AP Computer Science A: Using the `Run Tests` Button in BlueJ

Instructor: Mr. Alwin Tareen

Task Overview

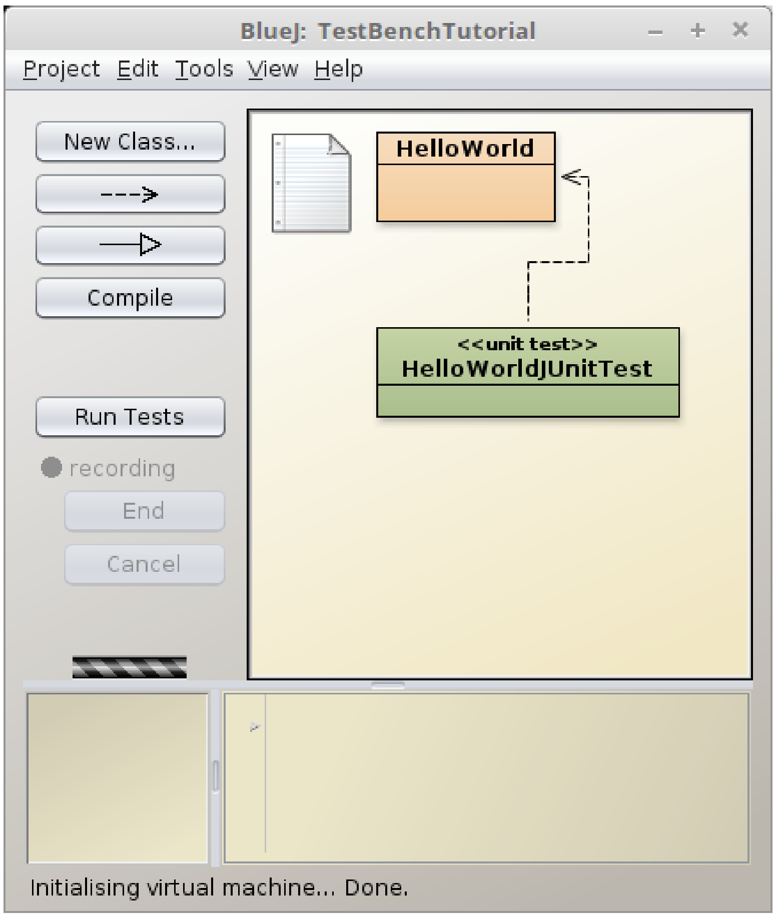
- Running a `JUnit` test bench on a `Java` program with a successful outcome.
- Introducing a single error into the `Java` program, then using the `JUnit` test bench to assist in identifying and correcting the error.

The `JUnit` Test Bench Framework

- `JUnit` is a simple framework that allows programmers to write customized and repeatable software tests. You can learn more about it by visiting the following website: `http://junit.org`
- Students will not be required to write `JUnit` test bench code. I will provide all of the testing code that will be used in this course.

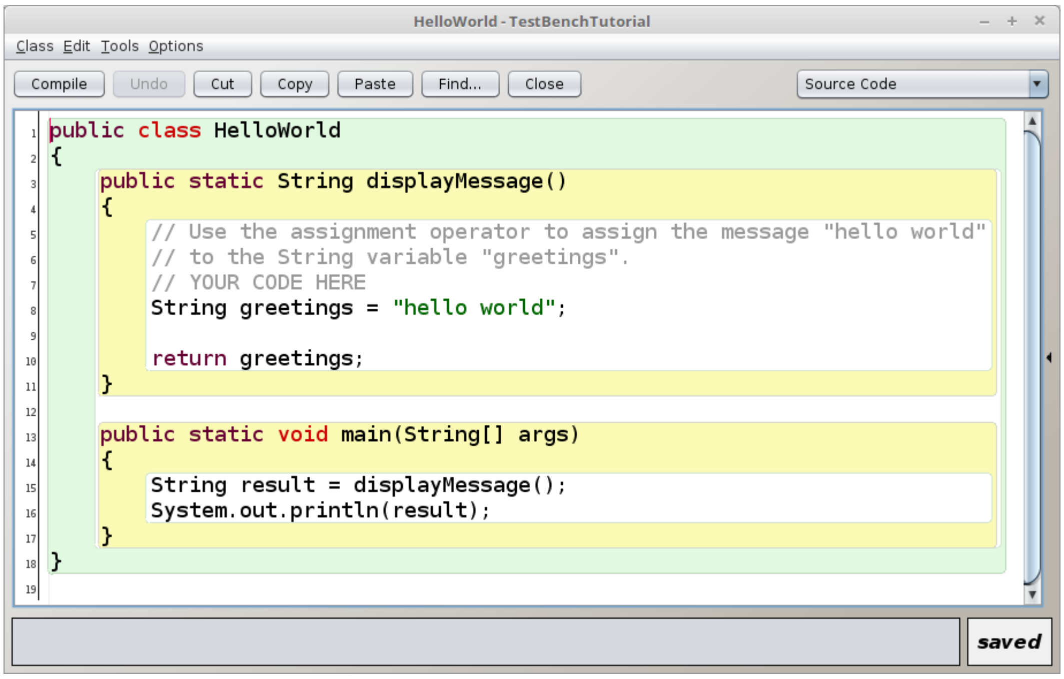
Running a Successful `JUnit` Test in BlueJ

