**IT7320 – Assignment 3**

Table of Contents

[PART 1 – Mocking Object in Junit 2](#_Toc463820443)

[PART 2 – Configuring Tomcat and Jenkins for Continuous Integration 4](#_Toc463820444)

[Part 3: Bug tracking using Bugzilla 5](#_Toc463820445)

[References 6](#_Toc463820446)

[Appendix 7](#_Toc463820447)

[Pom.xml 7](#_Toc463820448)

[IOperator.java 7](#_Toc463820449)

[OperatorImplementation.java 8](#_Toc463820450)

[OperatorImplementationTest.java 8](#_Toc463820451)

[Index.html 10](#_Toc463820452)

[Figure 1: Operation methods 2](#_Toc463820453)

[Figure 2: Tests success 3](#_Toc463820454)

[Figure 3: Git 4](#_Toc463820455)

[Figure 4: GitHub integration 4](#_Toc463820456)

[Figure 5: Jenkins integration 4](#_Toc463820457)

[Figure 6: list of possible issue duplicates 5](#_Toc463820458)

[Figure 7: Adding details of new bug 5](#_Toc463820459)

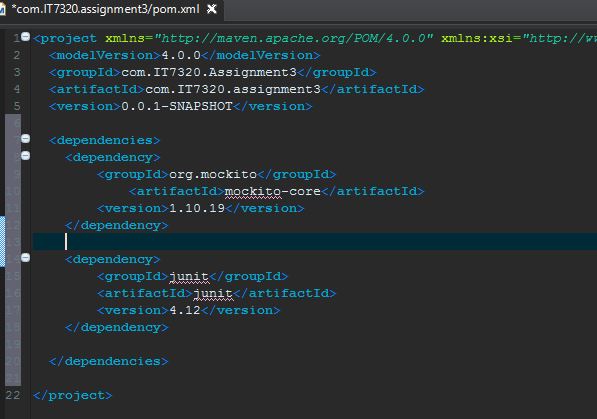
[Figure 8: dashboard 6](#_Toc463820460)

[Figure 9: Bug details 6](#_Toc463820461)

[Figure 10: details entered about bug 6](#_Toc463820462)

# PART 1 – Mocking Object in Junit

For this part of the assignment we are to investigate how to create a Maven project using a Java IDE, and add to it Mockito and Junit dependencies in the ‘pom.xml’ file.



After creating a new Maven project, a Java interface is created in the ‘src/main/java source’ package. In it we create 4 methods to perform basic arithmetic operations, and passing to them 2 variables. The 4 methods are:

* add (int x, int y)
* subtract (int x, int y)
* multiply (int x, int y)
* divide (int x, int y)

We then need an implementation class to define those 4 methods, created in the same package. An object of the interface is created and returns values for each method, as shown below.

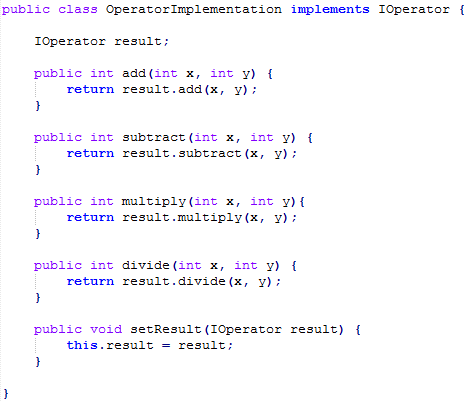


Figure : Operation methods

To ensure that the methods, and therefore the Operation interface and Implementation class, work as intended, we design a Junit test case in the ‘src/test/java’ folder. An object is created and values assigned to the 2 variables passes to the methods.

Using the ‘assertEquals’ method, we try to verify that the interface returns the values expected from the operation. Unfortunately, it will not work, as a mock object is to be created and compared to the tested one. For that to work, the ‘assertSame’ method is to be used instead.

After creating a mock object that returns values according to which operation method is executed, we simply compare that the actual values returned by the test object are the same as that of the mock object.

If the objects and methods are properly created and instantiated, Junit returns a pass for all test cases, as shown below:

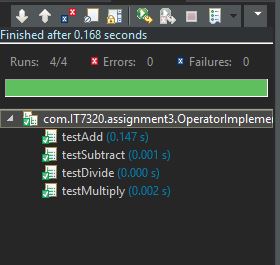


Figure : Tests success

# PART 2 – Configuring Tomcat and Jenkins for Continuous Integration

Apache Tomcat is an open-source web server which implements several Java EE, and provides a "pure Java" HTTP web server environment for Java code to run in (What is Apache Tomcat?, 2016).

Jenkins on the other side ensures that continuous integration can be applied by linking it to GitHub. Therefore, Git and GitHub also have to be implemented.

After creating a simple dynamic web page, and configuring Tomcat, we can run the program through the web browser (e.g. http://localhost:8080/index.html).

