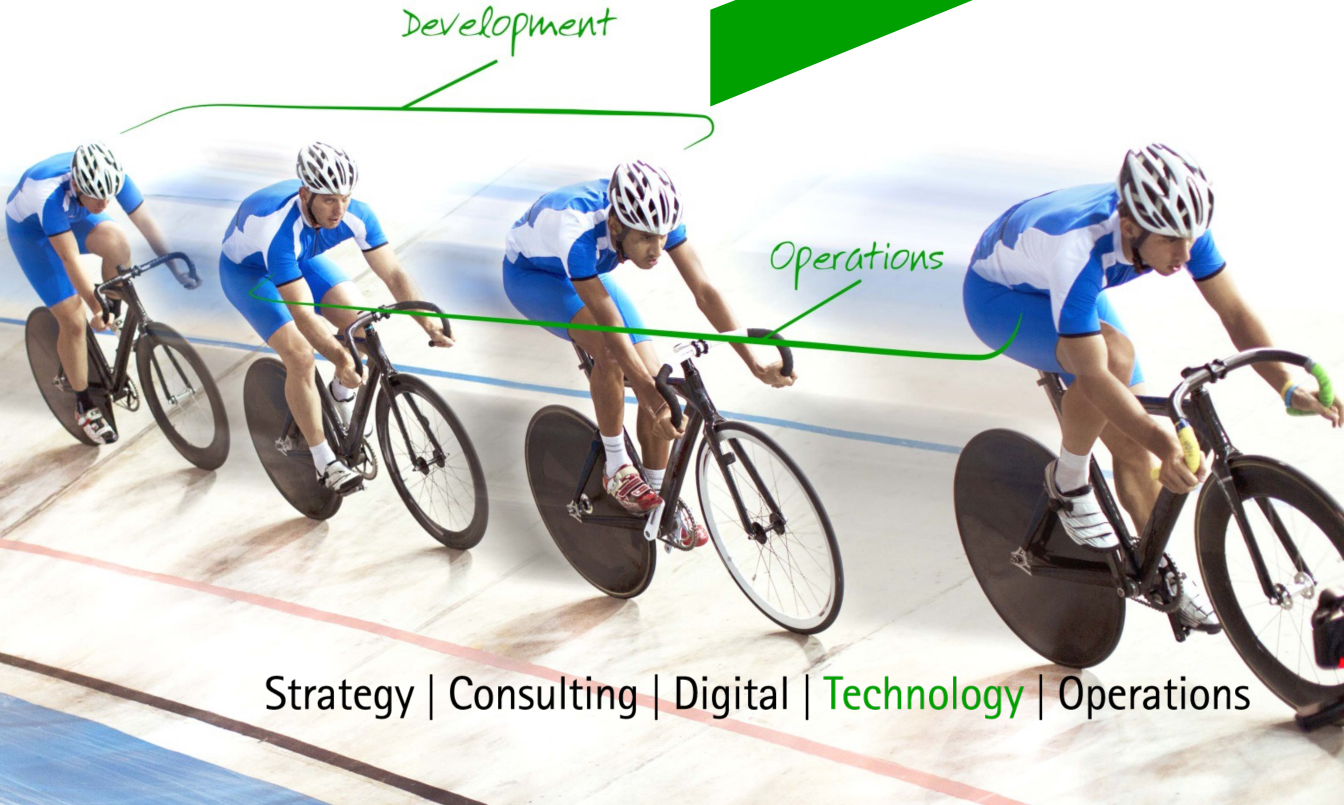
accenture technology

Module 4: Code Quality

Labs

High performance. Delivered.



Strategy | Consulting | Digital | Technology | Operations

TDD LAB

Lab-1: String Calculator

String Calculator Context

Requirement:

Create a simple String calculator that can take 0, 1 or 2 numbers separated by comma (,), for example “” or “1” or “1,2”

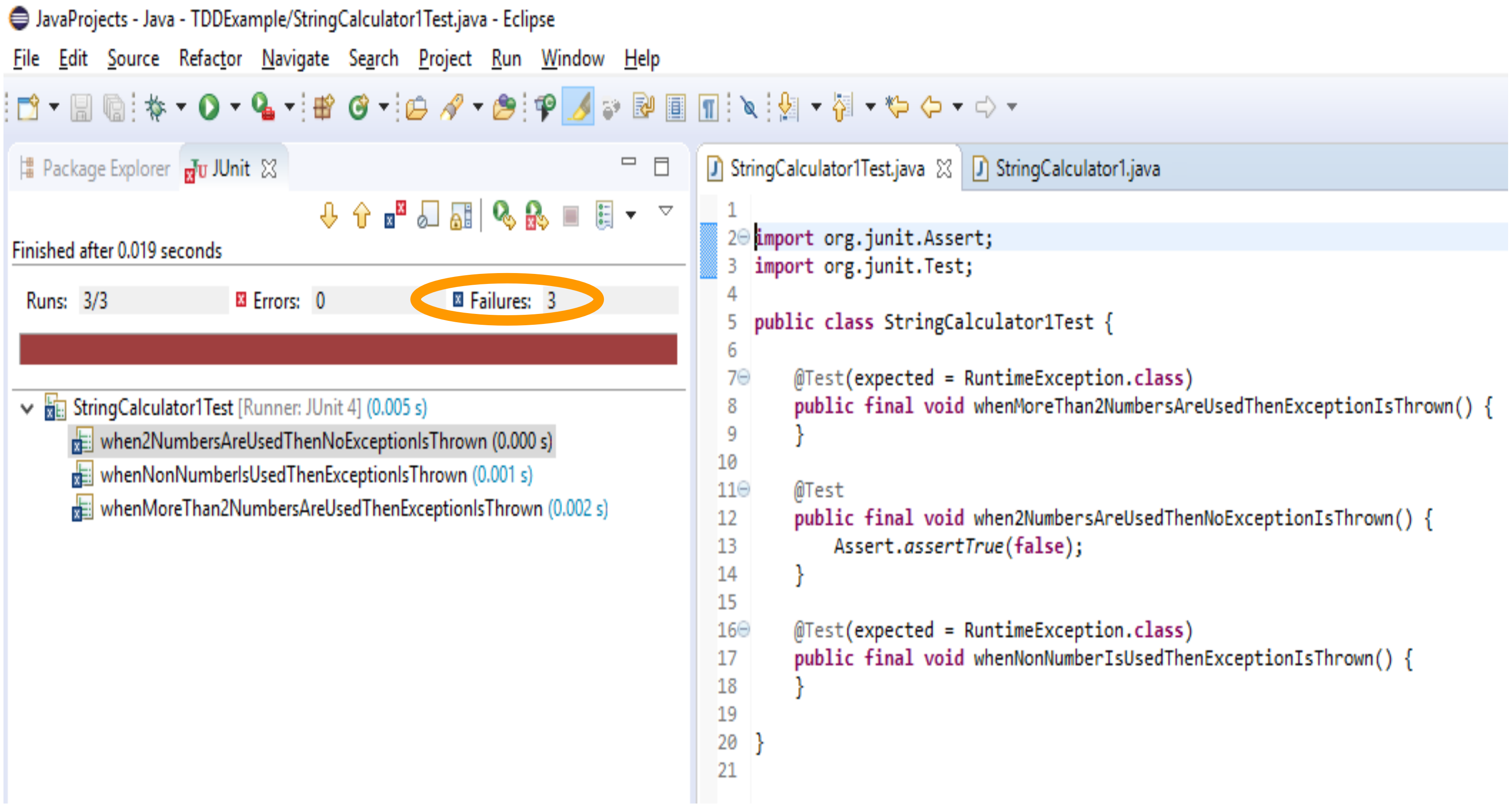
Let’s write the code that fulfills all the above requirements.

Step-1: Write test cases to fail first, in this example we can write 3 test cases as described in next slide.

https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/

Lab-1: String Calculator

Step 1 – Write Test Cases to fail


JavaProjects - Java - TDDExample/StringCalculator1Test.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
StringCalculator1Test.java
StringCalculator1.java

Finished after 0.019 seconds

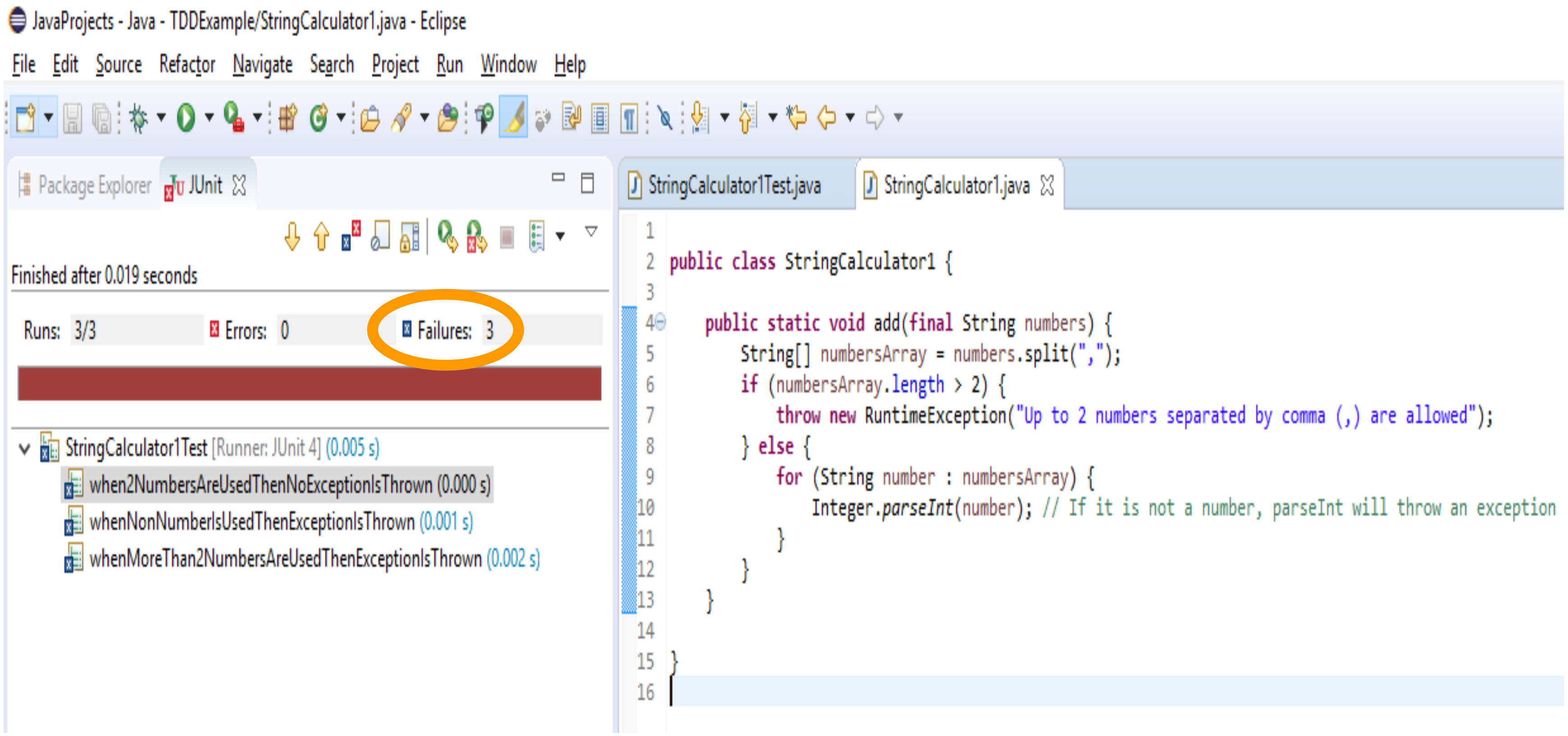
Runs: 3/3 Errors: 0 Failures: 3

StringCalculator1Test [Runner: JUnit 4] (0.005 s)
- when2NumbersAreUsedThenNoExceptionIsThrown (0.000 s)
- whenNonNumberIsUsedThenExceptionIsThrown (0.001 s)
- whenMoreThan2NumbersAreUsedThenExceptionIsThrown (0.002 s)

```java

import org.junit.Assert;
import org.junit.Test;

public class StringCalculator1Test {

    @Test(expected = RuntimeException.class)
    public final void whenMoreThan2NumbersAreUsedThenExceptionIsThrown() {
    }

    @Test
    public final void when2NumbersAreUsedThenNoExceptionIsThrown() {
        Assert.assertTrue(false);
    }

    @Test(expected = RuntimeException.class)
    public final void whenNonNumberIsUsedThenExceptionIsThrown() {
    }

}
```