- The following screenshot shows a BlueJ project which contains 2 files, `HelloWorld.java` and `HelloWorldJUnitTest.java`



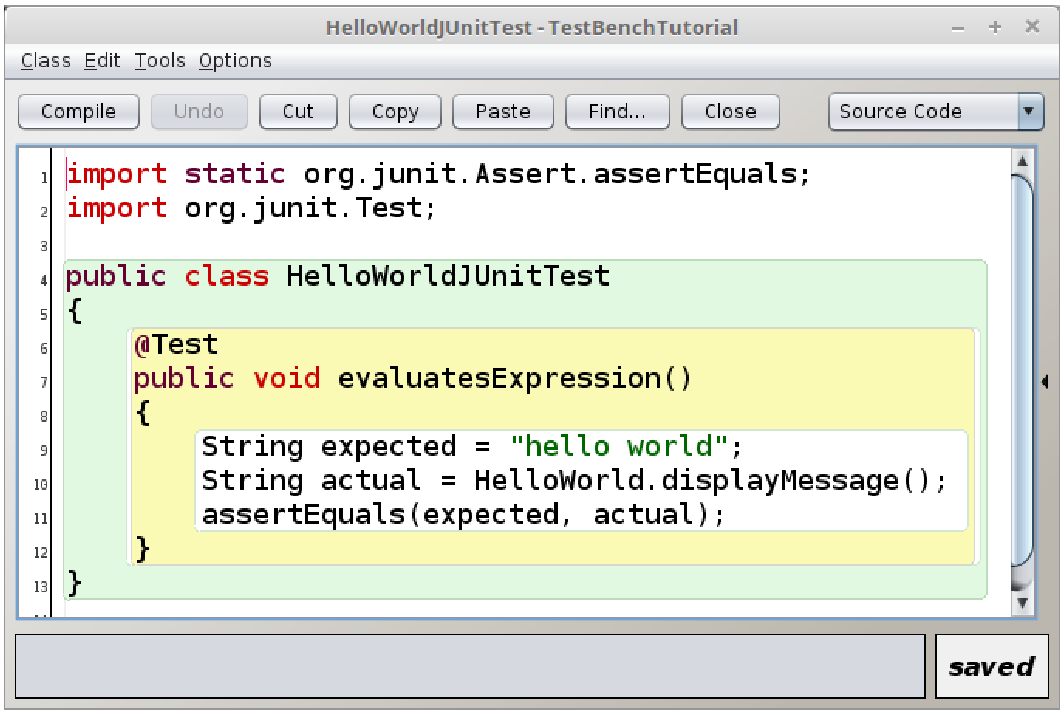
Initialising virtual machine... Done.

- The `HelloWorld.java` program is given in the following screenshot. There are quite a few details in the code that may not be immediately clear to you, but try to understand what is going on. Essentially, we have a function called `displayMessage()` which returns the `String` data in the variable `greetings`.