After copying Tomcat to the project folder, and writing the script of the dynamic web page, the server can be started to ensure that the webpage works properly.

We can then implement Git in the project, and subsequently GitHub. After making sure that the project is up and running on GitHub, we can implement Jenkins by copying the appropriate file to the Tomcat folder. We can then create an admin account using the given password, and integrate the designed web page to Jenkins for continuous integration.

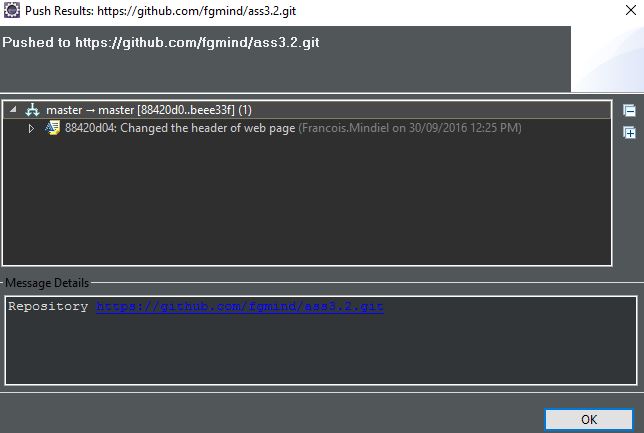


Figure : Git

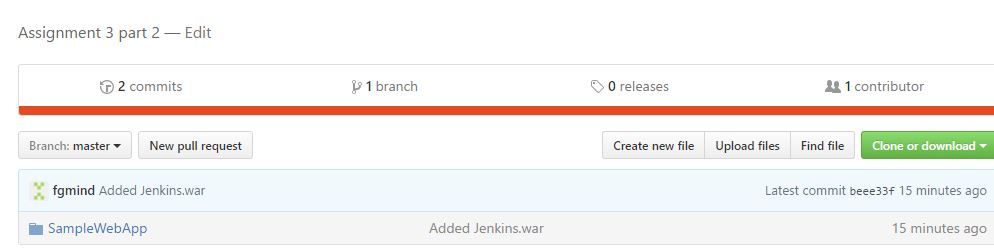


Figure : GitHub integration



Figure : Jenkins integration

# Part 3: Bug tracking using Bugzilla

Bugzilla is a web-based general-purpose bug tracker and testing tool originally developed and used by the Mozilla project. It is open-sourced and is used by a lot of organisations for reporting and tracking bug (Bugzilla, 2016). There is a wide community that can help in reviewing the bugs, and find possible solutions to resolve them.

Using Firefox web browser, we identified a possible bug, and after searching the database from Bugzilla to ensure that it didn’t already exist, we would report it.

The bug in question is the installation of an add-on to the web browser to download YouTube videos onto the computer.

In order to report bugs, an account has to first be created.

We just have to click on the “File a Bug” button, and after entering a name for the bug, Bugzilla displays a list of possible duplicate bug reporting. Since the bug is not already reported, we just click on the “My issue is not listed” button.

