”. It offered a user-friendly interface, making it a straight forward conversation between the user and the AI tool. The prompts were intuitive, the response time was impressive, different prompts yielded a variety of test cases and overall experience using the tool was positive.

To assess, the quality of unit test cases generated by Chatgpt using Junit test framework are good. It seamlessly adapted to the Junit testing framework and generated test cases that can be directly included into the assignment without major changes or debugging. The AI has generated the test cases with detailed explanation that helps the user to understand the test cases quickly without investing much time. It effectively covered various scenarios including boundary cases. In this way, it added a great value by saving time and enhancing test coverage, However, when asked for unit test cases in the first prompt it has missed certain edge case scenarios such as null input which is a basic test case to be tested. It generated the other two scenarios when specifically mentioned in the user prompt. Another room for improvement is providing more effective test cases like containing more complex arrays/inputs that can be test and making the tests high quality.