

**1. What is a unit testing framework?** Unit Testing frameworks provide a wireframe for developers/testers to write and execute unit test cases which can be used to test individual components or units of software. These units can be procedures, functions or classes. Additionally, they also provide comprehensive reports of test results. They are used in every stage of a software development life cycle including requirement gathering, performance optimization and quality assurance. A software developer/tester can simulate different scenarios to test the program and check if it fails. Unit Test frameworks allow us to quickly write test cases, speed up the coding process, and bug detection is easier. [1]

**2. How would a developer utilize a unit testing framework?** Unit testing framework is utilized by the developer to automate the test cases. A framework provides supporting functionality in every stage of software development. The developer can verify very complex architectures. The tests are more effective and better than developing standalone tests. The verification happens faster and quicker as the results are produced immediately. The developer is provided with base classes/interfaces that need to be inherited for testing purposes and attributes that can be placed in the program to mark certain functions/classes as test-based functions/classes. The internal structure of the code is also assessed through unit testing.

**3. What benefits does a unit testing framework provide?** Helps with easier and quicker creation of test cases. These frameworks help save time and money in the long run. These frameworks help in easy and early bug detection and refactoring of source code. Help improve the quality of code. It helps you understand the performance metrics of your software. [3]

**4. Identify 2 unit testing frameworks for further consideration (one framework must be JBehave).**

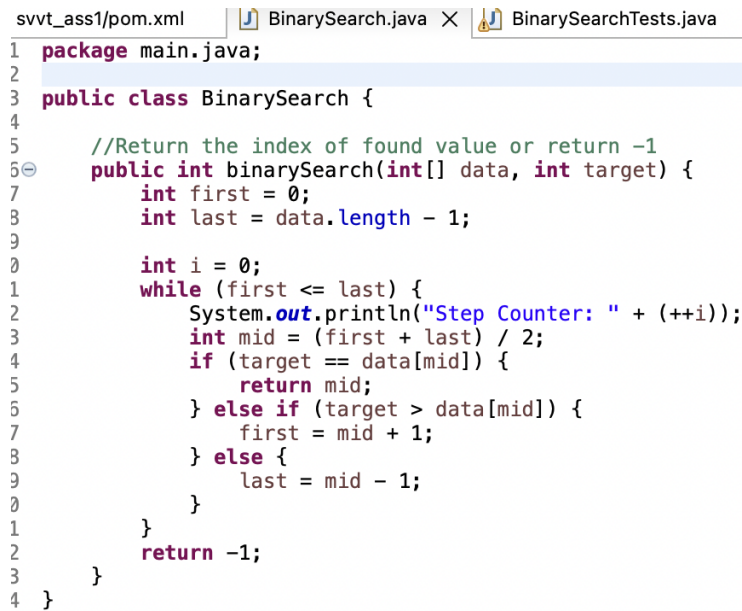
I compare 2 Java-based testing frameworks - JUnit and JBehave.

Comparison parameter	JUnit	JBehave
General info	JUnit is an open source Unit testing framework for java	JBehave is a Behavior-Driven Development testing framework for java.
Group fixtures	You can use setUp() and tearDown() inbuilt functions as group fixtures.	You can define group fixtures with JBehave.
Client-side	You can test front-end components such as individual classes and functions that create the front-end.	You can test front-end behavior (scenarios) with JBehave
Server-side	You can test classes and functions that compose the back-end such as database connections and so on.	JBehave tests scenarios and behaviors of components, it can test back-end behavior.
Mocks	JUnit does not support mocking internally but you can use a mock framework like Mockito to generate mock objects.	The best way to mock is to use third party libraries like Mockito, Jmock or Jmockit.

## Part 2

There are 2 Java files.

1. BinarySearch.java - This class has all the core logic for Binary Search.



```
svvt_ass1/pom.xml | BinarySearch.java | BinarySearchTests.java
1 package main.java;
2
3 public class BinarySearch {
4
5     //Return the index of found value or return -1
6     public int binarySearch(int[] data, int target) {
7         int first = 0;
8         int last = data.length - 1;
9
10        int i = 0;
11        while (first <= last) {
12            System.out.println("Step Counter: " + (++i));
13            int mid = (first + last) / 2;
14            if (target == data[mid]) {
15                return mid;
16            } else if (target > data[mid]) {
17                first = mid + 1;
18            } else {
19                last = mid - 1;
20            }
21        }
22        return -1;
23    }
24 }
```

## 2. BinarySearchTests.java - This class has all the test cases for BinarySearch.java

```
1 package test.java;
2
3 import org.junit.jupiter.api.Assertions;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 import main.java.BinarySearch;
8
9 import static org.junit.jupiter.api.Assertions.*;
10
11 public class BinarySearchTests {
12
13     private BinarySearch binarySearch;
14     private int[] sample;
15
16     @BeforeEach
17     public void setUp() {
18         binarySearch = new BinarySearch();
19     }
20
21     @Test
22     public void binarySearch_Found() {
23         int expected = 3;
24         int actual = binarySearch.binarySearch(new int[]{4, 5, 1, 4, 10, 3, -34, 7}, 4);
25         assertEquals(expected, actual);
26     }
27
28     @Test
29     public void binarySearch_NotFound() {
30         int expected = -1;
31         int actual = binarySearch.binarySearch(new int[]{-4, 5, 1, 4, 10, 3, -34, 7}, 40);
32         assertEquals(expected, actual);
33     }
34
35     @Test
36     public void binarySearch_EmptyArray() {
37         int expected = -1;
38         int actualIndex = binarySearch.binarySearch(new int[] {}, 1);
39         assertEquals(expected, actualIndex);
40     }
41
42     @Test
43     public void binarySearch_ArrayWithLengthOne() {
44         int expected = 0;
45         int actualIndex = binarySearch.binarySearch(new int[] {4}, 4);
46         assertEquals(expected, actualIndex);
47     }
48
49     @Test
50     public void binarySearch_ArrayWithEvenLengthTarget_Found() {
51         int expected = 0;
52         int actualIndex = binarySearch.binarySearch(new int[] {-4, 2, 5, 1, 4, 10, 3, 7}, -4);
53         assertEquals(expected, actualIndex);
54     }
55
56
57
```

The output after running the test cases are:

The screenshot shows an IDE's JUnit test runner interface. At the top, the 'Package Explorer' and 'JUnit' tabs are visible. Below the tabs, a status bar indicates 'Finished after 0.344 seconds'. A summary bar shows 'Runs: 5/5', 'Errors: 0', and 'Failures: 0'. A green progress bar is displayed below the summary. The test results are listed in a tree view under the 'BinarySearchTest' entry, which has a total execution time of 0.114 s. The individual test cases and their execution times are:

- binarySearch\_ArrayWithLengthOne() (0.087 s)
- binarySearch\_Found() (0.002 s)
- binarySearch\_EmptyArray() (0.006 s)
- binarySearch\_ArrayWithEvenLengthTarget\_Found() (0.003 s)
- binarySearch\_NotFound() (0.002 s)

All test cases are marked with a green checkmark icon, indicating they passed successfully.

Additionally we can configure a surefire plugin to generate reports. Attached is a screenshot of the same:

---

Last Published: 2022-09-08 | Version: 0.0.1-SNAPSHOT

---

 [Built by Maven](#)

## Surefire Report

### Summary

[\[Summary\]](#) [\[Package List\]](#) [\[Test Cases\]](#)

Tests	Errors	Failures	Skipped	Success Rate	Time
5	0	0	0	100%	0.021

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

### Package List

[\[Summary\]](#) [\[Package List\]](#) [\[Test Cases\]](#)

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
<a href="#">test.java</a>	5	0	0	0	100%	0.021

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

#### test.java

	Class	Tests	Errors	Failures	Skipped	Success Rate	Time
	<a href="#">BinarySearchTests</a>	5	0	0	0	100%	0.021

# Test Cases

[\[Summary\]](#) [\[Package List\]](#) [\[Test Cases\]](#)

## BinarySearchTests

	binarySearch_ArrayWithLengthOne	0.012
	binarySearch_Found	0
	binarySearch_EmptyArray	0.001
	binarySearch_ArrayWithEvenLengthTarget_Found	0.001
	binarySearch_NotFound	0

## References:

1. Tutorialspoint.com. 2022. UnitTest Framework - Overview. [online]  
[https://www.tutorialspoint.com/unittest\\_framework/unittest\\_framework\\_overview.html](https://www.tutorialspoint.com/unittest_framework/unittest_framework_overview.html)
2. Unit Test Frameworks by Paul Hamill [Reference book]  
<https://www.oreilly.com/library/view/unit-test-frameworks/0596006896/ch01.html>
3. Why is unit testing important for developers? [online]  
<https://www.techtarget.com/searchsoftwarequality/answer/Is-unit-testing-an-important-aspect-of-software-development>
4. JBehave vs JUnit comparison of testing frameworks What are the differences between JBehave and JUnit?  
[online] [https://knapsackpro.com/testing\\_frameworks/difference\\_between/jbehave/vs/junit](https://knapsackpro.com/testing_frameworks/difference_between/jbehave/vs/junit)