HelloWorld - TestBenchTutorial
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
saved

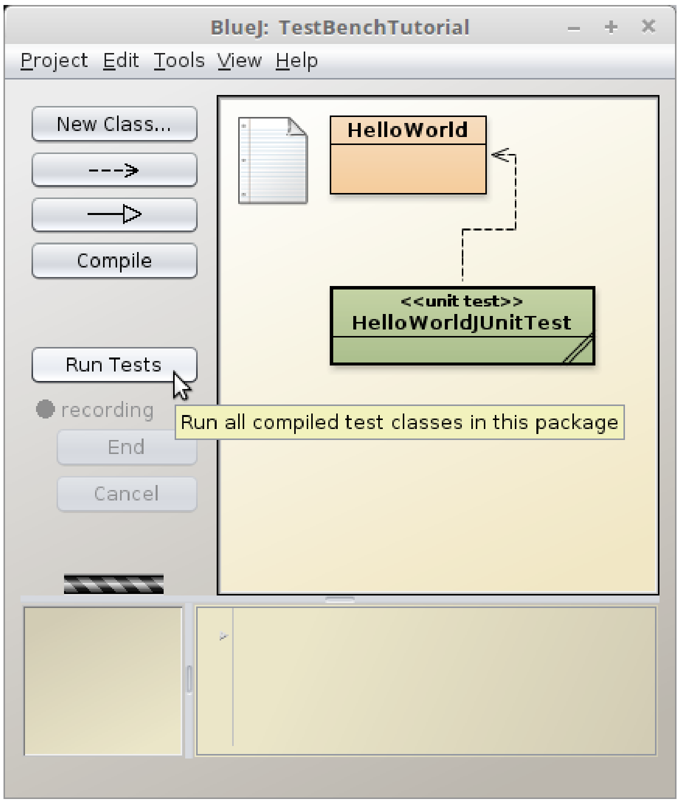
```
public class HelloWorld
{
    public static String displayMessage()
    {
        // Use the assignment operator to assign the message "hello world"
        // to the String variable "greetings".
        // YOUR CODE HERE
        String greetings = "hello world";

        return greetings;
    }

    public static void main(String[] args)
    {
        String result = displayMessage();
        System.out.println(result);
    }
}
```

- The following screenshot is the `JUnit` test bench code contained in the file `HelloWorldJUnitTest.java`
- This code is comparing the output of the `displayMessage()` function to the `String` data that it expects to receive. Students are welcome to open and examine the `JUnit` test bench code, but they will not be required to write it.

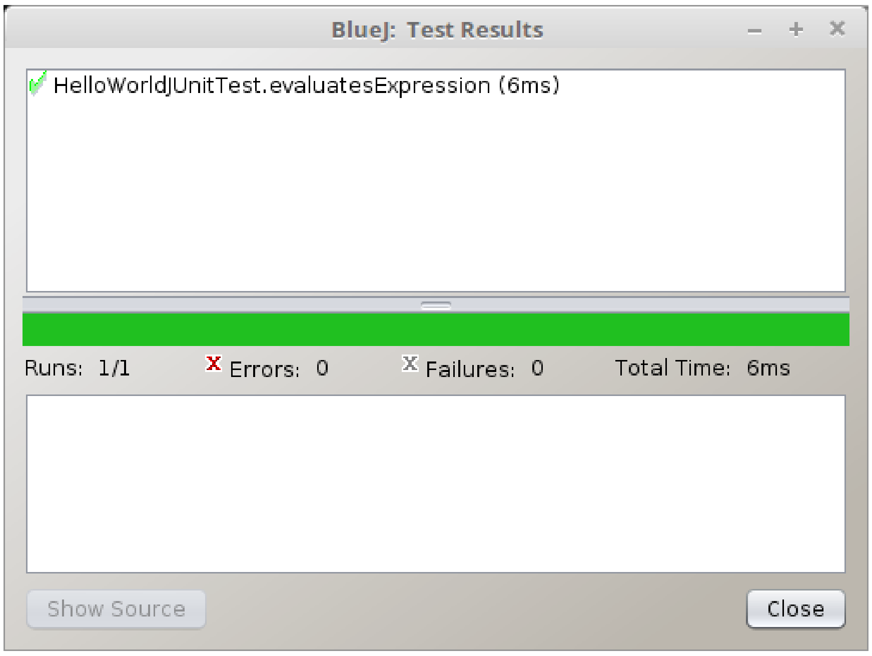

HelloWorldJUnitTest - TestBenchTutorial
Class Edit Tools Options
Compile Undo Cut Copy Paste Find... Close
Source Code
saved

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

public class HelloWorldJUnitTest
{
    @Test
    public void evaluatesExpression()
    {
        String expected = "hello world";
        String actual = HelloWorld.displayMessage();
        assertEquals(expected, actual);
    }
}
```

- In order to run the `JUnit` test bench, simply click on the `Run Tests` button. I have indicated it in the below screenshot with my cursor.



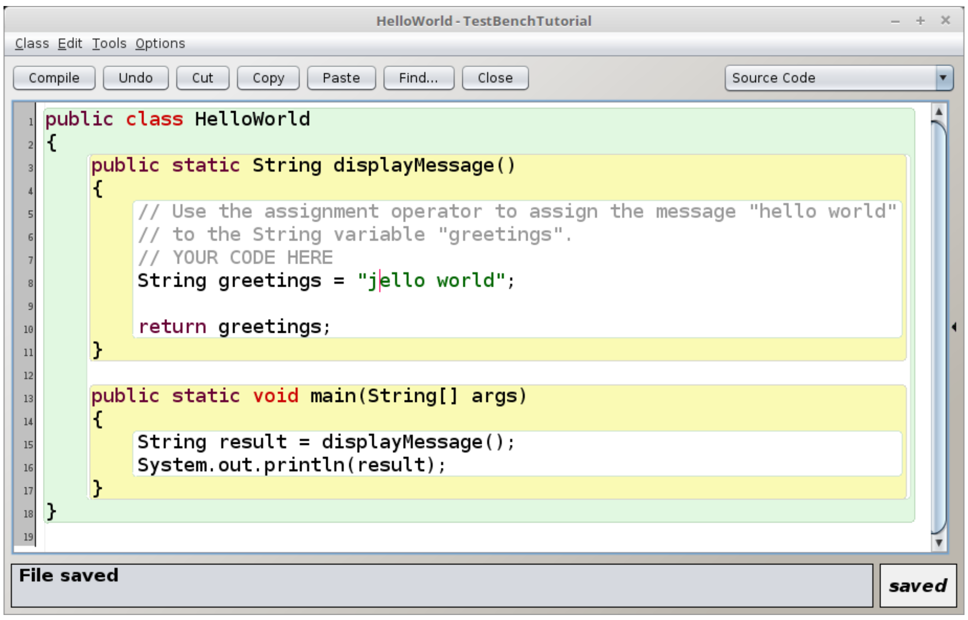
- Immediately after clicking on the `Run Tests` button, you should see a `BlueJ: Test Results` window appear.
- Since this in an example of a successful test, you should see a green bar appear. Also, the specific function that was tested has a green checkmark in front of it. Click on the `Close` button to exit.



```
✓ HelloWorldJUnitTest.evaluatesExpression (6ms)

Runs: 1/1    Errors: 0    Failures: 0    Total Time: 6ms
```

Running JUnit on a Java Program with an Error Present

- Now, we are going to use JUnit to help detect and correct an error which is present in the Java program. This is part of a process called *debugging*.
- Note that JUnit won't correct your errors for you, but it can help to locate them.
- In the following screenshot, I have purposefully introduced an error into HelloWorld.java
- Specifically, I have altered the string hello world to become jello world by turning the h into a j.

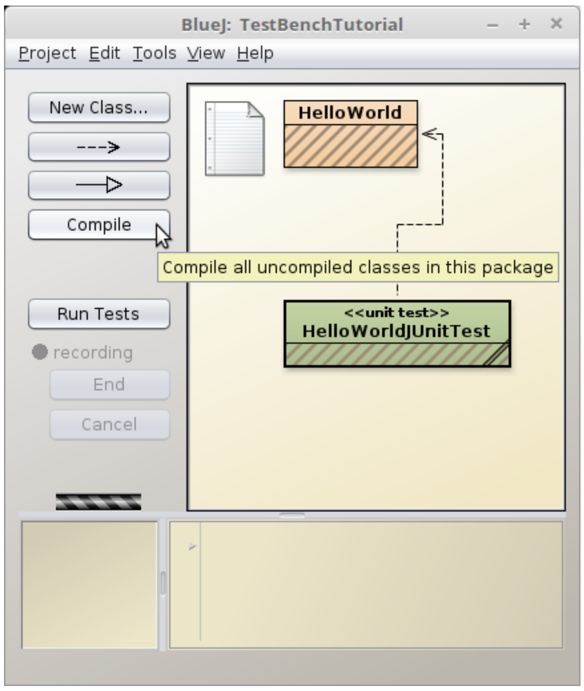