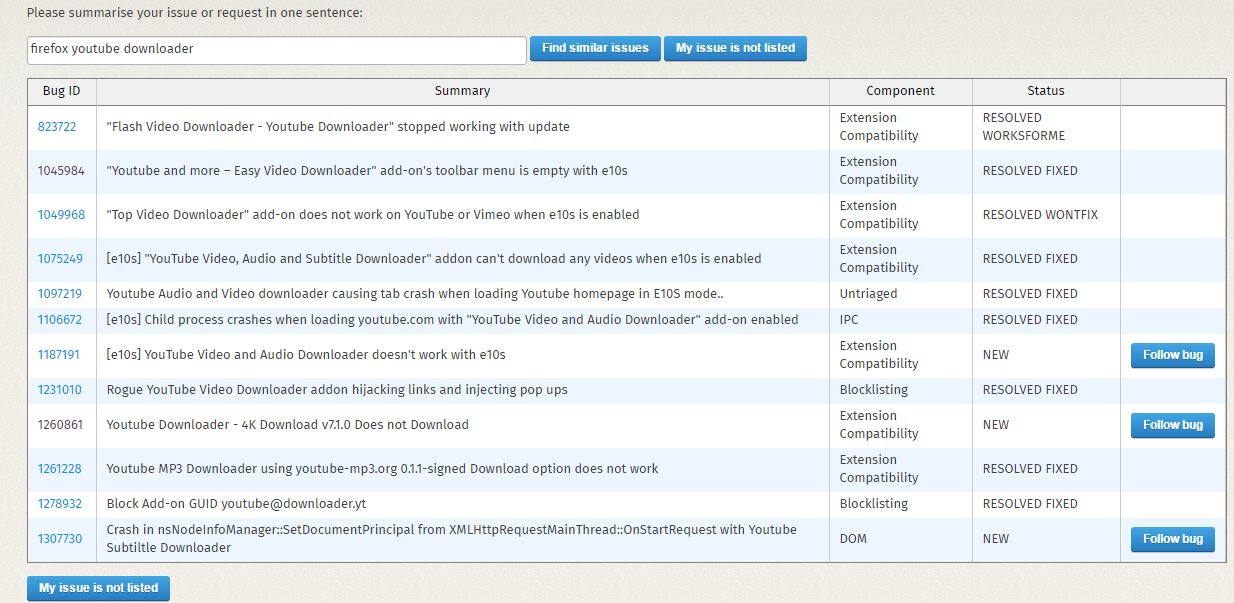


Figure : list of possible issue duplicates

We can then add the details concerning the bug, including how to duplicate the bug and what is the expected bug-free outcome. A file can be attached for screenshots or video of mistake. We can also add the version of Firefox, the system platform (32/64 bit) and operating system used (e.g. Windows 10).

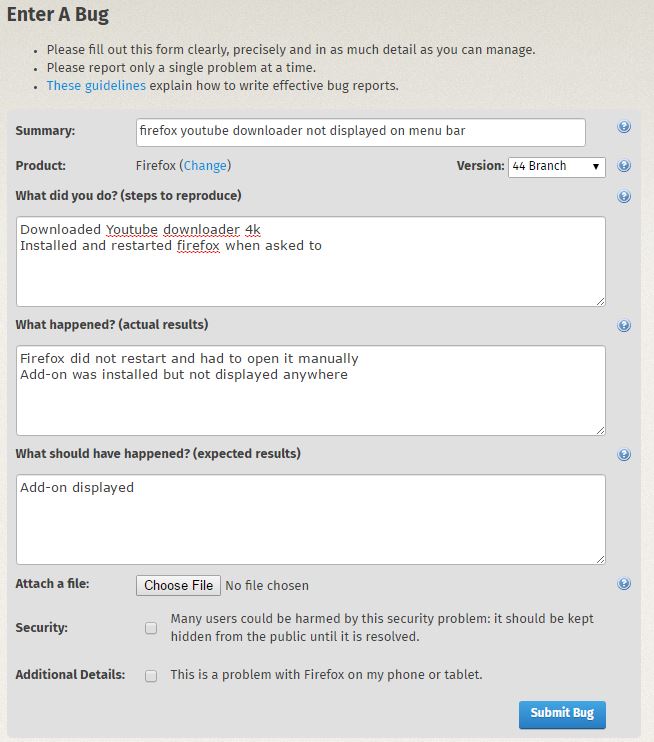


Figure : Adding details of new bug

After submission of the bug, we can see the bug in the user dashboard. The bug is assigned a unique identifier, as well as its status and summary.



Figure : dashboard

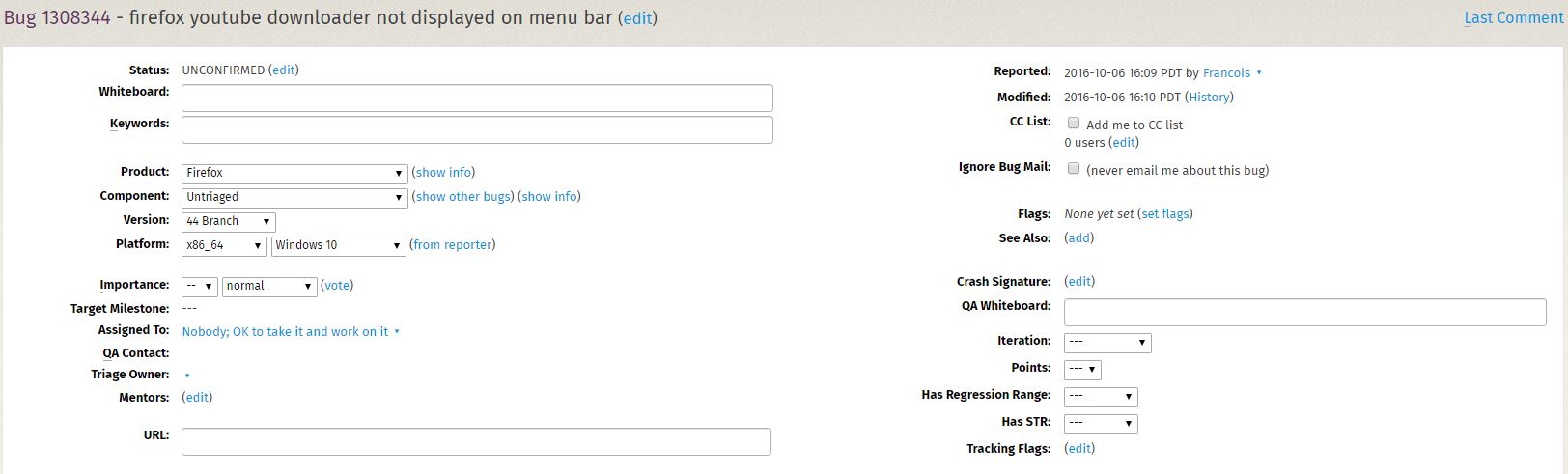


Figure : Bug details

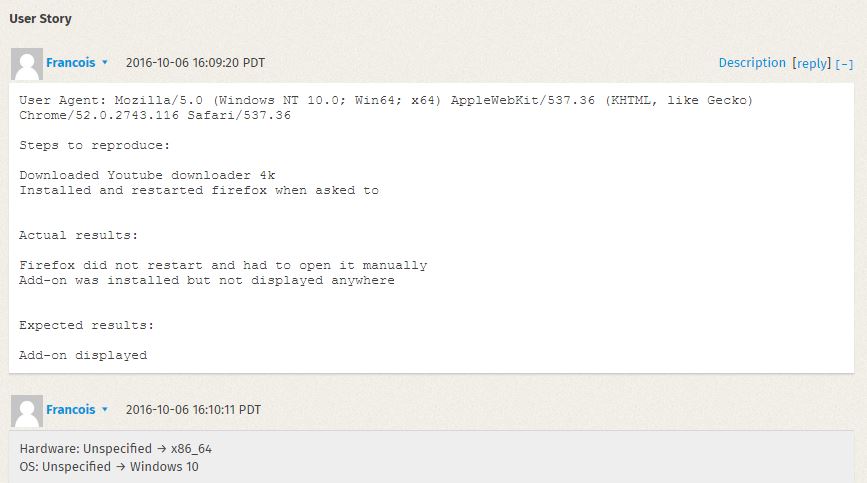


Figure : details entered about bug

# References

*Bugzilla*. (2016, 10 07). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Bugzilla

*What is Apache Tomcat?* (2016, 10 9). Retrieved from Quora: https://www.quora.com/What-is-Apache-Tomcat

# Appendix

## Pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.IT7320.Assignment3</groupId>

<artifactId>com.IT7320.assignment3</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>1.10.19</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

</dependencies>

</project>

## IOperator.java

package com.IT7320.assignment3;

public interface IOperator {

public int add (int x, int y);

public int subtract ( int x, int y);

public int multiply (int x, int y);

public int divide (int x, int y);

}

## OperatorImplementation.java

package com.IT7320.assignment3;

public class OperatorImplementation implements IOperator {

IOperator result;

public int add(int x, int y) {

return result.add(x, y);

}

public int subtract(int x, int y) {

return result.subtract(x, y);

}

public int multiply(int x, int y){

return result.multiply(x, y);

}

public int divide(int x, int y) {

return result.divide(x, y);

}

public void setResult(IOperator result) {

this.result = result;

}

}

## OperatorImplementationTest.java

package com.IT7320.assignment3;

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.mockito.Mockito.\*;

public class OperatorImplementationTest {

OperatorImplementation test;

int x,y;

@Before

public void setUp() throws Exception {

x = 10;

y = 5;

IOperator mock = mock(IOperator.class);

when(mock.add(x, y)).thenReturn(x+y);

when(mock.subtract(x, y)).thenReturn(x-y);

when(mock.multiply(x, y)).thenReturn(x\*y);

when(mock.divide(x, y)).thenReturn(x/y);

test = new OperatorImplementation();

test.setResult(mock);

}

@After

public void tearDown() throws Exception {

test = null;

}

@Test

public void testAdd() {

assertSame(15, test.add(x, y));

}

@Test

public void testSubtract(){

assertSame(5, test.subtract(x, y));

}

@Test

public void testMultiply(){

assertSame(50, test.multiply(x, y));

}

@Test

public void testDivide(){

assertSame(2, test.divide(x, y));

}

}

## Index.html

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function divide()

{

var n1 = document.getElementById('num1').value;

var n2 = document.getElementById('num2').value;

$x = n1 / n2;

document.write($x);

}

function multiply()

{

var n1 = document.getElementById('num1').value;

var n2 = document.getElementById('num2').value;

$x = n1 \* n2;

document.write($x);

}

</script>

<meta charset="ISO-8859-1">

<title>Java Project using Tomcat</title>

</head>

<body>

<h1>

This is a simple calculator that multiplies or divides 2 numbers!

</h1>

<form>

Number 1: &nbsp;&nbsp;<input type="text" id="num1"><br/>

Number 2: &nbsp;&nbsp;<input type="text" id="num2"><br/>

<input type="button" value=" / " onclick="divide()">&nbsp;&nbsp;

<input type="button" value=" \* " onclick="multiply()">

</form>

</body>

</html>