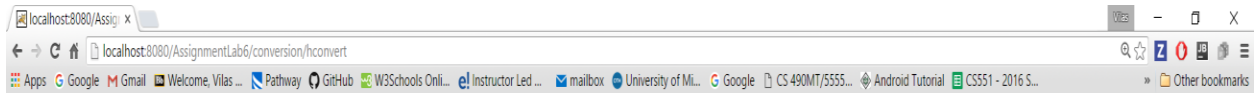


## LAB6 ASSIGNMENT:

Here we have taken the two conversions services. First one is height service. Here takes input as feet and inches and calculates and displays your height in cm as follows:

By default, it will take the input values and displays as below:

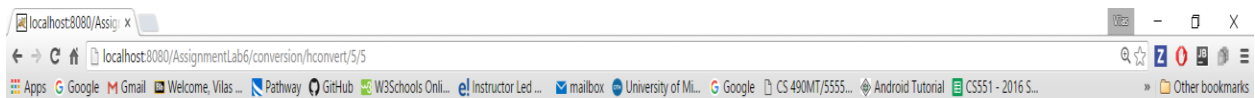


This XML file does not appear to have any style information associated with it. The document tree is shown below.

---

```
▼ <Height>
  <feet>1.0</feet>
  <inch>0.0</inch>
  <Heightoutput>Output: Height converted in CM 30.48</Heightoutput>
</Height>
```

You can give your height also as below (here 5 feet 5 inch is given as input)



This XML file does not appear to have any style information associated with it. The document tree is shown below.

---

```
▼ <Height>
  <feet>5.0</feet>
  <inch>5.0</inch>
  <Heightoutput>Output: Height converted in CM 165.1</Heightoutput>
</Height>
```

Another service is weight service:

Here it will convert your weight from kg to pounds as well as ounces. By default, it will read default inputs and gives result as below:



This XML file does not appear to have any style information associated with it. The document tree is shown below.

---

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Weight>
  <kg>1.0</kg>
  <pound>2.20462</pound>
  <ounce>35.274</ounce>
  <output>
    Output: Weight converted from kg to pounds and ounces 2.20462 and 35.274
  </output>
</Weight>
```

Here you can also give your input weight and see its converted values in pounds and ounces.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

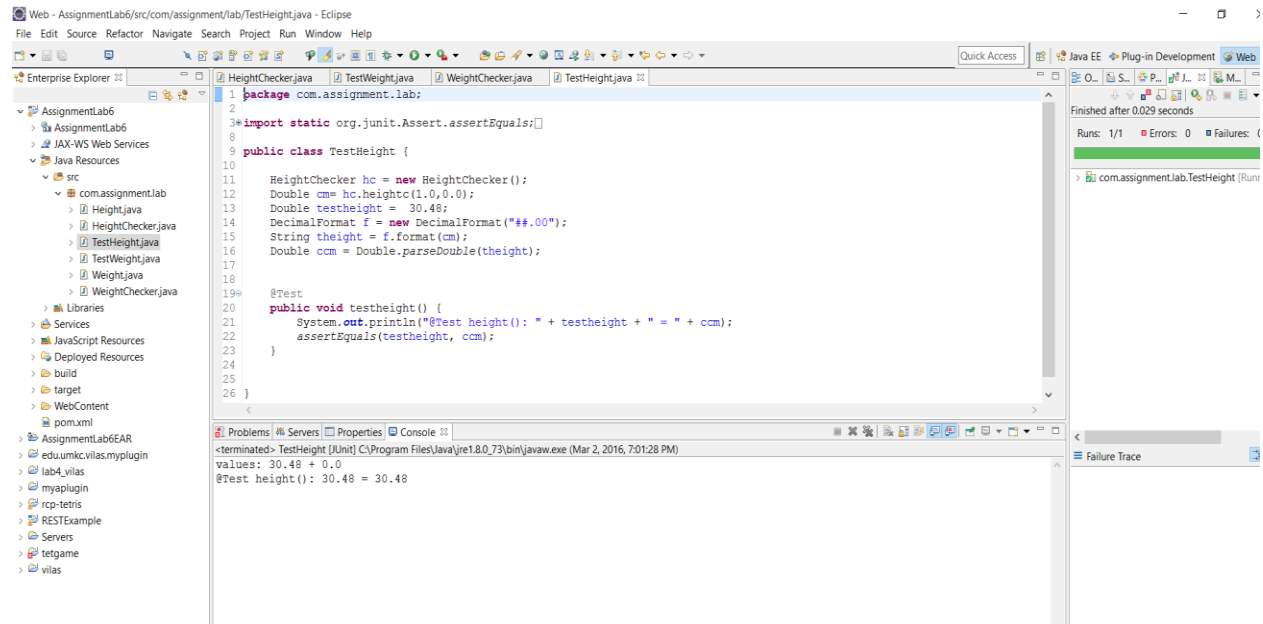
---

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Weight>
  <kg>75.0</kg>
  <pound>165.3465</pound>
  <ounce>2645.55</ounce>
  <output>
    Output: Weight converted from kg to pounds and ounces 165.3465 and 2645.55
  </output>
</Weight>
```

## Testcases:

Here we are having the two test cases related to height and weight. Consider the height here we get the two cases one is success case and failure test case.