Lab-1: String Calculator

Step 2 – Write Code to fulfill requirement – test cases will fail again


JavaProjects - Java - TDDExample/StringCalculator1.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
StringCalculator1Test.java
StringCalculator1.java

Finished after 0.019 seconds

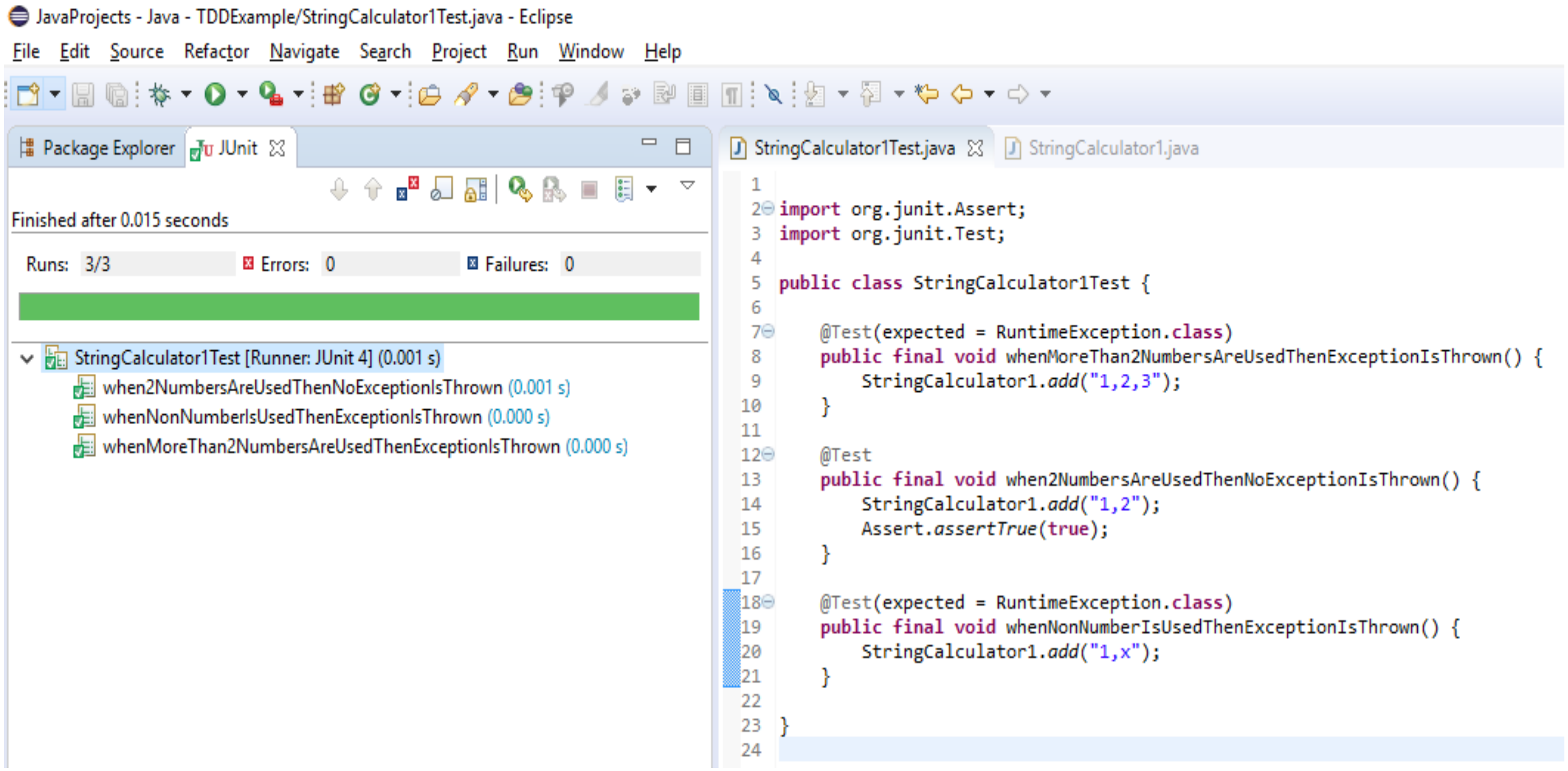
Runs: 3/3 Errors: 0 Failures: 3

```
StringCalculator1Test [Runner: JUnit 4] (0.005 s)
    when2NumbersAreUsedThenNoExceptionIsThrown (0.000 s)
    whenNonNumberIsUsedThenExceptionIsThrown (0.001 s)
    whenMoreThan2NumbersAreUsedThenExceptionIsThrown (0.002 s)
```

```java

public class StringCalculator1 {

    public static void add(final String numbers) {
        String[] numbersArray = numbers.split(",");
        if (numbersArray.length > 2) {
            throw new RuntimeException("Up to 2 numbers separated by comma (,) are allowed");
        } else {
            for (String number : numbersArray) {
                Integer.parseInt(number); // If it is not a number, parseInt will throw an exception
            }
        }
    }

}
```

Lab-1: String Calculator

Step 3 – Write code to pass test


JavaProjects - Java - TDDExample/StringCalculator1Test.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JUnit
StringCalculator1Test.java
StringCalculator1.java

Finished after 0.015 seconds

Runs: 3/3 Errors: 0 Failures: 0

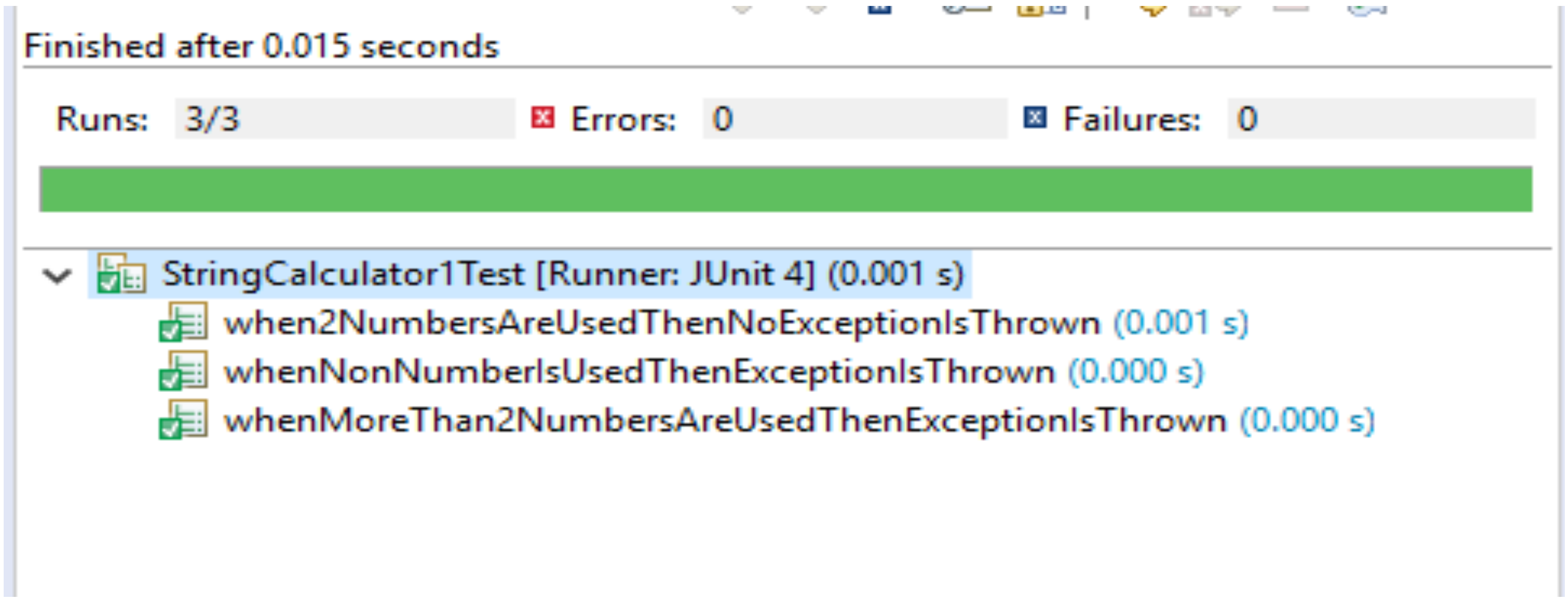
```
StringCalculator1Test [Runner: JUnit 4] (0.001 s)
    when2NumbersAreUsedThenNoExceptionIsThrown (0.001 s)
    whenNonNumberIsUsedThenExceptionIsThrown (0.000 s)
    whenMoreThan2NumbersAreUsedThenExceptionIsThrown (0.000 s)
```

```java

import org.junit.Assert;
import org.junit.Test;

public class StringCalculator1Test {

    @Test(expected = RuntimeException.class)
    public final void whenMoreThan2NumbersAreUsedThenExceptionIsThrown() {
        StringCalculator1.add("1,2,3");
    }

    @Test
    public final void when2NumbersAreUsedThenNoExceptionIsThrown() {
        StringCalculator1.add("1,2");
        Assert.assertTrue(true);
    }

    @Test(expected = RuntimeException.class)
    public final void whenNonNumberIsUsedThenExceptionIsThrown() {
        StringCalculator1.add("1,x");
    }

}
```

Lab-1: String Calculator

Output

Output is PASSED as shown below;



```
Finished after 0.015 seconds

Runs: 3/3    Errors: 0    Failures: 0

StringCalculator1Test [Runner: JUnit 4] (0.001 s)
    when2NumbersAreUsedThenNoExceptionIsThrown (0.001 s)
    whenNonNumberIsUsedThenExceptionIsThrown (0.000 s)
    whenMoreThan2NumbersAreUsedThenExceptionIsThrown (0.000 s)
```

LAB-2: TDD and Pair Programming(45 Mins)

Lab-2: TDD and Pair Programming

Implement test cases and refactor code for following requirements

Requirements for String Calculator

- *Requirement-1: The method can take 0, 1 or 2 numbers separated by comma (,). (Implemented in Lab-1)*
- Requirement-2: For an empty string the method will return 0
- Requirement-3: Method will return their sum of numbers
- Requirement-4: Allow the Add method to handle an unknown amount of numbers
- Requirement-5: Allow the Add method to handle new lines between numbers (instead of commas).
- Requirement-6: Support different delimiters
- Requirement-7: Negative numbers will throw an exception
- Requirement-8: Numbers bigger than 1000 should be ignored

Note: Some of the requirements are conflicting, you will be required to refactor the code and test cases accordingly.

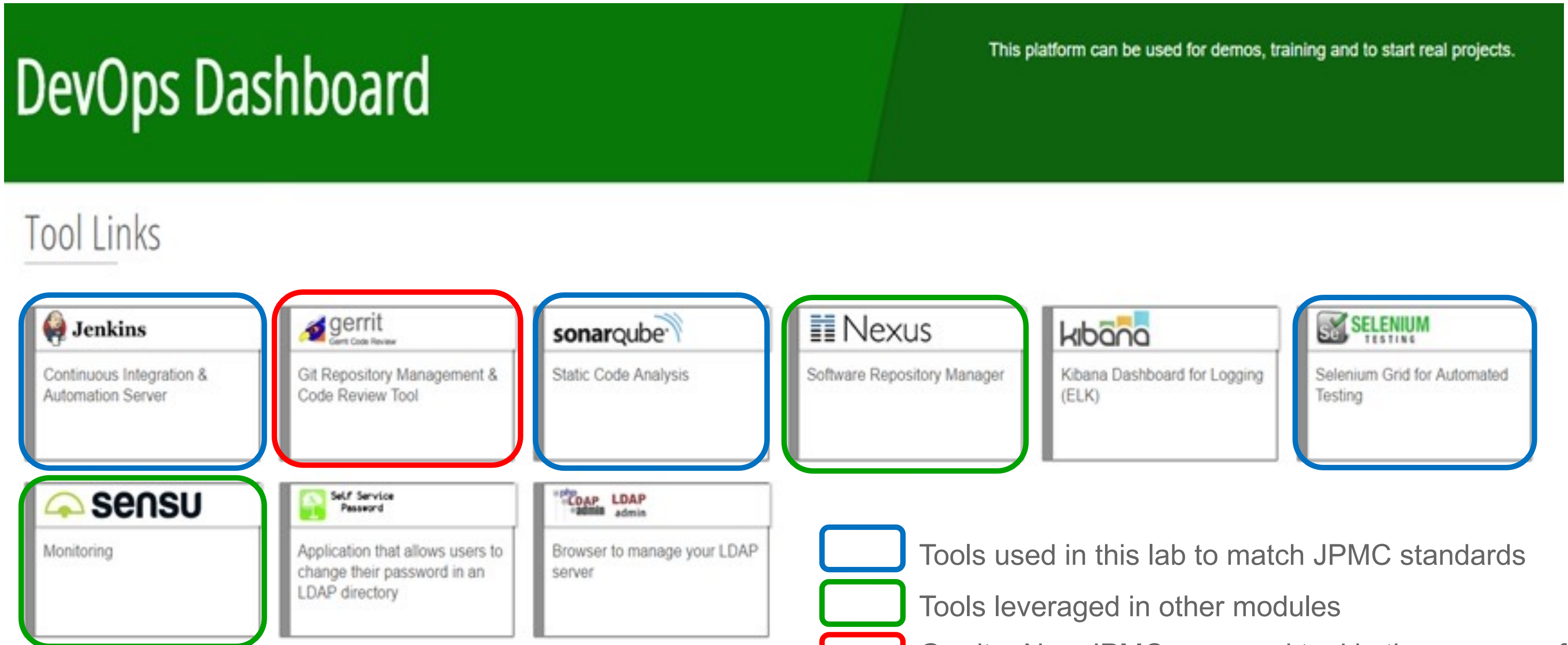
Lab reference : http://osherove.com/tdd-kata-1/

LAB-3: Observe Tools in Lab Environment (15 Mins)

Lab-3: Observe Tools

Step-1 – ADOP Core Platform

- Links to all the tools can be found on the NGINX homepage, http://<Server-Public-IP>/
- Replace <Server-Public-IP> with the public IP of the server with IP assigned to you.
- You will notice the screen below. Click the tool and log in with username/password provided to you



DevOps Dashboard

This platform can be used for demos, training and to start real projects.

Tool Links

Tool	Description
Jenkins	Continuous Integration & Automation Server
gerrit	Git Repository Management & Code Review Tool
sonarqube	Static Code Analysis
Nexus	Software Repository Manager
kibana	Kibana Dashboard for Logging (ELK)
SELENIUM TESTING	Selenium Grid for Automated Testing
sensu	Monitoring
Self Service Password	Application that allows users to change their password in an LDAP directory
LDAP admin	Browser to manage your LDAP server

🟦 Tools used in this lab to match JPMC standards

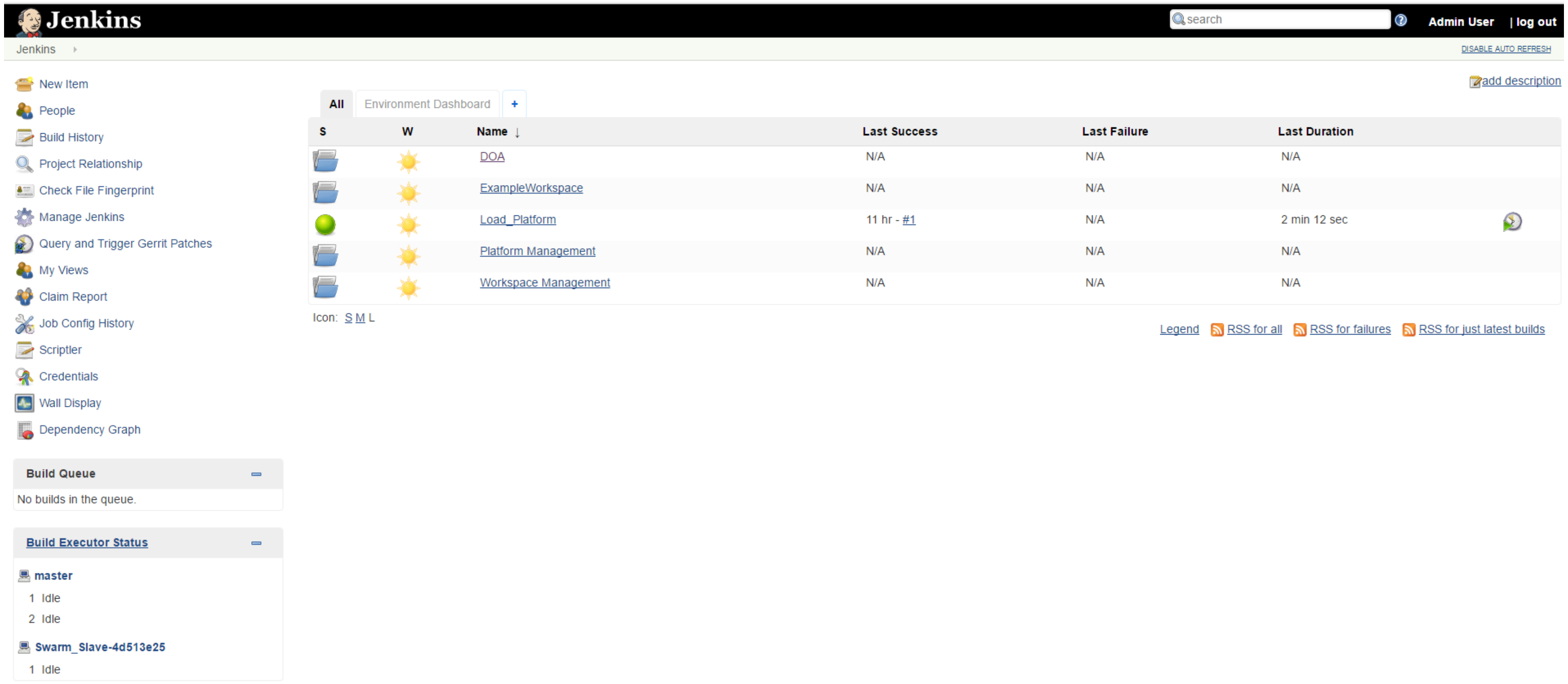
🟩 Tools leveraged in other modules

🟥 Gerrit – Non-JPMC approved tool in the process of being replaced by Bitbucket

Lab-3: Observe Tools

Step 2 - Jenkins

- Jenkins will be available from http://<Server-Public-IP>/jenkins
- Note this is not an empty Jenkins installation, it has been pre-built with plugins and jobs that we need for today.



Jenkins | search | Admin User | log out

Jenkins › DISABLE AUTO REFRESH

- New Item
- People
- Build History
- Project Relationship
- Check File Fingerprint
- Manage Jenkins
- Query and Trigger Gerrit Patches
- My Views
- Claim Report
- Job Config History
- Scriptler
- Credentials
- Wall Display
- Dependency Graph

Build Queue

No builds in the queue.

Build Executor Status

master
1 Idle
2 Idle

Swarm_Slave-4d513e25
1 Idle

add description

All | Environment Dashboard | +

S	W	Name ↓	Last Success	Last Failure	Last Duration
	☀	DOA	N/A	N/A	N/A
	☀	ExampleWorkspace	N/A	N/A	N/A
●	☀	Load_Platform	11 hr - #1	N/A	2 min 12 sec
	☀	Platform Management	N/A	N/A	N/A
	☀	Workspace Management	N/A	N/A	N/A

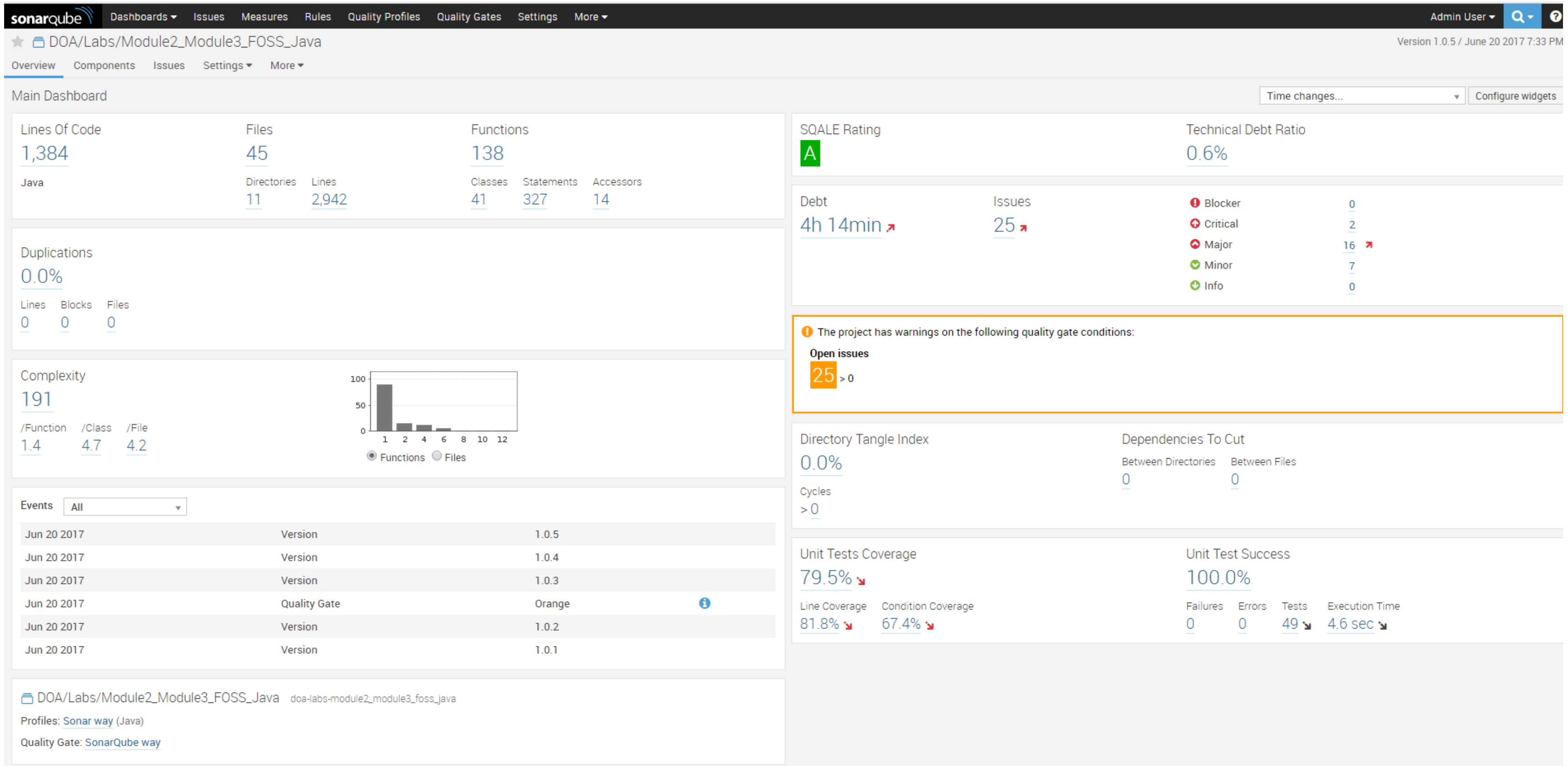
Icon: S M L

Legend | RSS for all | RSS for failures | RSS for just latest builds

Lab-3: Observe Tools

Step 3 - Sonar

- Sonar will be available from http://<Server-Public-IP>/sonar
- Have a browse, but don't change any configuration



DOA/Labs/Module2_Module3_FOSS_Java

Version 1.0.5 / June 20 2017 7:33 PM

Main Dashboard

Lines Of Code 1,384
Java

Files 45
Directories 11 Lines 2,942

Functions 138
Classes 41 Statements 327 Accessors 14

Duplications 0.0%
Lines 0 Blocks 0 Files 0

Complexity 191
/Function 1.4 /Class 4.7 /File 4.2

Events	All		
Jun 20 2017	Version	1.0.5	
Jun 20 2017	Version	1.0.4	
Jun 20 2017	Version	1.0.3	
Jun 20 2017	Quality Gate	Orange	
Jun 20 2017	Version	1.0.2	
Jun 20 2017	Version	1.0.1	

DOA/Labs/Module2_Module3_FOSS_Java doa-labs-module2_module3_foss_java
Profiles: Sonar way (Java)
Quality Gate: SonarQube way

SQALE Rating A

Technical Debt Ratio 0.6%

Debt 4h 14min

Issues 25

Blocker	0
Critical	2
Major	16
Minor	7
Info	0

The project has warnings on the following quality gate conditions:
Open issues
25 > 0

Directory Tangle Index 0.0%
Cycles > 0

Dependencies To Cut
Between Directories 0 Between Files 0

Unit Tests Coverage 79.5%
Line Coverage 81.8% Condition Coverage 67.4%

Unit Test Success 100.0%
Failures 0 Errors 0 Tests 49 Execution Time 4.6 sec

Lab-3: Observe Tools

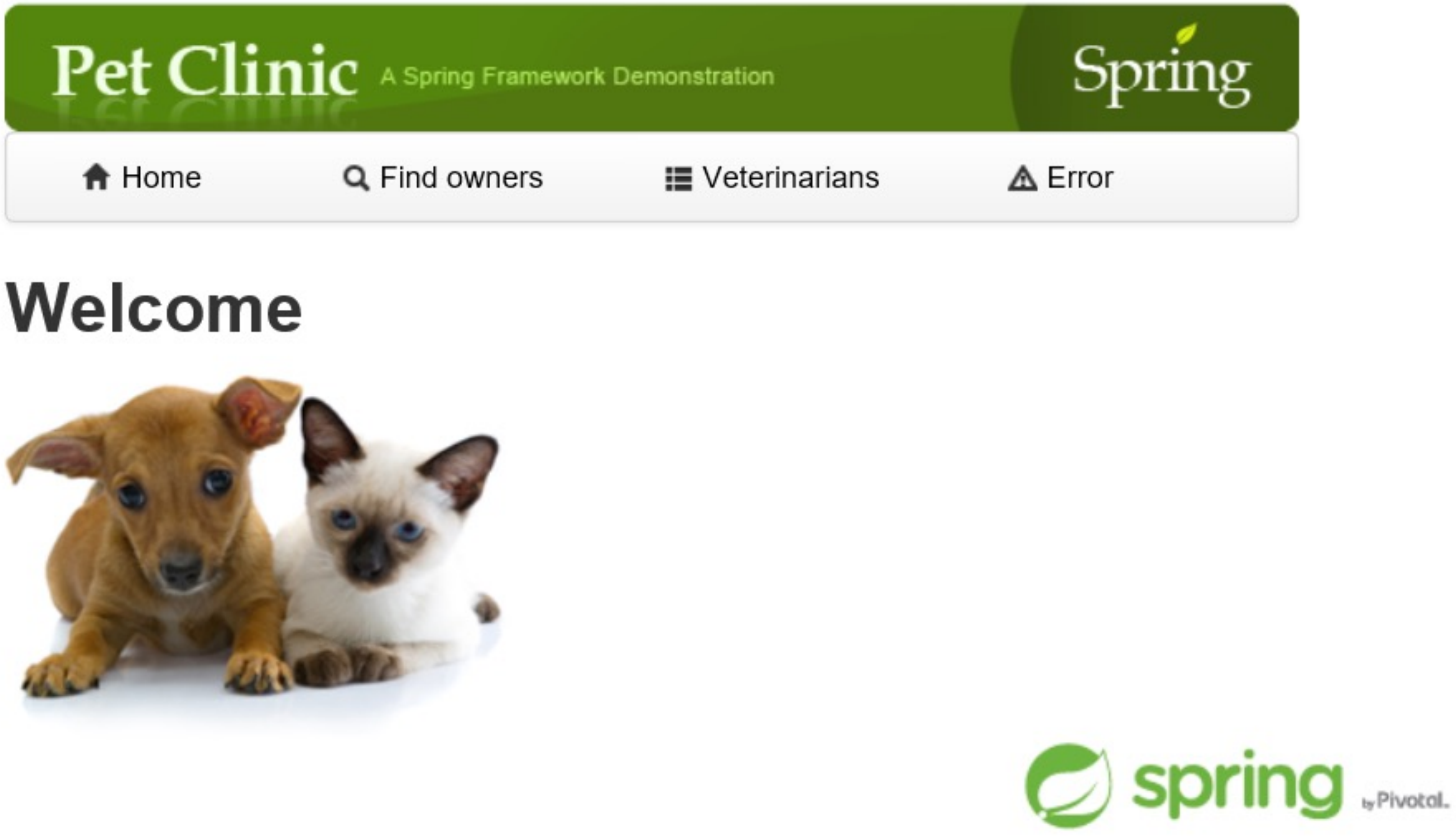
Step 4 - Review Landing Page of Application "Pet Clinic"

We need to run the "Pet Clinic" application build one time to ensure it is running correctly:

- git clone *http://devopsuser**'n'**@<public-IP>/gerrit/DOA/Labs/Module2_Module3_FOSS_Java/spring-petclinic* (username and public-ip provided to you by instructor)
- In your browser, navigate to Jenkins and then navigate to the **DOA/Labs/Module 2 and 3 - FOSS_Java** folder
- Click on the Java_Reference_Application tab
- Click on the Reference_Application_Build link (in the gray box).
- We now want to launch the pipeline by starting the Reference_Application_Build task, clicking on the "Build Now" link on the left of the page.
- Now navigate back and click on the "Java_Reference_Application" tab link to view the overall pipeline progress
- Once the pipeline has completed, (the last job to execute will be "Reference_Application_Performance_Tests") navigate to the Pet Clinic homepage (you may have to log in) and review (http://DOA_Labs_Module2_Module3_FOSS_Java_CI.<public-IP>.xip.io/petclinic).

Lab-3: Observe Tools

Step 5 – Landing page of "Pet Clinic"



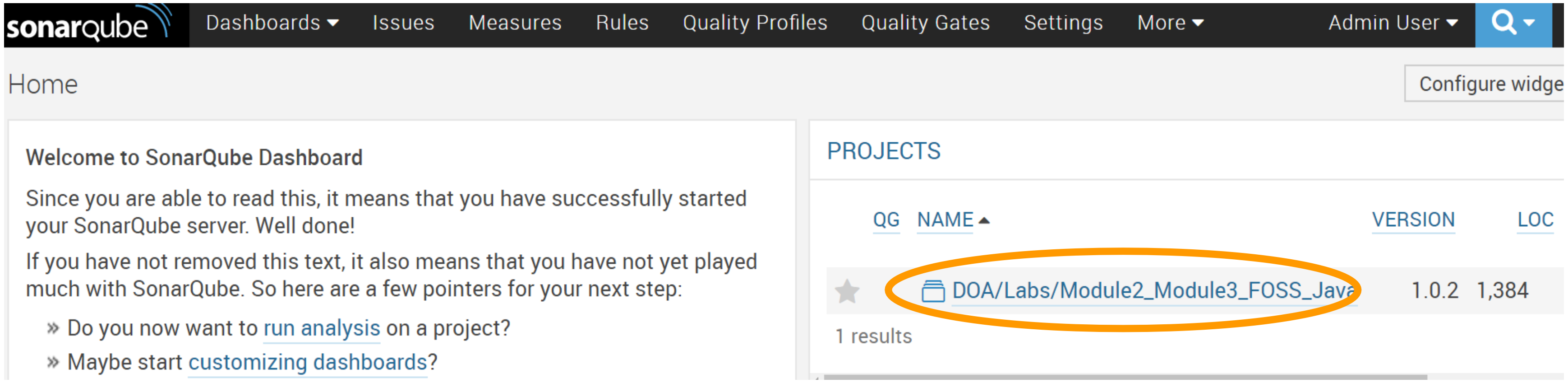
Welcome

LAB-4: Quality Gate (45 Mins)

Lab-4: Quality Gate

Step 1 – Navigate to Quality Gates

Navigate to Sonar (http://<Server-Public-IP>/sonar), select the project and go to Quality Gates (top bar)



Welcome to SonarQube Dashboard

Since you are able to read this, it means that you have successfully started your SonarQube server. Well done!

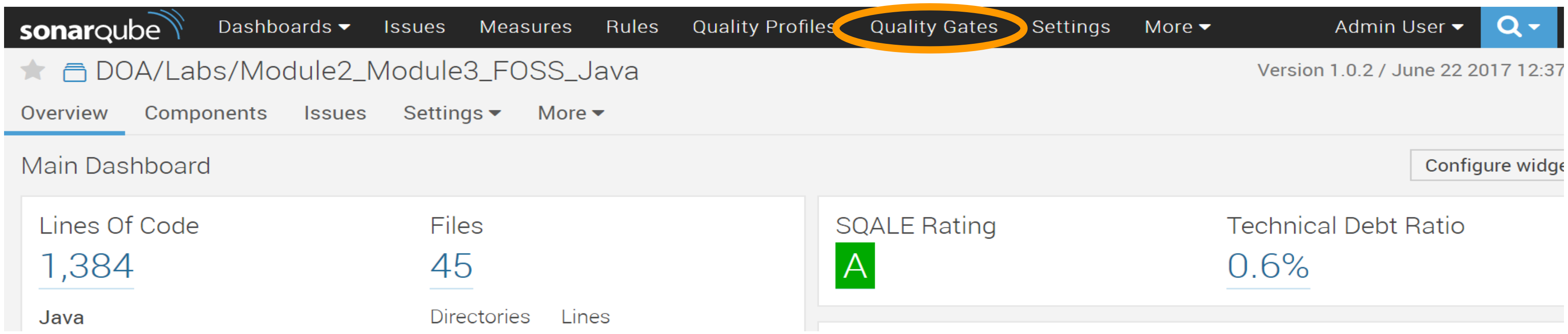
If you have not removed this text, it also means that you have not yet played much with SonarQube. So here are a few pointers for your next step:

» Do you now want to run analysis on a project?
» Maybe start customizing dashboards?

PROJECTS

QG	NAME	VERSION	LOC
	DOA/Labs/Module2_Module3_FOSS_Java	1.0.2	1,384

1 results



DOA/Labs/Module2_Module3_FOSS_Java

Version 1.0.2 / June 22 2017 12:37

Main Dashboard

Lines Of Code 1,384 Java

Files 45 Directories Lines

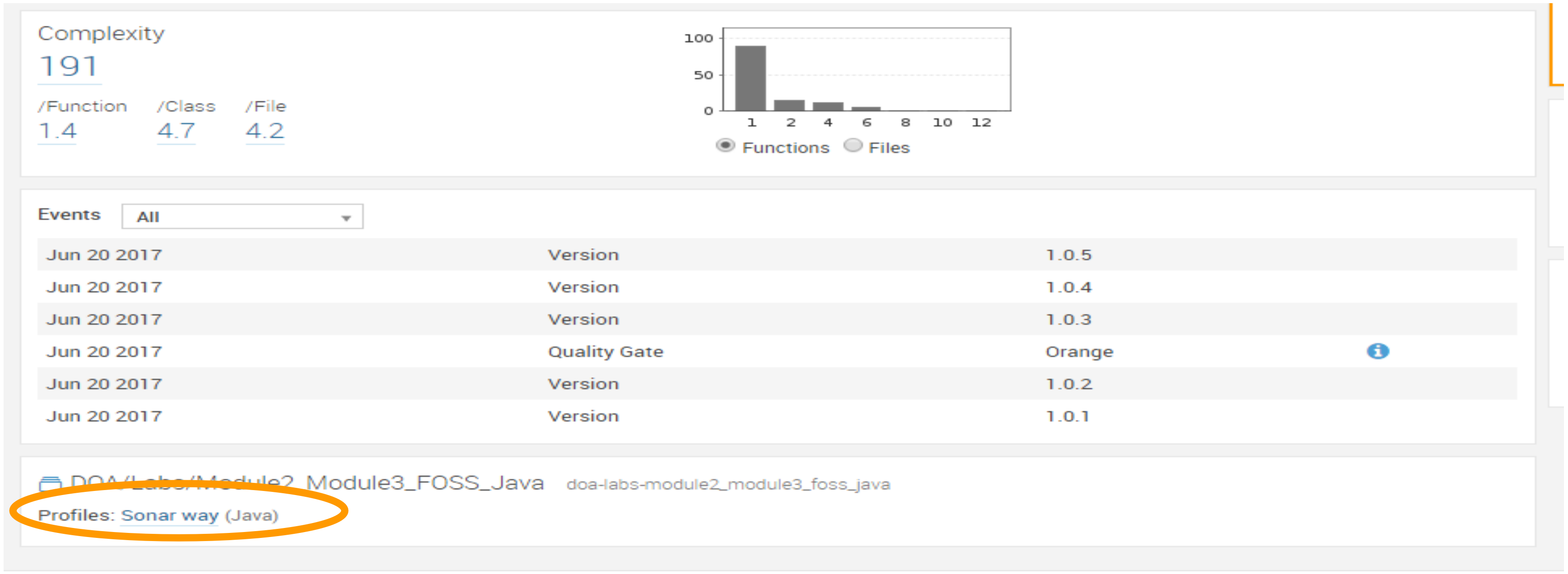
SQALE Rating A

Technical Debt Ratio 0.6%

Lab-4: Quality Gate

Step 2 – Observe the quality gate of Pet Clinic project

- In its initial state the "Module2_Module3_FOSS_Java" project does not have a Quality Gate assigned and therefore the Sonar step will never fail no matter how many Blockers or Critical errors it detects
- Navigate back to the dashboard for the "Module2_Module3_FOSS_Java" project. At the bottom you'll see the Quality Profile that is assigned. It should state the "Sonar Way" as shown below. **Note**: there is no Quality Gate assigned



Jun 20 2017	Version	1.0.5	
Jun 20 2017	Version	1.0.4	
Jun 20 2017	Version	1.0.3	
Jun 20 2017	Quality Gate	Orange	ℹ
Jun 20 2017	Version	1.0.2	
Jun 20 2017	Version	1.0.1	

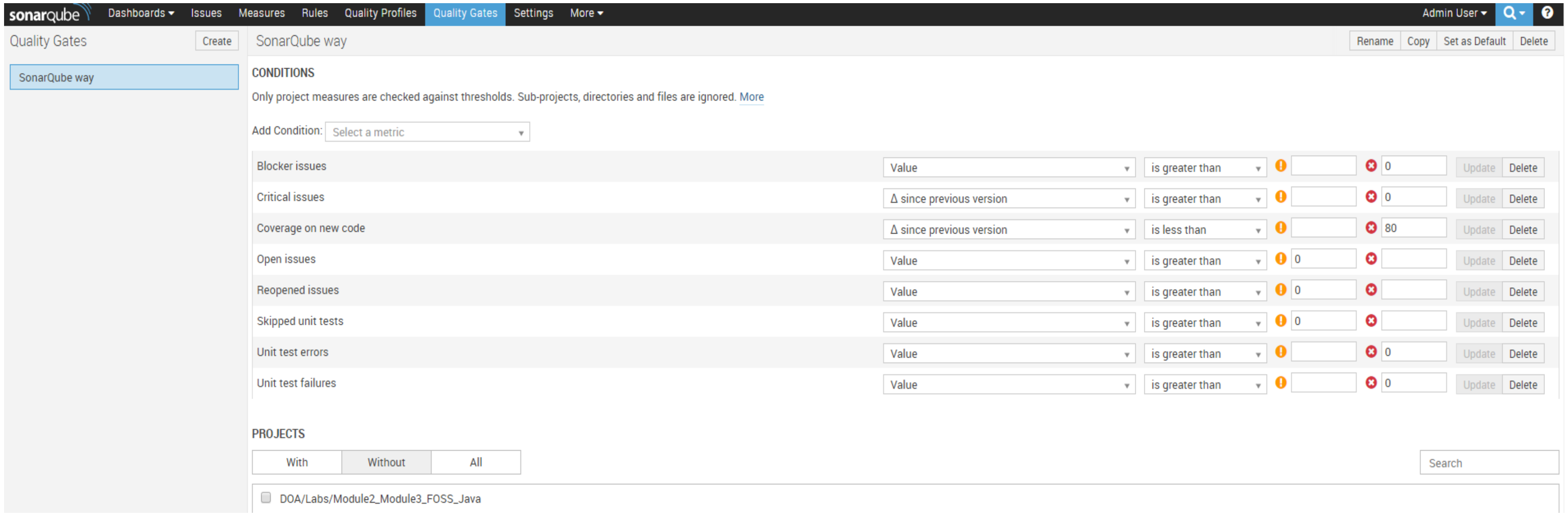
DOA/Labs/Module2_Module3_FOSS_Java doa-labs-module2_module3_foss_java

Profiles: Sonar way (Java)

Lab-4: Quality Gate

Step 3 - Assigning a quality gate to the Pet Clinic project

- At the bottom of the page, where you see a "PROJECTS" heading and 3 tabs; With, Without, and All.
- You can see the "Module2_Module3_FOSS_Java" project shows up in the Without tab, indicating that this Quality Gate is not associated with it.
- To assign this gate to the "Module2_Module3_FOSS_Java" project, click on the checkbox next to the project name as shown above. This will move "Module2_Module3_FOSS_Java" from the Without category to With category.



SonarQube way

CONDITIONS

Only project measures are checked against thresholds. Sub-projects, directories and files are ignored. More

Add Condition: Select a metric

Blocker issues	Value	is greater than		0	Update Delete
Critical issues	Δ since previous version	is greater than		0	Update Delete
Coverage on new code	Δ since previous version	is less than		80	Update Delete
Open issues	Value	is greater than	0		Update Delete
Reopened issues	Value	is greater than	0		Update Delete
Skipped unit tests	Value	is greater than	0		Update Delete
Unit test errors	Value	is greater than		0	Update Delete
Unit test failures	Value	is greater than		0	Update Delete

PROJECTS

With | Without | All

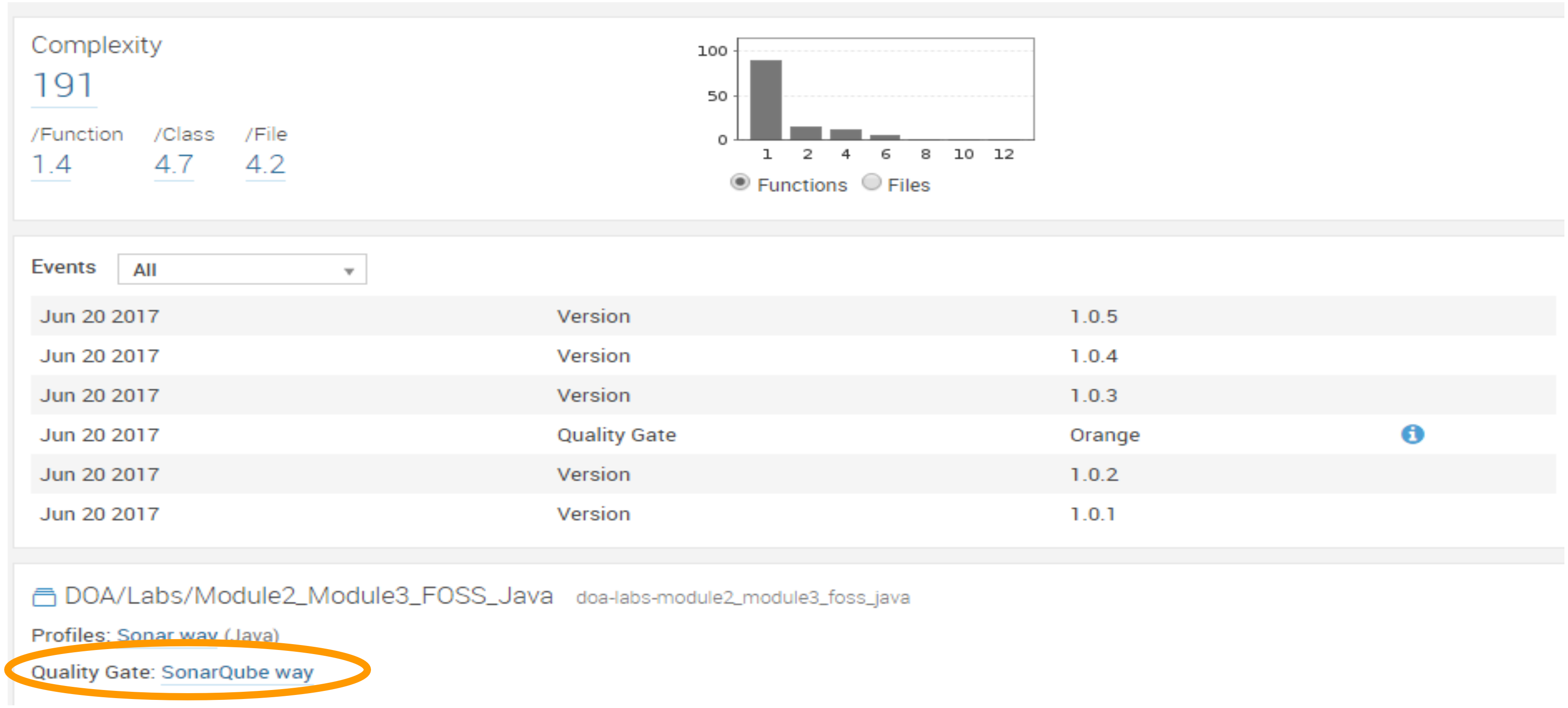
Search

☐ DOA/Labs/Module2_Module3_FOSS_Java

Lab-4: Quality Gate

Step 4 – Verify quality gate on Pet Clinic project

- Clicking on the With tab and you will see "Module2_Module3_FOSS_Java" there.
- Also confirm that it is assigned by checking the "Module2_Module3_FOSS_Java" dashboard where you should see "SonarQube Way" selected as the Quality



Complexity
191

/Function	/Class	/File
1.4	4.7	4.2

Events: All

Jun 20 2017	Version	1.0.5
Jun 20 2017	Version	1.0.4
Jun 20 2017	Version	1.0.3
Jun 20 2017	Quality Gate	Orange
Jun 20 2017	Version	1.0.2
Jun 20 2017	Version	1.0.1

DOA/Labs/Module2_Module3_FOSS_Java doa-labs-module2_module3_foss_java

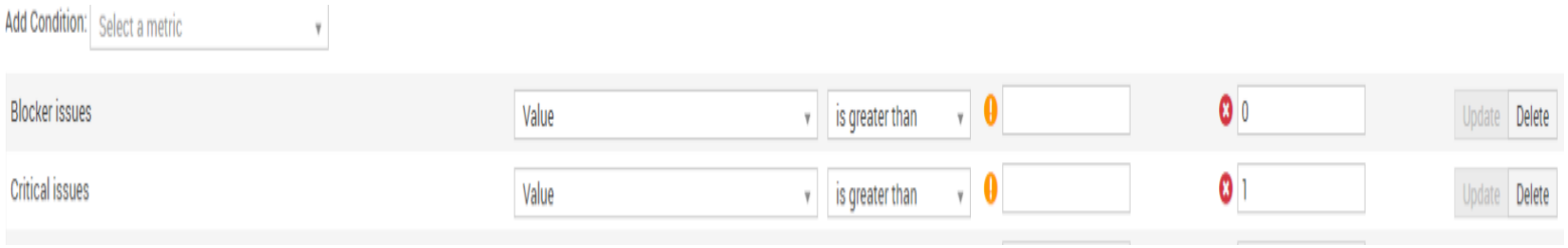
Profiles: Sonar way (Java)

Quality Gate: SonarQube way

Lab-4: Quality Gate

Step 5 – Configure quality gate

- Check the dashboard for "Module2_Module3_FOSS_Java" and you should see the results for the last run of the Sonar tests and you should also see that there were 2 Critical errors found.
- We do expect to fail due to stricter quality gates so modify the condition parameters accordingly.
- Condition currently set for Critical Issues is checking for the delta between the last run and the current. Select that condition and change it to "Value". Then change the numeric value in the error threshold field from 0 to 1, then click the Update button (to the right).



Add Condition: Select a metric

Blocker issues	Value	is greater than		0	Update Delete
Critical issues	Value	is greater than		1	Update Delete

Lab-4: Quality Gate

Step 6 –Change the code to see it working

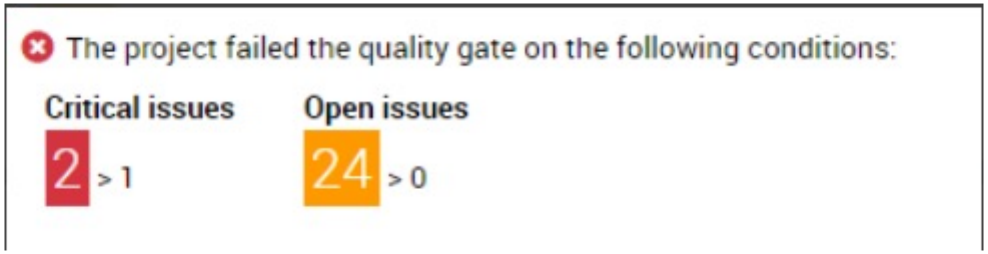
- open the file ~\spring-petclinic\src\main\resources\messages\messages.properties
- change the welcome message
- commit and push the changes using
- following git bash commands should be used to commit

```
$ git checkout master
$ git add .
$ git commit -am "Updating welcome banner"
$ git push
```

Lab-4: Quality Gate

Step 7 – Review Sonar report

- Sonar reports says

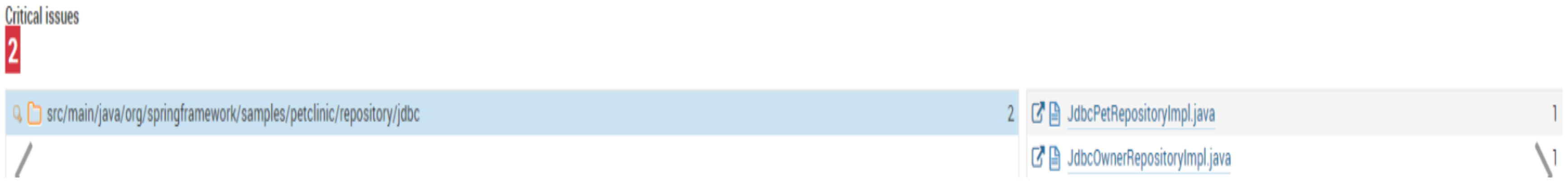


The project failed the quality gate on the following conditions:

Critical issues 2 > 1

Open issues 24 > 0

- The job failed as our code did not comply to our defined quality profile.
- go and check Sonar and try and drill down to the exact offending piece of code.
- To drill down, click on the "2" shown above, which will show that the JdbcPetRepositoryImpl.java and the JdbcOwnerRepositoryImpl.java contains 1 of the 2 critical errors each.



Critical issues
2

src/main/java/org/springframework/samples/petclinic/repository/jdbc 2

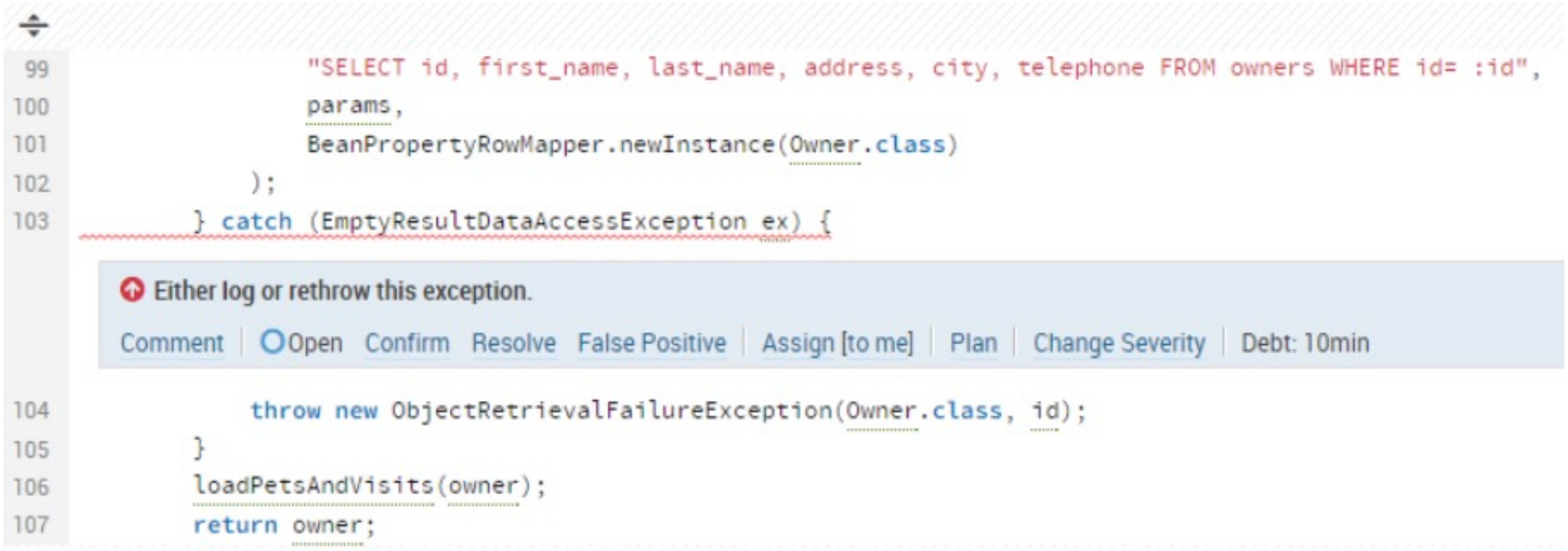
JdbcPetRepositoryImpl.java 1

JdbcOwnerRepositoryImpl.java 1

Lab-4: Quality Gate

Step 8 – Fix the build

- Click on the JdbcOwnerRepositoryImpl.java to see the errors and their location.



```
                "SELECT id, first_name, last_name, address, city, telephone FROM owners WHERE id= :id",
                params,
                BeanPropertyRowMapper.newInstance(Owner.class)
            );
        } catch (EmptyResultDataAccessException ex) {
```

Either log or rethrow this exception.

Comment | Open Confirm Resolve False Positive | Assign [to me] | Plan | Change Severity | Debt: 10min

```
            throw new ObjectRetrievalFailureException(Owner.class, id);
        }
        loadPetsAndVisits(owner);
        return owner;
```

- To resolve this issue, we will change the Quality Gate rules to what they used to be, please refer step-4 and revert to original settings.
- After reverting the changes done at step-4, re-run the pipeline by clicking "Build Now" on the Reference_Application_Build job and note that the pipeline will now <u>pass</u> the static code analysis.
- Open PetClinic website to see new welcome message.

LAB-5: TDD Challenge(45 Mins)

Lab-5: TDD Challenge

Challenge

We want to give Veterinarians an option to pick their own “display name” instead of only having the option of showing a First and Last Name only.

Requirements / Rules:
Given a nickname has already been established (e.g. Freddy),
then displayName should be ‘nickname + lastname’

Given no nickname has been entered (e.g. null “”)
Then displayName should be ‘firstname + lastname’

1. Write unit tests to verify these Requirements / Rules
   Hint* - the file to update is @ ~\spring-petclinic\src\test\java\org\springframework\samples\petclinic\model\ValidatorTests.java

2. Execute your Jenkins Pipeline
   Q: Did the build fail?

3. Write the code that will make these tests pass
   Hint* - the file to update is @ ~\spring-petclinic\src\main\java\org\springframework\samples\petclinic\model Person.Java

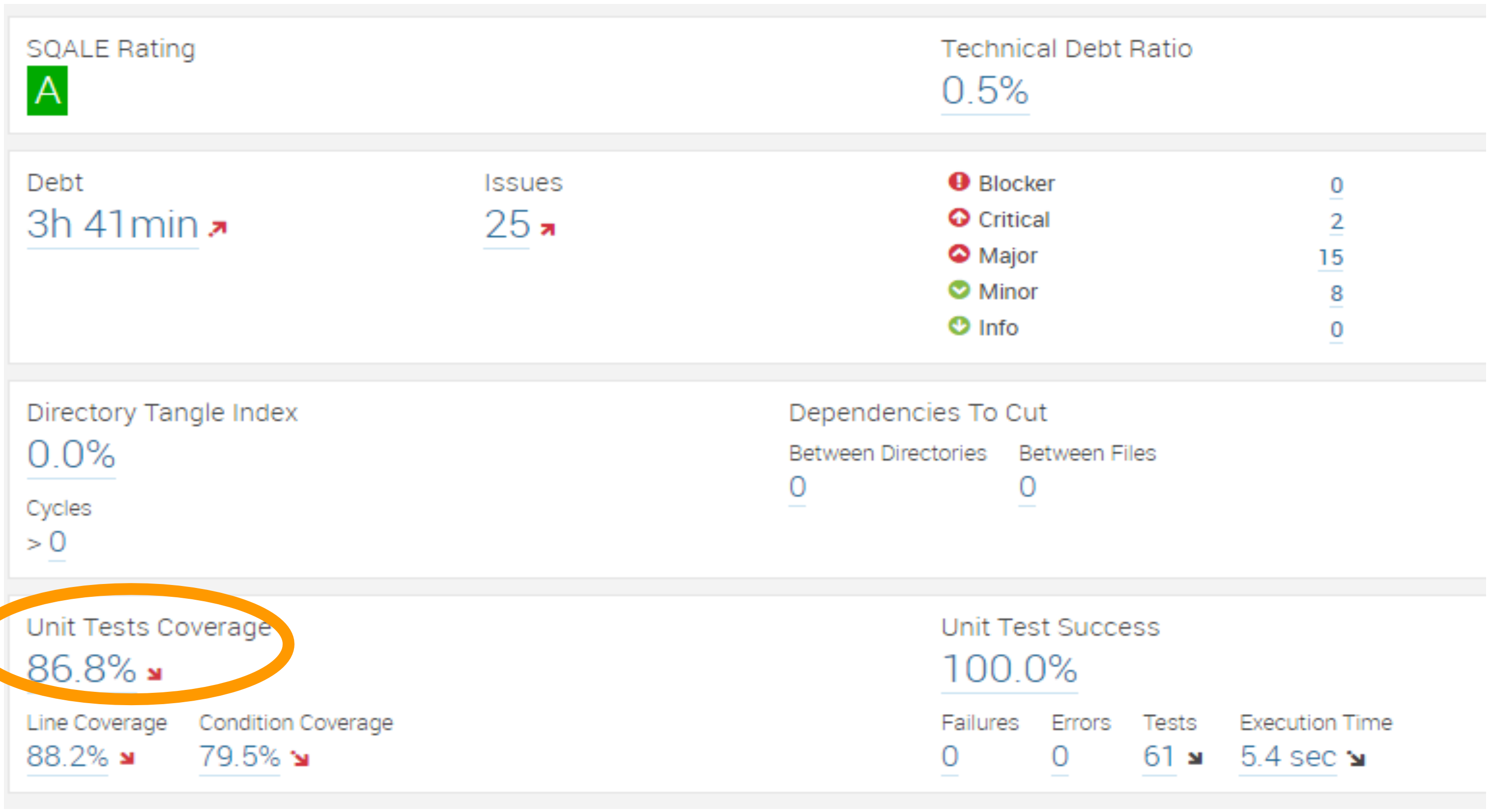
4. Execute your Jenkins Pipeline
   Q: Did the build pass?

LAB-6: Scavenger Hunt - Identify files with low test coverage (15 mins)

Lab-6: Low Coverage Check

Step 1 – Observe the Current UTC from Sonar Dashboard

- Navigate to DOA/Labs/Module2_Module3_FOSS_Java



SQALE Rating
A

Technical Debt Ratio
0.5%

Debt
3h 41min

Issues
25

Blocker	0
Critical	2
Major	15
Minor	8
Info	0

Directory Tangle Index
0.0%

Cycles
> 0

Dependencies To Cut

Between Directories	Between Files
0	0

Unit Tests Coverage
86.8%

Line Coverage	Condition Coverage
88.2%	79.5%

Unit Test Success
100.0%

Failures	Errors	Tests	Execution Time
0	0	61	5.4 sec

Lab-6: Low Coverage Check

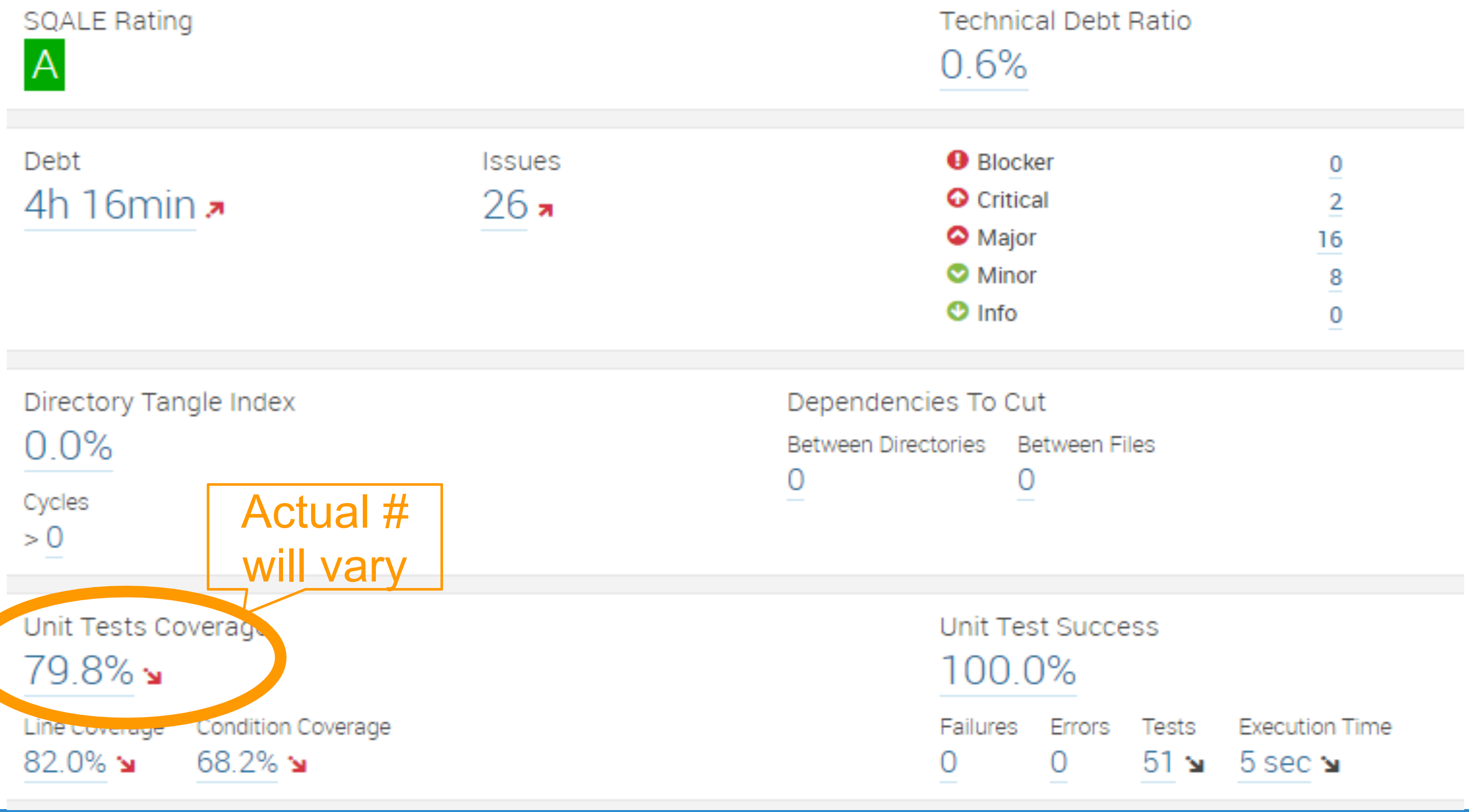
Step 2 – In your Pair, Person 1 comments out some unit tests without telling his/her partner the location of the file

- E.g. Open **OwnerControllerTests.java** located @ ~\spring-petclinic\src\test\java\org\springframework\samples\petclinic\web
- Comment out the last 7 test cases and save this file.
- Push your changes to master branch and check Jenkins console, build should pass.

Lab-6: Low Coverage Check

Step 3 – Browse Sonar dashboard

- Navigate to DOA/Labs/Module2_Module3_FOSS_Java
- Observe Unit Test coverage should be lower than it was originally



SQALE Rating
A

Technical Debt Ratio
0.6%

Debt
4h 16min

Issues
26

Blocker	0
Critical	2
Major	16
Minor	8
Info	0

Directory Tangle Index
0.0%

Cycles
> 0

Dependencies To Cut

Between Directories	Between Files
0	0

Actual # will vary

Unit Tests Coverage
79.8%

Line Coverage	Condition Coverage
82.0%	68.2%

Unit Test Success
100.0%

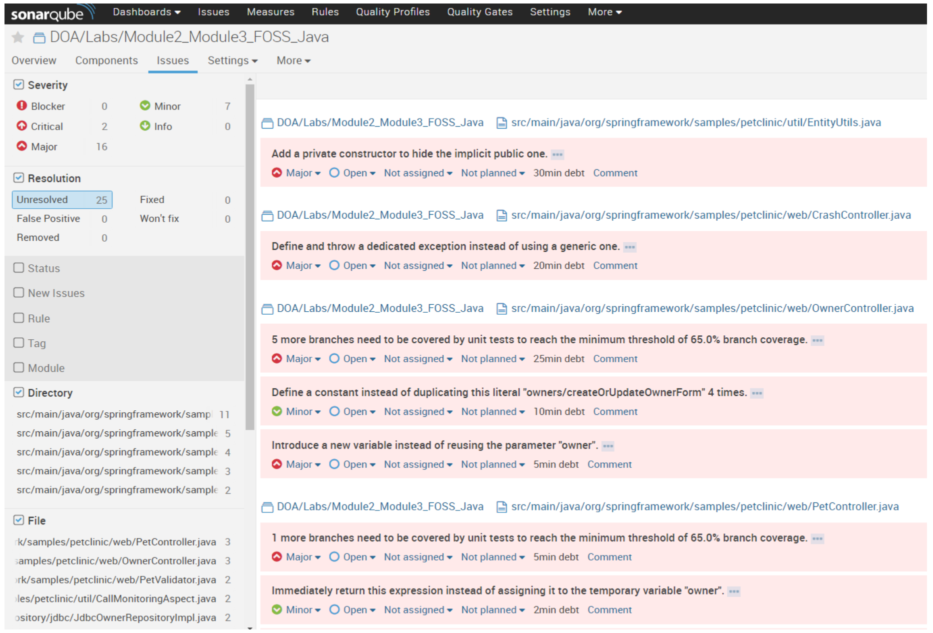
Failures	Errors	Tests	Execution Time
0	0	51	5 sec

Lab-6: Low Coverage Check

Step 4 – Person 2 from the Pair attempts to find the class with low test coverage

For example,

- Click on ~/samples/petclinic/web
- OwnerController.java is showing limited coverage
- Click on OwnerController.java
- Sonar displays all methods that are not covered with unit test case



DOA/Labs/Module2_Module3_FOSS_Java

Severity

Blocker	0	Minor	7
Critical	2	Info	0
Major	16		

Resolution

Unresolved	25	Fixed	0
False Positive	0	Won't fix	0
Removed	0		

Status
New Issues
Rule
Tag
Module

Directory

src/main/java/org/springframework/samp	11
src/main/java/org/springframework/sample	5
src/main/java/org/springframework/sample	4
src/main/java/org/springframework/sample	3
src/main/java/org/springframework/sample	2

File

k/samples/petclinic/web/PetController.java	3
amples/petclinic/web/OwnerController.java	3
rk/samples/petclinic/web/PetValidator.java	2
les/petclinic/util/CallMonitoringAspect.java	2
ository/jdbc/JdbcOwnerRepositoryImpl.java	2

DOA/Labs/Module2_Module3_FOSS_Java src/main/java/org/springframework/samples/petclinic/util/EntityUtils.java

Add a private constructor to hide the implicit public one.
Major Open Not assigned Not planned 30min debt Comment

DOA/Labs/Module2_Module3_FOSS_Java src/main/java/org/springframework/samples/petclinic/web/CrashController.java

Define and throw a dedicated exception instead of using a generic one.
Major Open Not assigned Not planned 20min debt Comment

DOA/Labs/Module2_Module3_FOSS_Java src/main/java/org/springframework/samples/petclinic/web/OwnerController.java

5 more branches need to be covered by unit tests to reach the minimum threshold of 65.0% branch coverage.
Major Open Not assigned Not planned 25min debt Comment

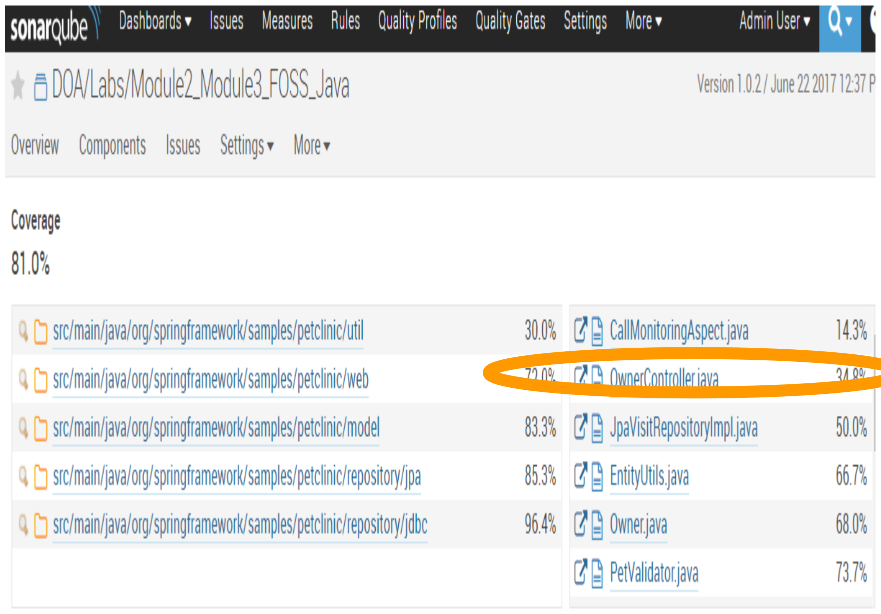
Define a constant instead of duplicating this literal "owners/createOrUpdateOwnerForm" 4 times.
Minor Open Not assigned Not planned 10min debt Comment

Introduce a new variable instead of reusing the parameter "owner".
Major Open Not assigned Not planned 5min debt Comment

DOA/Labs/Module2_Module3_FOSS_Java src/main/java/org/springframework/samples/petclinic/web/PetController.java

1 more branches need to be covered by unit tests to reach the minimum threshold of 65.0% branch coverage.
Major Open Not assigned Not planned 5min debt Comment

Immediately return this expression instead of assigning it to the temporary variable "owner".
Minor Open Not assigned Not planned 2min debt Comment



DOA/Labs/Module2_Module3_FOSS_Java

Version 1.0.2 / June 22 2017 12:37 P

Coverage

81.0%

src/main/java/org/springframework/samples/petclinic/util	30.0%	CallMonitoringAspect.java	14.3%
src/main/java/org/springframework/samples/petclinic/web	72.0%	OwnerController.java	34.8%
src/main/java/org/springframework/samples/petclinic/model	83.3%	JpaVisitRepositoryImpl.java	50.0%
src/main/java/org/springframework/samples/petclinic/repository/jpa	85.3%	EntityUtils.java	66.7%
src/main/java/org/springframework/samples/petclinic/repository/jdbc	96.4%	Owner.java	68.0%
		PetValidator.java	73.7%

Lab-6: Low Coverage Check

Step 5 – As a Pair, add unit test cases to increase coverage

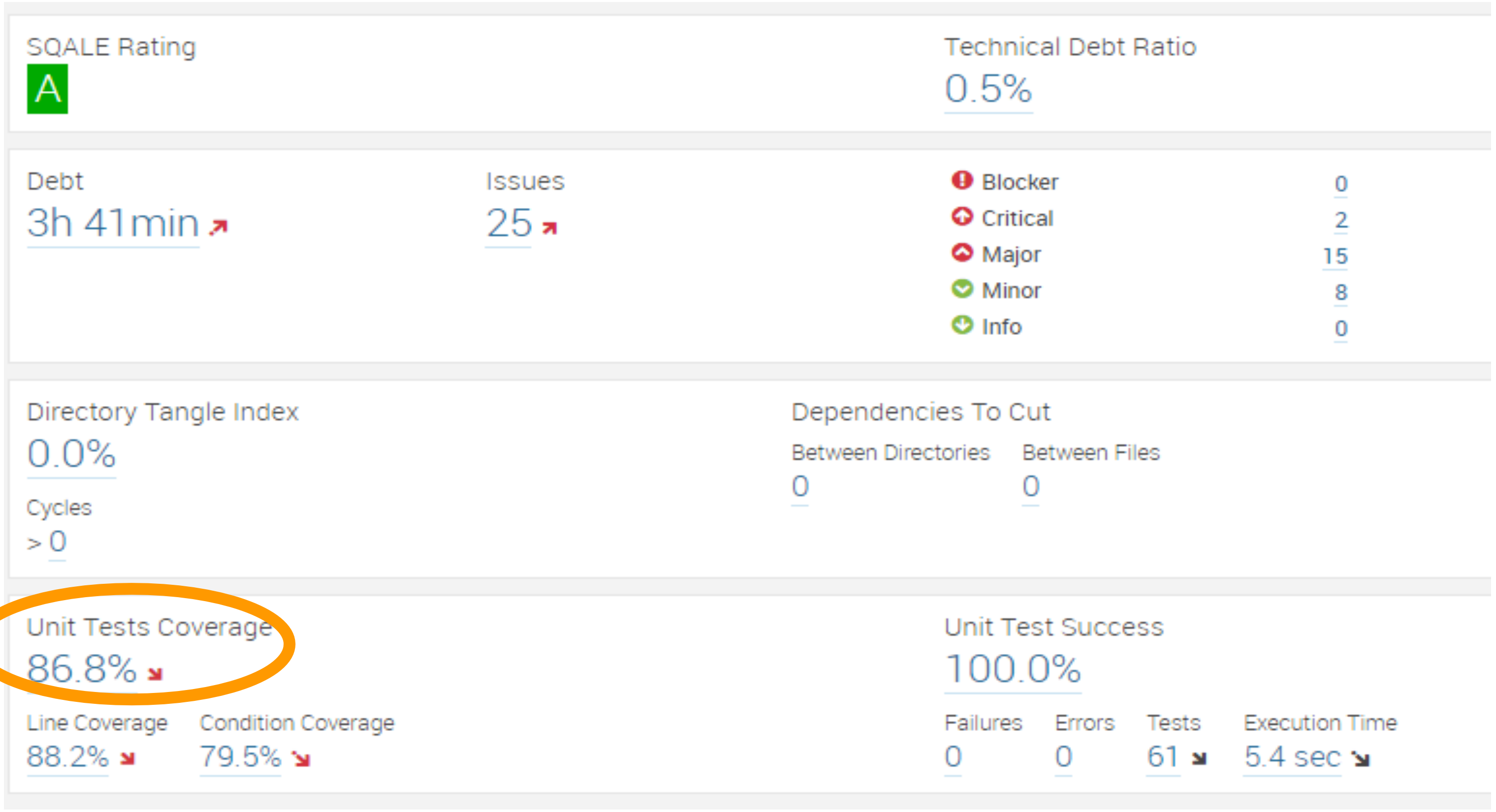
For example,

- Open OwnerControllerTests.java located @ ~\spring-petclinic\src\test\java\org\springframework\samples\petclinic\web
- Uncomment the test cases
- Push your changes to master branch and check Jenkins console, build should pass

Lab-6: Low Coverage Check

Step 6 – Browse Sonar dashboard

- Navigate again to DOA/Labs/Module2_Module3_FOSS_Java
- Observe improved unit test coverage



SQALE Rating
A

Technical Debt Ratio
0.5%

Debt
3h 41min

Issues
25

Blocker	0
Critical	2
Major	15
Minor	8
Info	0

Directory Tangle Index
0.0%

Cycles
> 0

Dependencies To Cut

Between Directories	Between Files
0	0

Unit Tests Coverage
86.8%

Line Coverage	Condition Coverage
88.2%	79.5%

Unit Test Success
100.0%

Failures	Errors	Tests	Execution Time
0	0	61	5.4 sec