Assignment [1]: **Id: 1207654691**

Part 1. (1 page)

1.What is a unit testing framework?

→ **Unit testing** is the testing method to test the functions or units of the code to determine that the functions are working properly. That means for the particular input sets the functions return the expected output. **Unit testing frameworks** are the third party products which are used to simplify the process of the unit testing.

2. How would a developer utilize a framework?

→ Developer might utilize a framework which is existing already for the features that it provides through the APIs of the implementation of the framework. This way the developer does not need to reinvent the wheel for the features that are a part of the framework implementation, and can directly use the features.

3. What benefits does a framework provide?

Benefits of framework:

1). Reduce the time and energy to develop/test a software.

2). It provides a standard working environment which helps the user to develop the module of the application or whole application.

3). Framework helps to developers to provide more time in developing/testing instead of preparing the environment for the developing/testing the software.

4). It helps to separate the business logic from the user interface and make code more clean and extensible.

5). Reuse the pre developed or pre tested code.

6). Reduce the programming and testing efforts.

7). Increase the reliability of a new software.

4. Identify 2 frameworks for further consideration (one framework must be CppUnit)

Compare the frameworks in terms of common capabilities as well as any differences

→ Frameworks: [1] Junit: eclipse, [2] CppUnit: eclipse.

Both are important in development in test driven development. Both are part of xUnit.

Junit	**CppUnit**
Used to test java code	Used to test C++ code
xUnit that originated with SUnit	C++ port of Junit
Junit is used to develop other unit testing	Developed from Junit

Part 2.

Download or write code to implement the binary search algorithm. Utilize a unit testing framework of your choice to test your code. Submit a copy of the code to test, the test cases generated and an output report (screen shot from the tool).

Code: 1) Binary search:

```
static int[] data;
static int size, key;
public static int binarySearch(int key,int[] data){
         int low = 0;
         int size = data.length;
         int high = size - 1;

         if(data == null)
                    return -2;
             else if(data.length == 0) {
           }
           for (int i = 0; i < data.length-1; i++) {

             if(data[i] > data[i+1]) {
                    return -2;
               }
               else{

               }
           }
         while(high >= low) {
             int middle = (low + high) / 2;
             if(data[middle] == key) {
              return middle;
             }
             if(data[middle] < key) {
                 low = middle + 1;
             }
             if(data[middle] > key) {
                 high = middle - 1;
             }
        }
        return -1;
}
```

Test cases:

1. **Key found**

```
public class BsearchTest {

        int[] data = {1,2,3,4,5,6};
        int key = 4;
        @Test
        public void testFound() {
                Bsearch bs = new Bsearch();
                int ans = bs.binarySearch(key, data);
                assertEquals(3, ans);
        }
}
```

2. **Key not found**

```
public class keynotfound {

        int[] data = {1,2,3,4,5,6};
        int key = 7;
        @Test
        public void test() {
                Bsearch bs = new Bsearch();
                int ans = bs.binarySearch(key, data);
                assertEquals(5, ans);
        }
  }
```