The successful test case evaluates the input from the user and calculated value as below and if success will display as below:



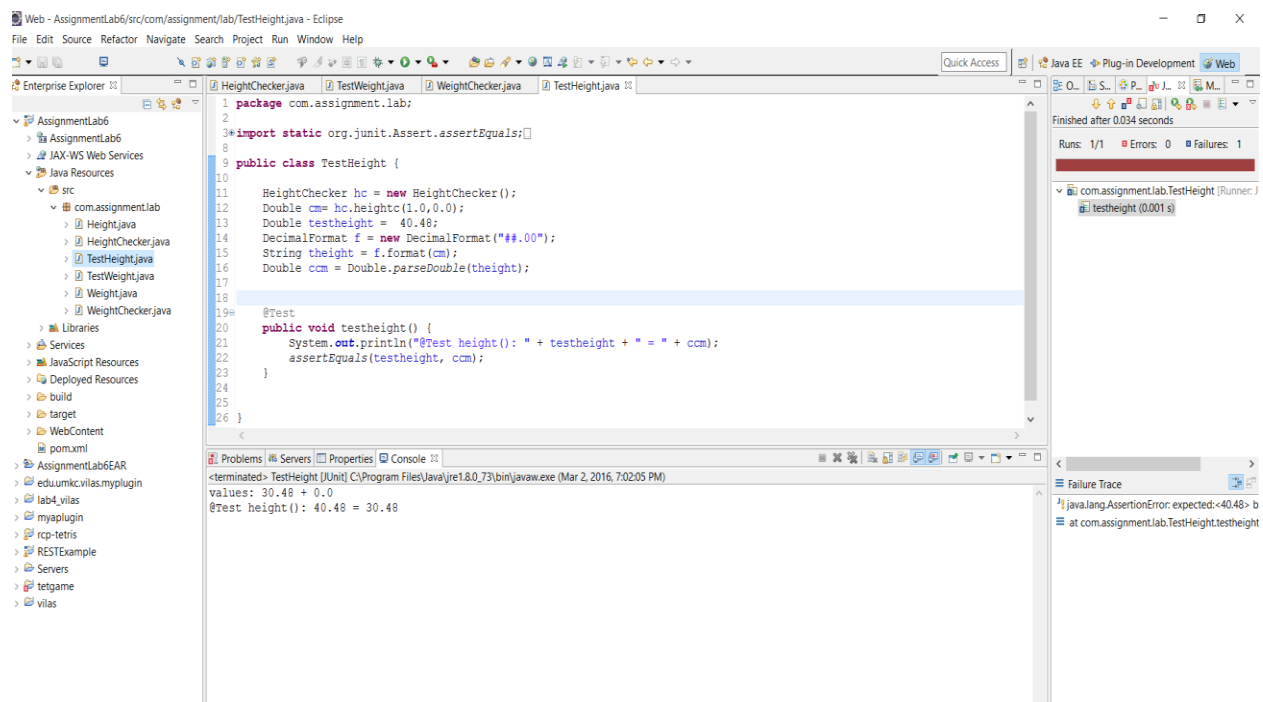
The screenshot shows the Eclipse IDE with the file `TestHeight.java` open. The code defines a `TestHeight` class with a `testheight()` method. The method uses `HeightChecker` to calculate a value, formats it, and asserts it against the input. The console output shows the test passing: `@Test height(): 30.48 = 30.48`. The right-hand side of the IDE shows the 'Run' button and a progress bar indicating the test finished after 0.029 seconds with 0 errors and 0 failures.

```
1 package com.assignment.lab;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class TestHeight {
6
7     HeightChecker hc = new HeightChecker();
8     Double cm = hc.heightc(1.0,0.0);
9     Double testheight = 30.48;
10    DecimalFormat f = new DecimalFormat("##.00");
11    String theight = f.format(cm);
12    Double ccm = Double.parseDouble(theight);
13
14    @Test
15    public void testheight() {
16        System.out.println("@Test height(): " + testheight + " = " + ccm);
17        assertEquals(testheight, ccm);
18    }
19 }
20
21
22
23
24
25
26
```

Console Output:  
@Test height(): 30.48 = 30.48

Run Summary:  
Finished after 0.029 seconds  
Runs: 1/1 Errors: 0 Failures: 0

Whereas for the failure case you will get the output as below:



The screenshot shows the Eclipse IDE with the file `TestHeight.java` open. The code is the same as in the previous screenshot, but the `testheight` variable is set to 40.48. The console output shows the test failing: `@Test height(): 40.48 = 30.48`. The right-hand side of the IDE shows the 'Run' button and a progress bar indicating the test finished after 0.034 seconds with 1 error and 1 failure. The 'Failure Trace' pane shows the `java.lang.AssertionError` message.