```
public class HelloWorld
{
    public static String displayMessage()
    {
        // Use the assignment operator to assign the message "hello world"
        // to the String variable "greetings".
        // YOUR CODE HERE
        String greetings = "jello world";

        return greetings;
    }

    public static void main(String[] args)
    {
        String result = displayMessage();
        System.out.println(result);
    }
}
```

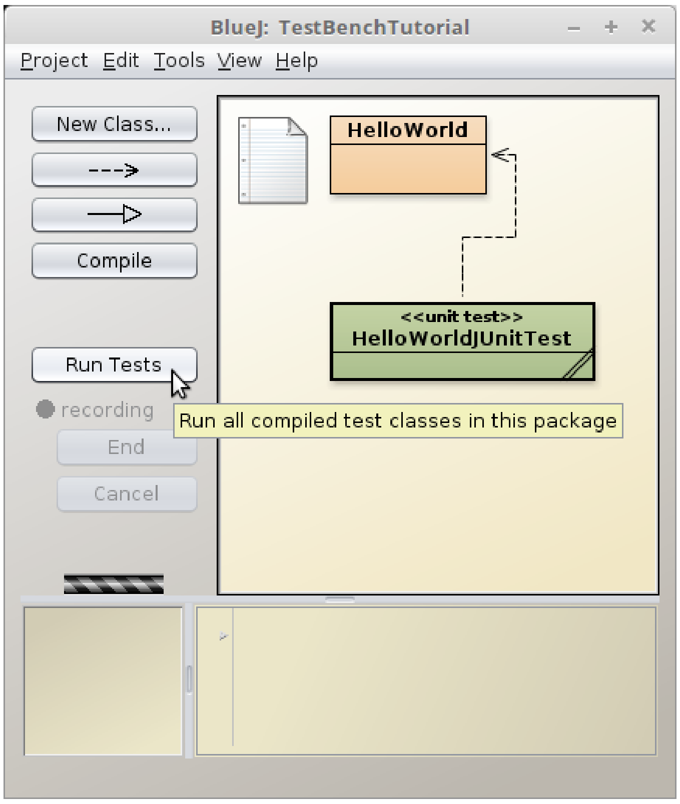
File saved

saved

- The next step is to compile the updated program. Go back to the BlueJ project window, and click on the Compile button.

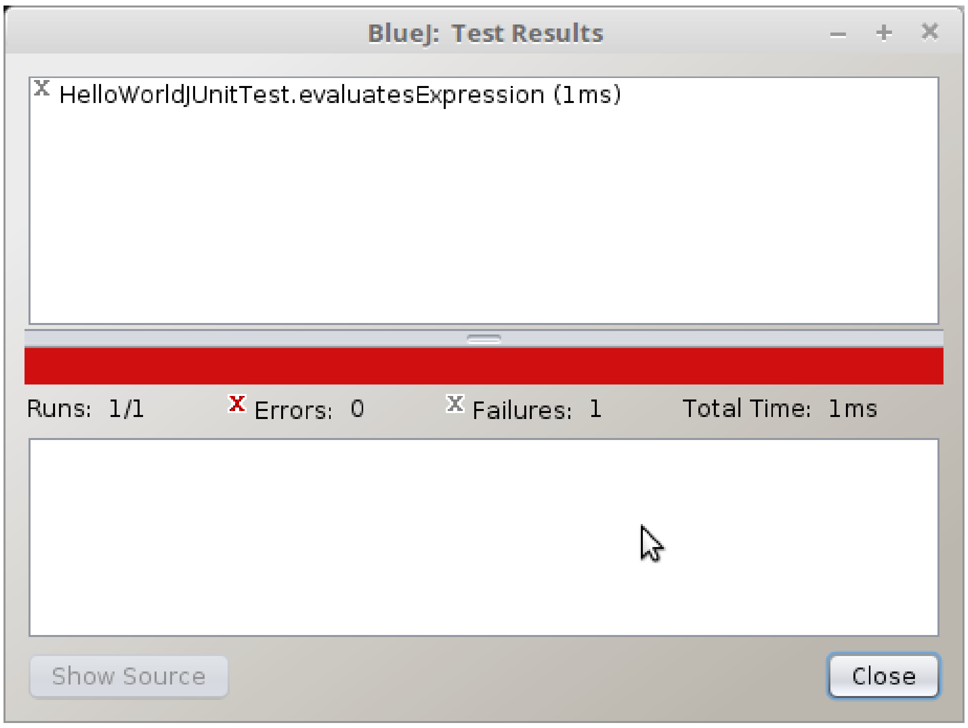


- Then, we must run the `JUnit` test bench on this updated code. Click on the `Run Tests` button to perform this action.



Run all compiled test classes in this package

- Now, we can see what happens when there is an error in our code. The `JUnit` test bench framework indicates this with a red bar, and it places an "X" in front of the test that failed.



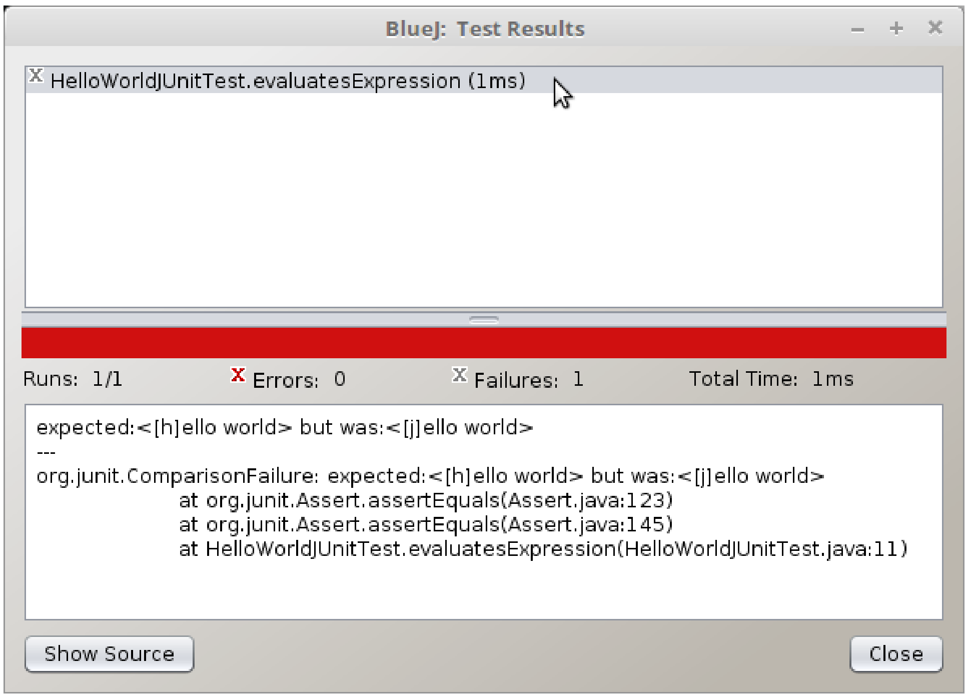
X HelloWorldJUnitTest.evaluatesExpression (1ms)

Runs: 1/1   X Errors: 0   X Failures: 1   Total Time: 1ms

- Click on the `HelloWorldJUnitTest.evaluatesExpression` element to discover more information about it.
- In the bottom panel, a small hint appears that could help us determine the problem:

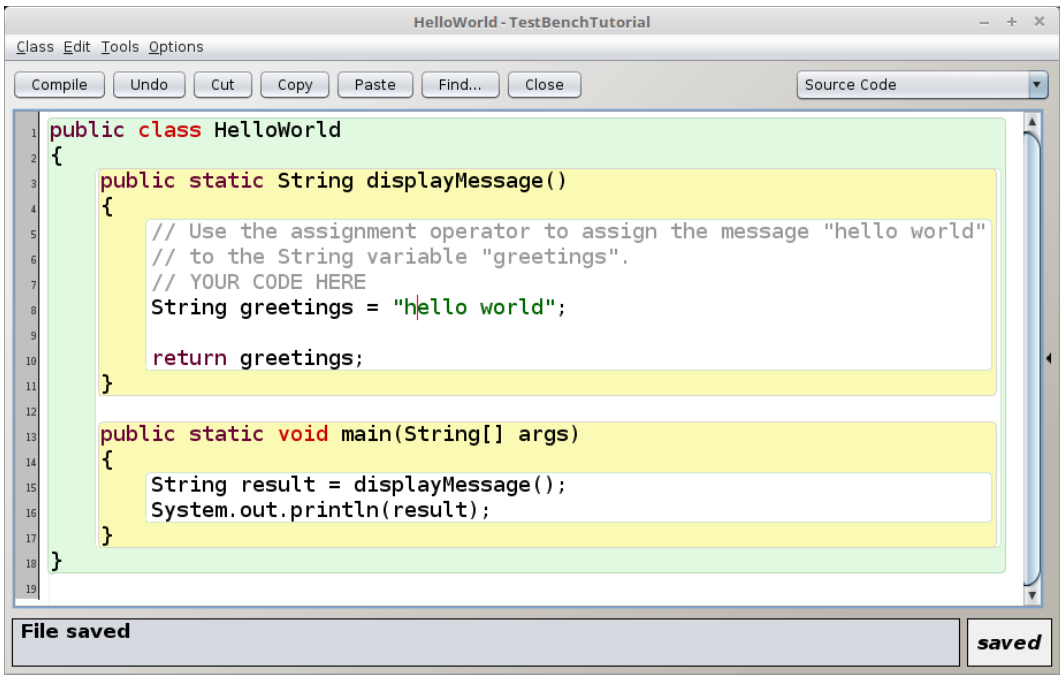
  ```
  expected:<[h]ello world> but was:<[j]ello world>
  ```

- `JUnit` did indeed recognize that the `h` had been changed to a `j`.



```
expected:<[h]ello world> but was:<[j]ello world>
---
org.junit.ComparisonFailure: expected:<[h]ello world> but was:<[j]ello world>
        at org.junit.Assert.assertEquals(Assert.java:123)
        at org.junit.Assert.assertEquals(Assert.java:145)
        at HelloWorldJUnitTest.evaluatesExpression(HelloWorldJUnitTest.java:11)
```

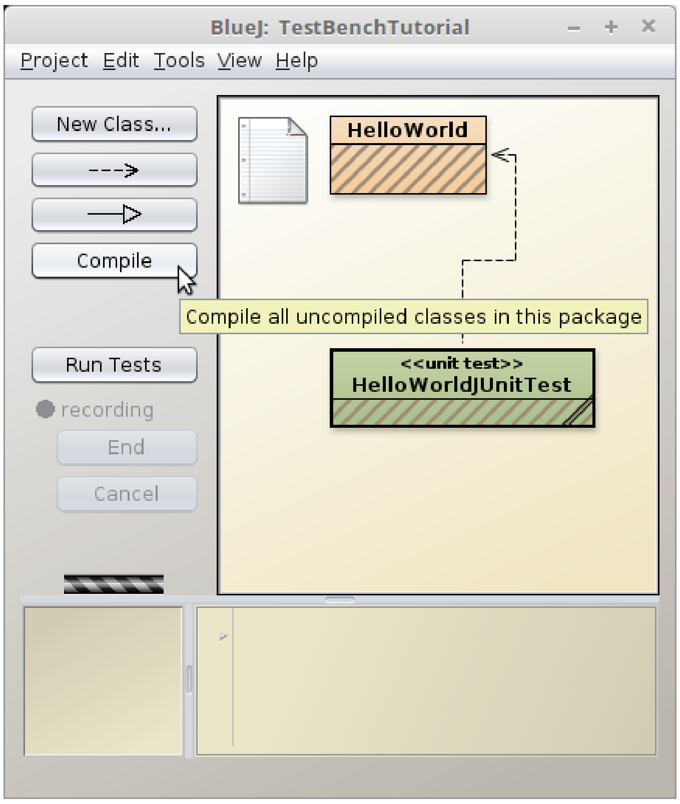
- This means that we must go back to `HelloWorld.java` and correct this error. The following screenshot shows that I have changed the `j` back to an `h`.



```java
public class HelloWorld
{
    public static String displayMessage()
    {
        // Use the assignment operator to assign the message "hello world"
        // to the String variable "greetings".
        // YOUR CODE HERE
        String greetings = "hello world";

        return greetings;
    }

    public static void main(String[] args)
    {
        String result = displayMessage();
        System.out.println(result);
    }
}
```

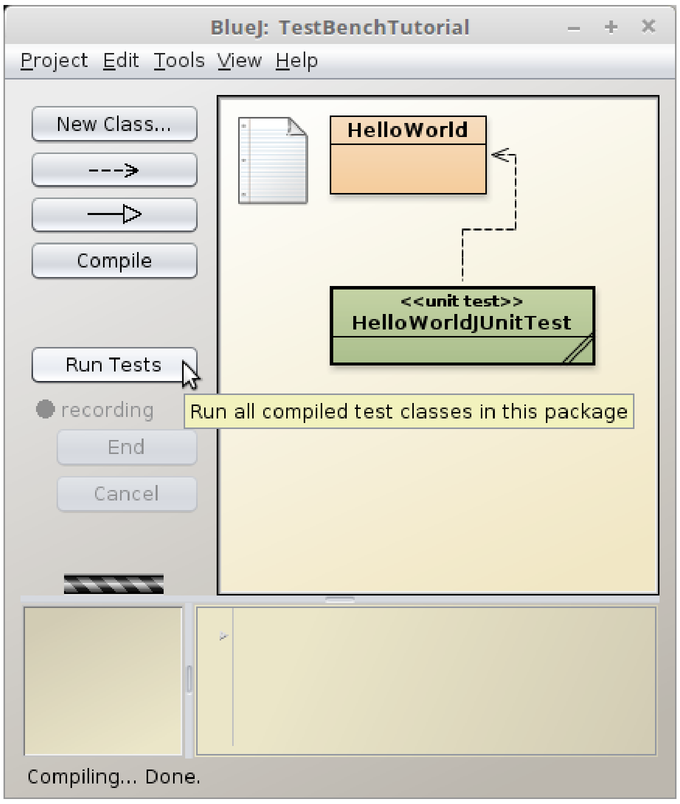
File saved

- Then, I will compile the `HelloWorld.java` program by pressing the `Compile` button.



Compile all uncompiled classes in this package

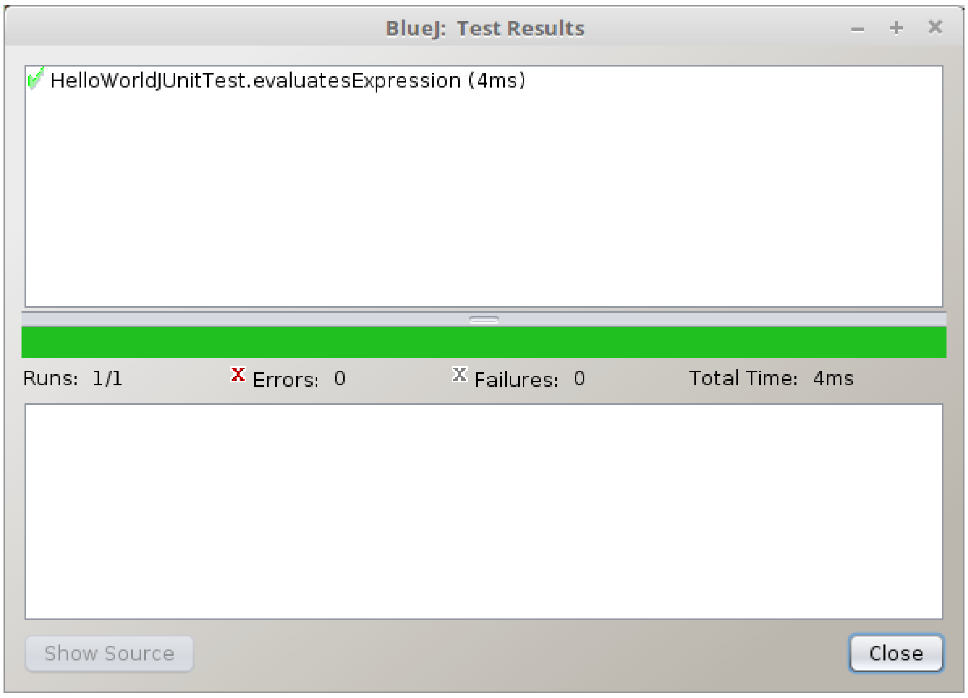
- Finally, I will click on the `Run Tests` button to run the `JUnit` test bench.



Run all compiled test classes in this package

Compiling... Done.

- You should see the `BlueJ: Test Results` window appear, and this time we have a green bar which indicates that all of our tests have passed. Click on the `Close` button to exit.



```
✓ HelloWorldJUnitTest.evaluatesExpression (4ms)

Runs: 1/1        Errors: 0        Failures: 0        Total Time: 4ms
```