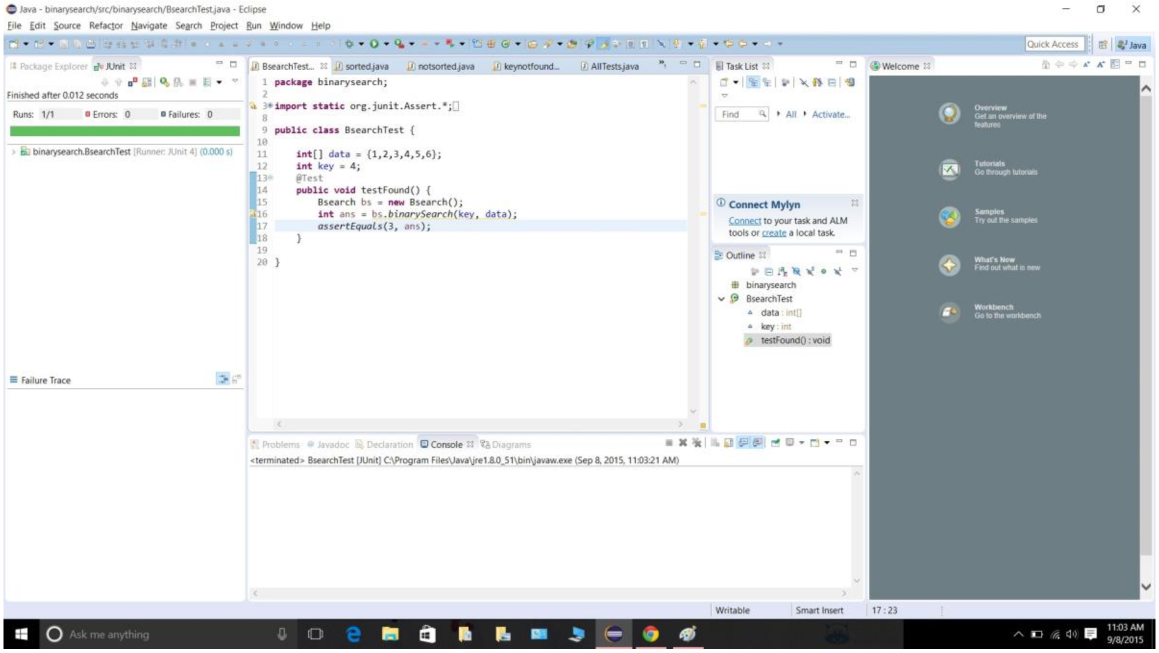
3. **Array is sorted**

```
public class sorted {

        int[] data = {1,2,5,9,20,25,32};
        int key = 20;

        @Test
        public void testsort() {
                Bsearch bs = new Bsearch();
                int ans = bs.binarySearch(key, data);

           assertEquals(4, ans);
        }
}
```

4. **Array is not sorted**

```
public class notsorted {

        int[] data = {1,2,5,9,8,25,4};
        int key = 4;
        @Test
        public void testNotSored() {
                Bsearch bs = new Bsearch();
                int ans = bs.binarySearch(key, data);

           assertEquals(7, ans);
        }
}
```

Screenshots:



Finished after 0.012 seconds

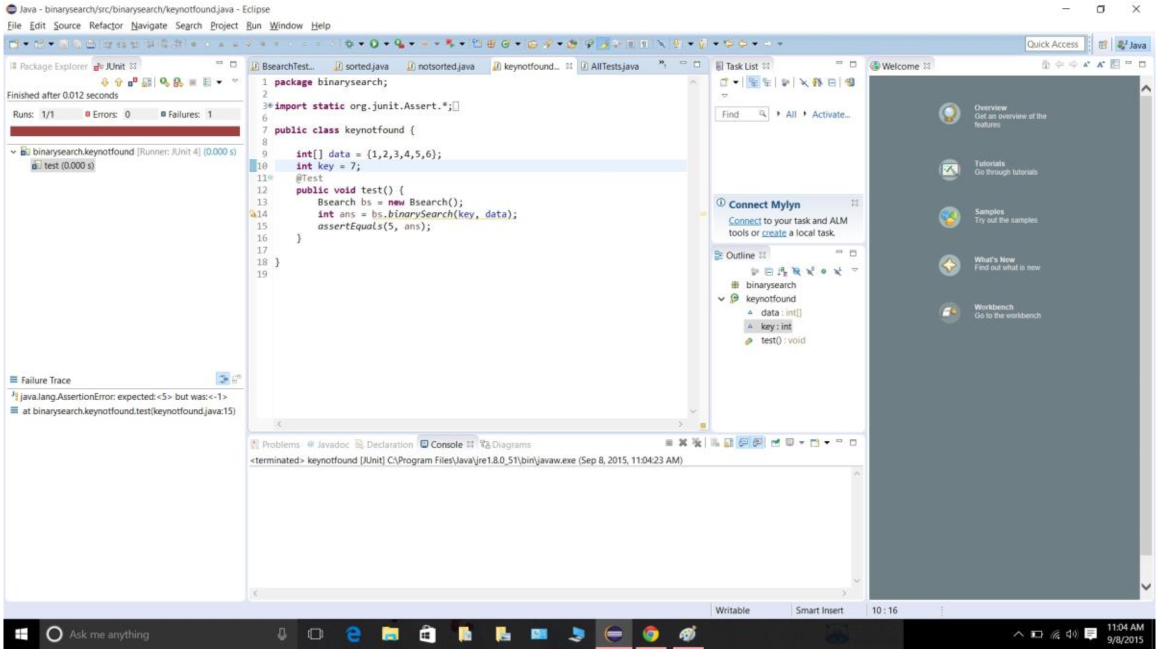
Runs: 1/1 Errors: 0 Failures: 0

binarysearch.BsearchTest [Runner: JUnit 4] (0.000 s)

```java
package binarysearch;

import static org.junit.Assert.*;□

public class BsearchTest {

    int[] data = {1,2,3,4,5,6};
    int key = 4;
    @Test
    public void testFound() {
        Bsearch bs = new Bsearch();
        int ans = bs.binarySearch(key, data);
        assertEquals(3, ans);
    }

}
```

```
<terminated> BsearchTest [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 8, 2015, 11:03:21 AM)
```

Test case: 1



Finished after 0.012 seconds

Runs: 1/1 Errors: 0 Failures: 1

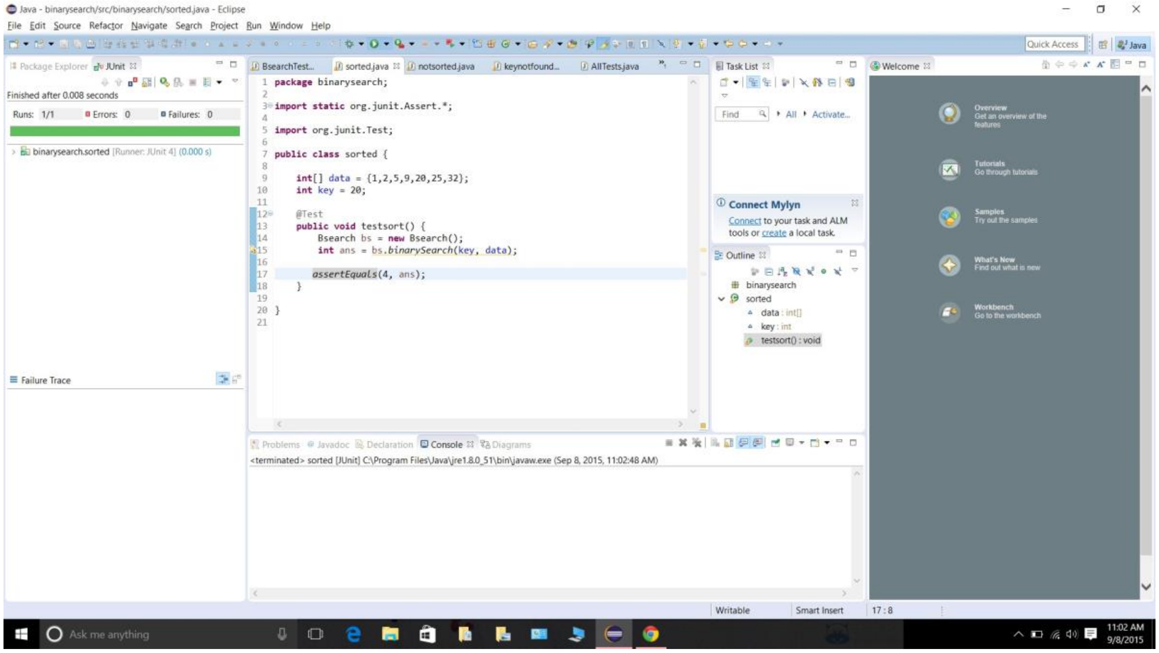
binarysearch.keynotfound [Runner: JUnit 4] (0.000 s)
test (0.000 s)

```
java.lang.AssertionError: expected:<5> but was:<-1>
at binarysearch.keynotfound.test(keynotfound.java:15)
```

```java
package binarysearch;

import static org.junit.Assert.*;□

public class keynotfound {

    int[] data = {1,2,3,4,5,6};
    int key = 7;
    @Test
    public void test() {
        Bsearch bs = new Bsearch();
        int ans = bs.binarySearch(key, data);
        assertEquals(5, ans);
    }

}
```

```
<terminated> keynotfound [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 8, 2015, 11:04:23 AM)
```

Test case: 2



Finished after 0.008 seconds

Runs: 1/1 Errors: 0 Failures: 0

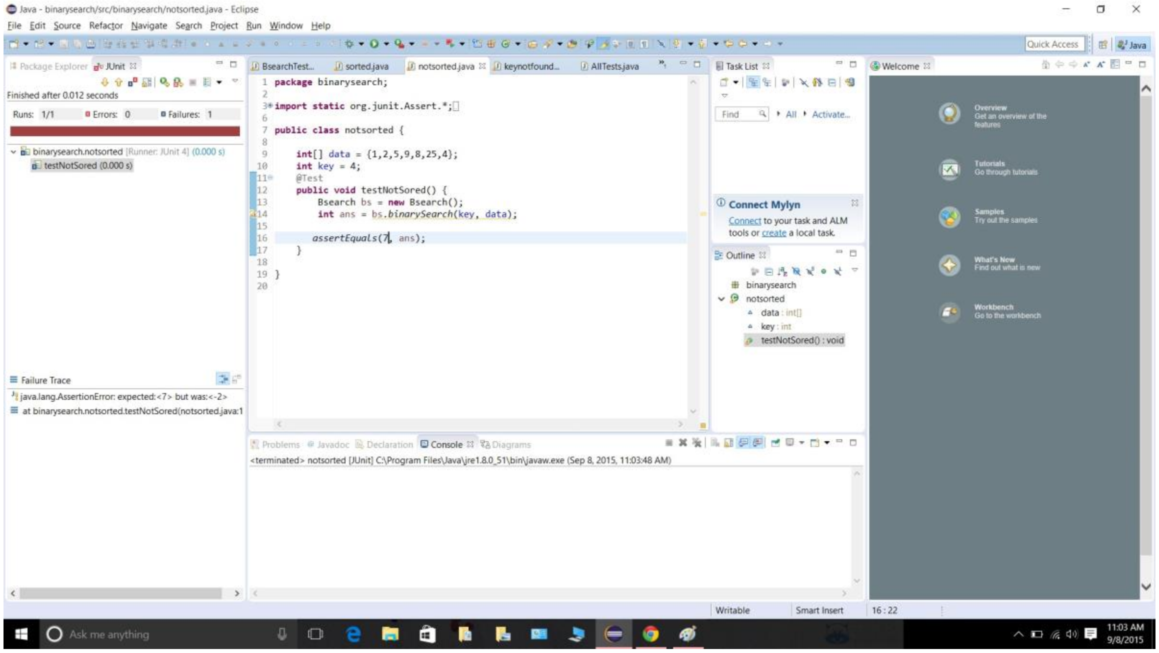
binarysearch.sorted [Runner: JUnit 4] (0.000 s)

Failure Trace

```
package binarysearch;

import static org.junit.Assert.*;

import org.junit.Test;

public class sorted {

    int[] data = {1,2,5,9,20,25,32};
    int key = 20;

    @Test
    public void testsort() {
        Bsearch bs = new Bsearch();
        int ans = bs.binarySearch(key, data);

        assertEquals(4, ans);
    }

}
```

```
<terminated> sorted [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 8, 2015, 11:02:48 AM)
```

Test case: 3



Finished after 0.012 seconds

Runs: 1/1 Errors: 0 Failures: 1

binarysearch.notsorted [Runner: JUnit 4] (0.000 s)
testNotSored (0.000 s)

Failure Trace

```
java.lang.AssertionError: expected:<7> but was:<-2>
at binarysearch.notsorted.testNotSored(notsorted.java:1
```

```
package binarysearch;

import static org.junit.Assert.*;

public class notsorted {

    int[] data = {1,2,5,9,8,25,4};
    int key = 4;
    @Test
    public void testNotSored() {
        Bsearch bs = new Bsearch();
        int ans = bs.binarySearch(key, data);

       assertEquals(7, ans);
    }

}
```

```
<terminated> notsorted [JUnit] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Sep 8, 2015, 11:03:48 AM)
```

Test case: 4

References:

[1] http://info.cimetrix.com/blog/bid/22339/What-is-a-Software-Framework-And-why-should-you-like-em

[2] https://nagbhushan.wordpress.com/2010/10/03/framework-advantages-and-disadvantages/

[3] http://code.tutsplus.com/articles/the-beginners-guide-to-unit-testing-what-is-unit-testing--wp-25728

[4] https://en.wikipedia.org/wiki/Unit_testing