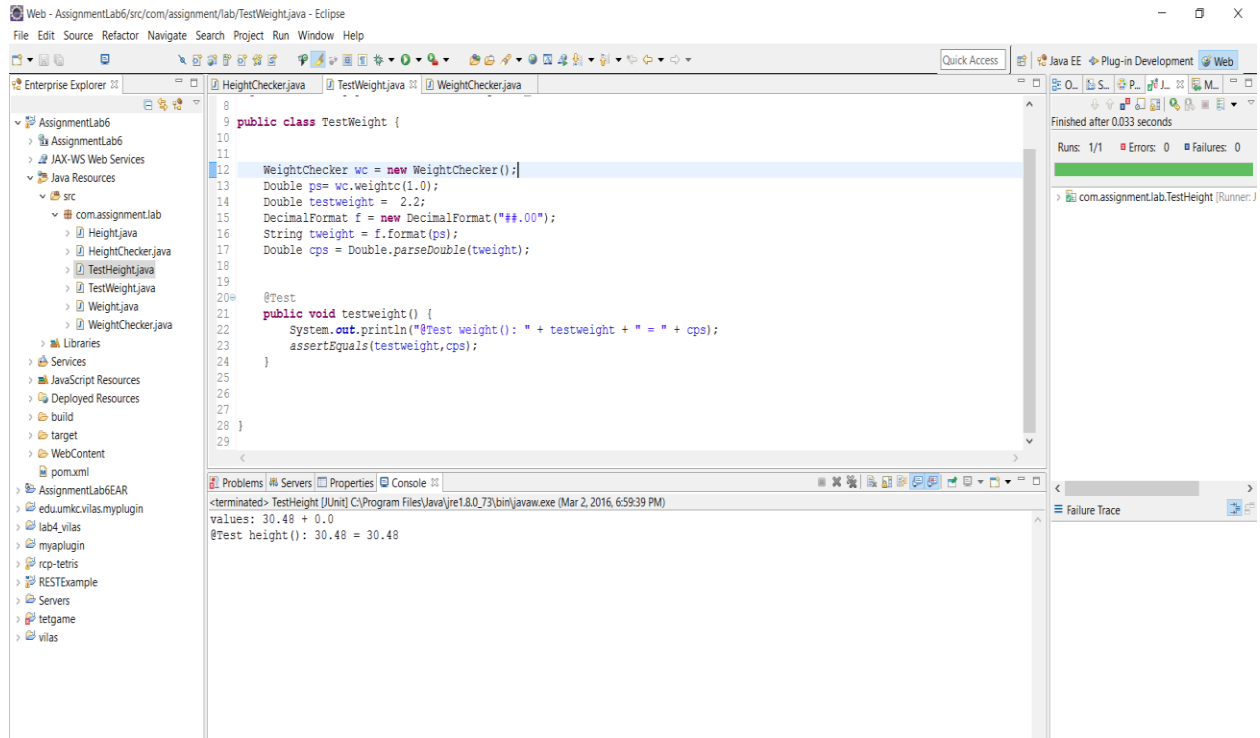
```
1 package com.assignment.lab;
2
3 import static org.junit.Assert.assertEquals;
4
5 public class TestHeight {
6
7     HeightChecker hc = new HeightChecker();
8     Double cm = hc.heightc(1.0,0.0);
9     Double testheight = 40.48;
10    DecimalFormat f = new DecimalFormat("##.00");
11    String theight = f.format(cm);
12    Double ccm = Double.parseDouble(theight);
13
14    @Test
15    public void testheight() {
16        System.out.println("@Test height(): " + testheight + " = " + ccm);
17        assertEquals(testheight, ccm);
18    }
19 }
20
21
22
23
24
25
26
```

Console Output:  
@Test height(): 40.48 = 30.48

Run Summary:  
Finished after 0.034 seconds  
Runs: 1/1 Errors: 1 Failures: 1

Failure Trace:  
java.lang.AssertionError: expected:<40.48> but was:<30.48>  
at com.assignment.lab.TestHeight.testheight()

For weight calculation test cases also we have success case and the failure cases



Failure case:

