

Software Testing

Project1

Continuous Integration

Tutorial

Junit + Github + Jenkins+JaCoCo

Instructor: 王凡教授

TA: 鄭大容

Email: R06943080@ntu.edu.tw

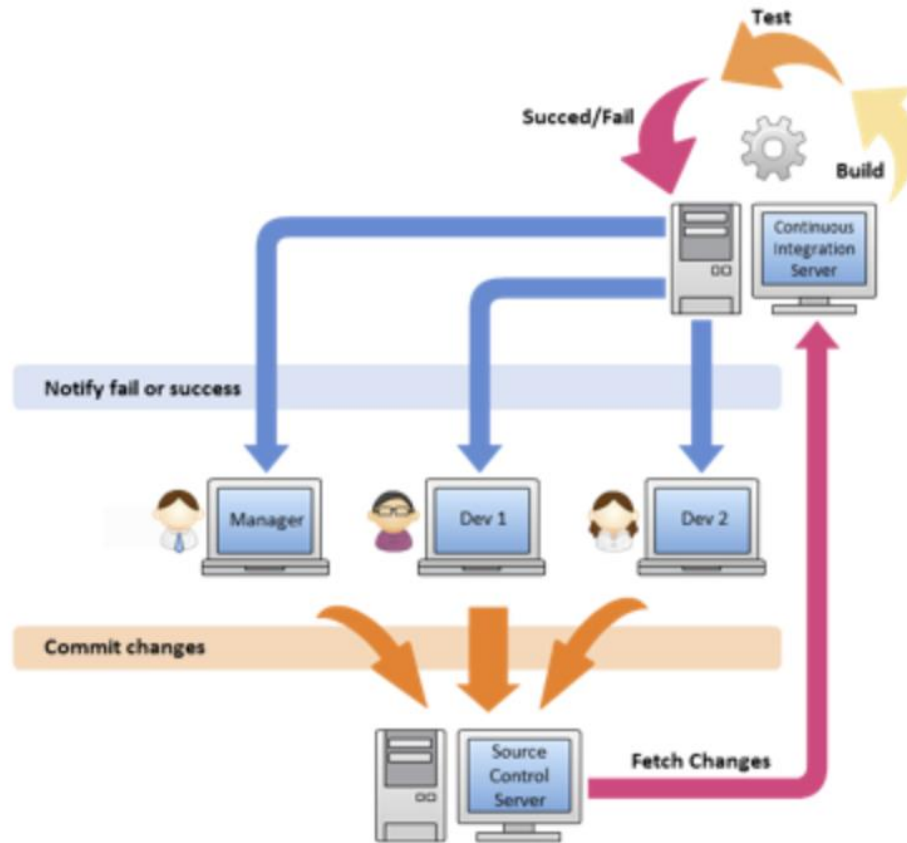
Unit Test

- 對於程式碼中的單元如class, method進行測試的方法。
- 對於大型專案的單元測試，已有現成的IDE工具可供使用。
Ex. Eclipse

What is CI?

- Continuous integration is the method used to determine that the entire system is in the running state and the changes made have not lead to errors in some parts of the system.

How CI works?



What is Jenkins?

- An open source continuous integration and build automation tool
- Helps automate tasks:
 - Building
 - Testing
 - Deploying

Goals

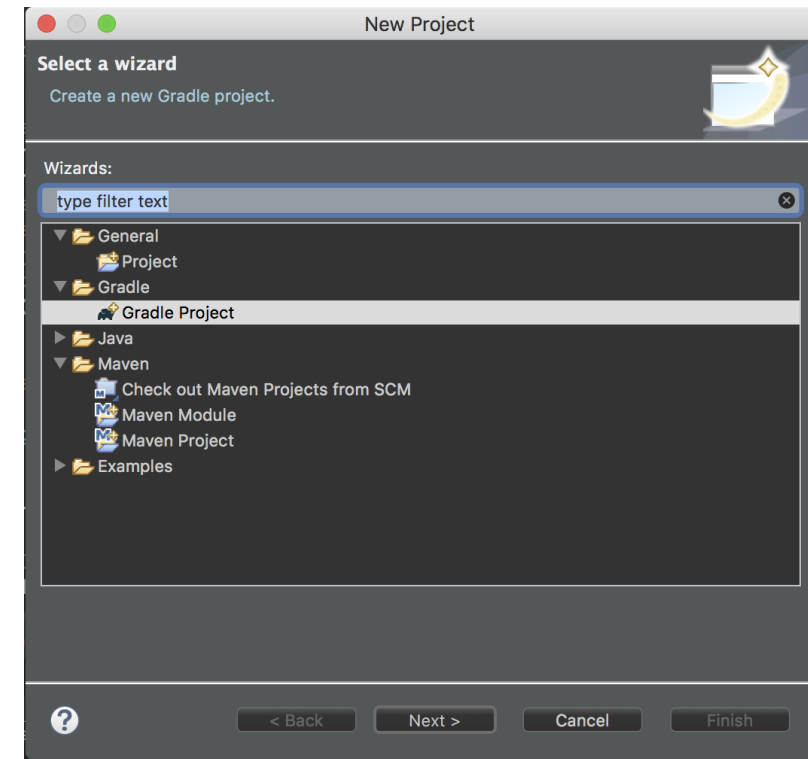
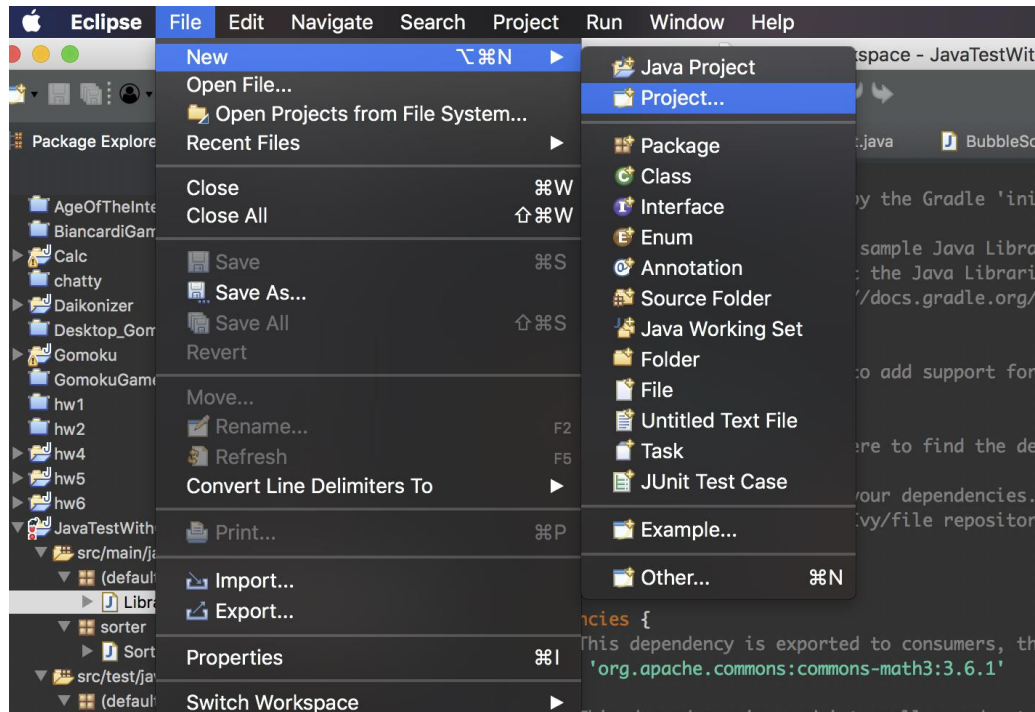
1. Create a Java project
2. Use an external library. (ex. Sort)
3. Writing testcases for this project
4. Create a public repo and push your work to Github
5. Setup Jenkins server
6. Automatic fetch and build your java project from Github on Jenkins
7. Coverage report with JaCoCo plugin
8. Find an external library and design your testcases

Prerequisite

- Ubuntu 16.04
 - This tutorial was written and tested on Ubuntu 16.04
 - There might be some difference if you use other OS.
- Install Java SE 8 first
 - For Ubuntu
 - `sudo add-apt-repository ppa:webupd8team/java`
 - `sudo apt-get update`
 - `sudo apt-get install oracle-java8-installer`
- Eclipse IDE

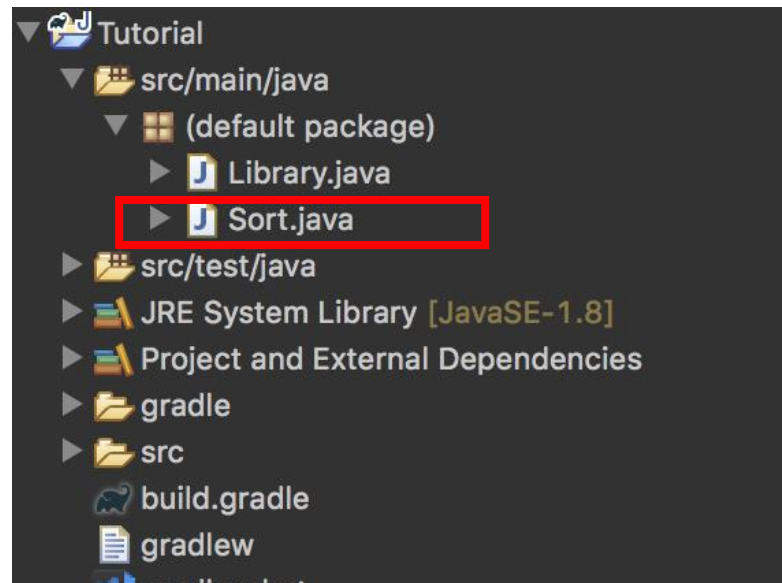
Goal 1: Create a Java project

- We use eclipse for demo:
 - Download here: <https://www.eclipse.org/downloads/packages/release/2018-09/r/eclipse-ide-java-developers>
- We'll be using Gradle to manage our Java project
- File >> New >> Project >> **Select Gradle Project**



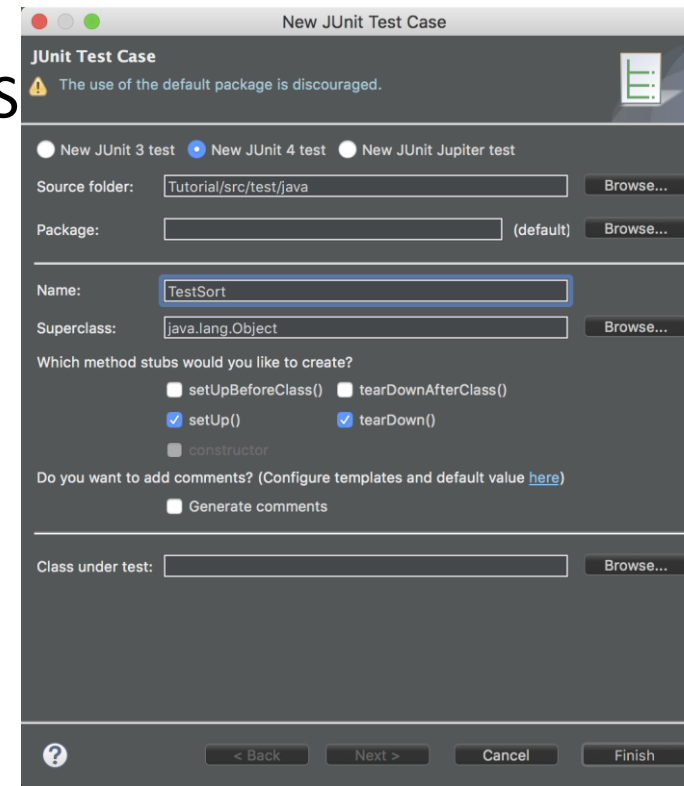
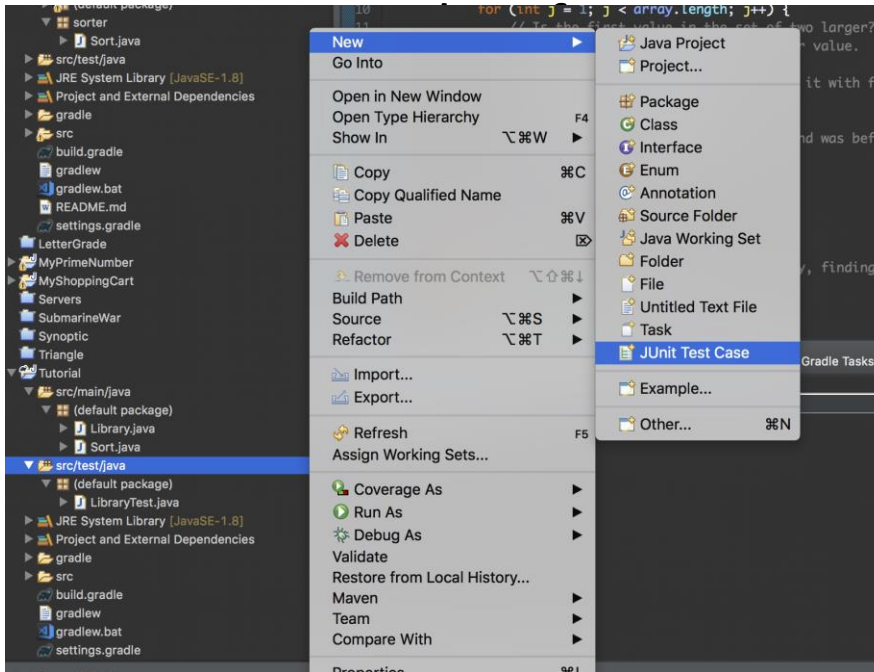
Goal 2: Use an external library

- In our demo:
 - We use a sort algorithm library: [LINK](#)
- Drag & Drop the file into src/main/java



Goal 3: Writing testcases

- In src/test/java >> Right click >> New >> Junit Testcase:



Goal 3: Writing testcases

- Let's say, we're testing on the Bubble sort algorithm
 - Method with decorator `@Test` will be executed while testing
 - You can import other decorator such as `@BeforeClass`, `@AfterClass`, `@Ignore`
 - See https://netbeans.org/kb/docs/java/junit-intro.html#Exercise_10
 - Using the external class `Sort`

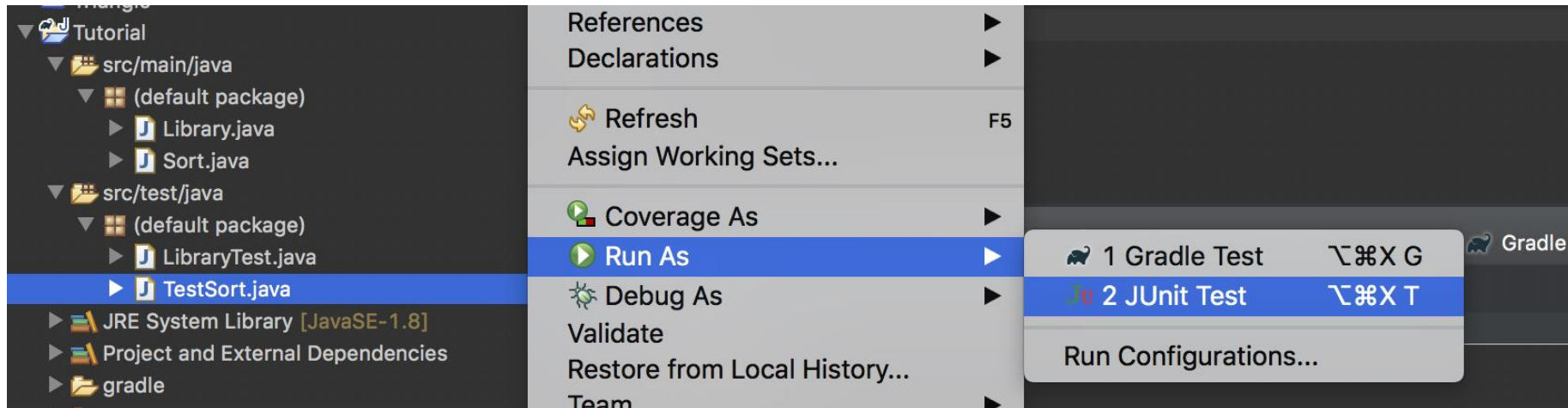
Goal 3: Writing testcases

- Example

```
1 package sorter;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.After;
6 import org.junit.Before;
7 import org.junit.Test;
8
9 public class SortTest {
10
11     @Before
12     public void setUp() throws Exception {
13     }
14
15     @After
16     public void tearDown() throws Exception {
17     }
18
19     @Test
20     public void testBubble() {
21         System.out.println("bubble");
22         Sort sort = new Sort();
23         assertEquals(new int[] {1,2,3}, sort.bubble(new int[] {3,1,2}));
24         assertEquals(new int[] {1,2,3,4}, sort.bubble(new int[] {3,1,4,2}));
25         assertEquals(new int[] {1,2,3,4,5}, sort.bubble(new int[] {5,3,1,4,2}));
26         assertEquals(new int[] {1,2,3,4,5,6}, sort.bubble(new int[] {6,3,1,5,4,2}));
27         assertEquals(new int[] {1,2,3,4,5,6,7}, sort.bubble(new int[] {3,7,1,6,4,2,5}));
28         assertEquals(new int[] {1,2,3,4,5,6,7,8}, sort.bubble(new int[] {3,7,1,8,6,4,2,5}));
29         assertEquals(new int[] {1,2,3,4,5,6,7,8,9}, sort.bubble(new int[] {3,7,9,1,8,6,4,2,5}));
30         assertEquals(new int[] {1,2,3,4,5,6,7,8,9,10}, sort.bubble(new int[] {3,7,9,1,8,6,4,2,10,5}));
31     }
32
33     @Test
34     public void testSelection() {
35         System.out.println("selection");
36         Sort sort = new Sort();
37         assertEquals(new int[] {1,2,3}, sort.selection(new int[] {3,1,2}));
38         assertEquals(new int[] {1,2,3,4}, sort.selection(new int[] {3,1,4,2}));
39         assertEquals(new int[] {1,2,3,4,5}, sort.selection(new int[] {5,3,1,4,2}));
40         assertEquals(new int[] {1,2,3,4,5,6}, sort.selection(new int[] {6,3,1,5,4,2}));
41         assertEquals(new int[] {1,2,3,4,5,6,7}, sort.selection(new int[] {3,7,1,6,4,2,5}));
```

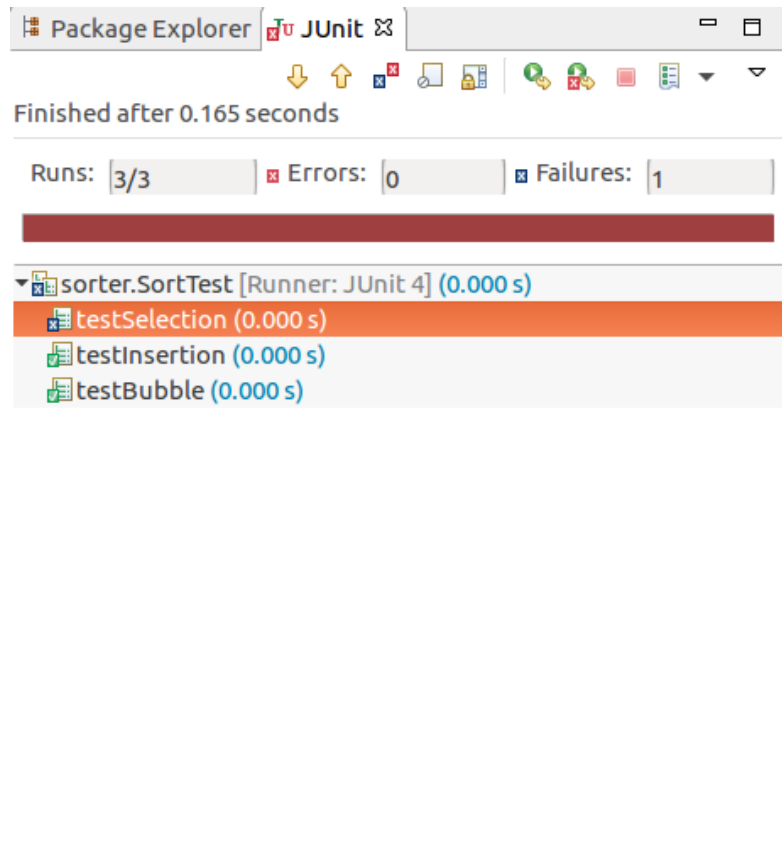
Goal 3: Writing testcases

- We can run the test locally
 - Right click on the testing class >> Run as >> JUnit Test



Goal 3: Writing testcases

- Test Result:



Goal 3: Writing testcases

- Buggy method:

```
public int[] selection(int[] array) {  
    // Go through the array once, from last place to first  
    for (int i = array.length - 1; i > 0; i--) {  
        // The highest number in the starting is always the first  
        int highest = array[0];  
        int index = 0;  
        // Cycle through all the unsorted numbers, to find the highest  
        for (int j = 0; j < i; j++) {  
            // Is this number higher than the largest number so far?  
            if (array[j] > highest) {  
                // Then replace the current record with this number.  
                highest = array[j];  
                // ...and store which index it's in for later.  
                index = j;  
            }  
        }  
        // After the highest unsorted number has been found, switch the highest number with the last unsorted number.  
        int temp = array[i - 1] = highest;  
        array[index] = temp;  
    }  
    return array;  
}
```

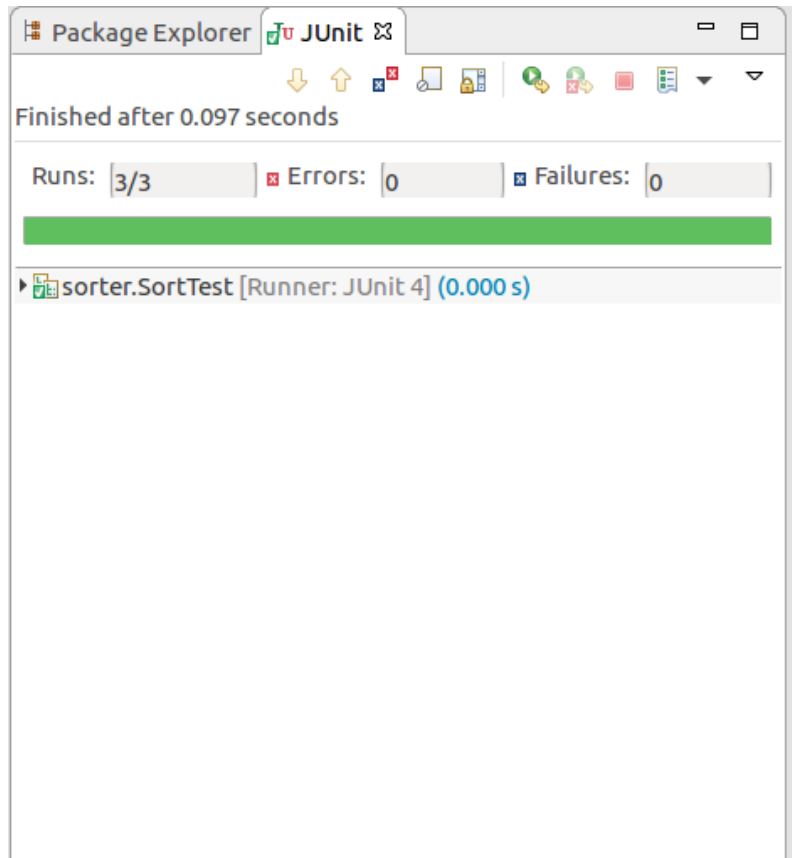
Goal 3: Writing testcases

- Replaced with correct codes:

```
26 public int[] selection(int[] array) {  
27     // Go through the array once, from last place to first  
28     for (int i = array.length - 1; i > 0; i--) {  
29         // The highest number in the starting is always the first.  
30         int highest = array[i];  
31         int index = i;  
32         // Cycle through all the unsorted numbers, to find the highest  
33         for (int j = 0; j < i; j++) {  
34             // Is this number higher than the largest number so far?  
35             if (array[j] > highest) {  
36                 // Then replace the current record with this number.  
37                 highest = array[j];  
38                 // ...and store which index it's in for later.  
39                 index = j;  
40             }  
41         }  
42         // After the highest unsorted number has been found, switch the highest number with the last unsorted number.  
43         int temp = array[i];  
44         array[i] = array[index];  
45         array[index] = temp;  
46     }  
47     return array;  
48 }
```


Goal 3: Writing testcases

- Test result:



Goal 4: Create a public repo and push your work

- If you do not have a Github account, **register first on** <https://github.com/>
- Create a new public repo

Repositories

[New](#)

Find a repository...

- [jeffwang0516/Etrip](#)
- [jeffwang0516/gemini-solution-generat...](#)
- [jeffwang0516/Gomoku-AI2AI](#)
- [jeffwang0516/django-restful-with-react](#)
- [jeffwang0516/JavaTestWithCI](#)
- [jeffwang0516/ios-test](#)
- [jeffwang0516/Event-ticket-DApp](#)

Show more

You are in the public [Beta](#) of the dashboard!
[Opt out](#) or [send feedback](#).

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner **Repository name**

[jeffwang0516](#) /

Great repository names are short and memorable. Need inspiration? How about [turbo-winner](#).

Description (optional)

☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: [None](#) | Add a license: [None](#) ⓘ

Goal 4: Create a public repo and push your work

- On your terminal, cd to your project folder
- If your not familiar with git: Tutorial
- Sample instruction:

```
$ cd to your project folder

$ git init
$ git add .
$ git commit -m "Initial commit"

# Replace xxxxxx with your own account
$ git remote add origin https://github.com/xxxxxx/JavaTest.git

$ git push -u origin master
```

Goal 4: Terminal Snapshot

```
Tutorial — jeff@jeffwang — ..pace/Tutorial — -zsh
Last login: Tue Sep 25 22:15:01 on ttys000
jeff@jeffwang ~ ➤ cd eclipse-workspace/Tutorial
jeff@jeffwang ~/eclipse-workspace/Tutorial ➤ git init
Initialized empty Git repository in /Users/jeff/eclipse-workspace/Tutorial/.git/
jeff@jeffwang ~/eclipse-workspace/Tutorial ➤ ↗ master ➤ git add .
jeff@jeffwang ~/eclipse-workspace/Tutorial ➤ ↗ master ➤ git commit -m "Initial commit"
"
jeff@jeffwang ~/eclipse-workspace/Tutorial ➤ ↗ master ➤ git remote add origin https://github.com/jeffwang0516/JavaTest.git
jeff@jeffwang ~/eclipse-workspace/Tutorial ➤ ↗ master ➤ git push -u origin master
Counting objects: 60, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (46/46), done.
Writing objects: 100% (60/60), 65.52 KiB | 5.96 MiB/s, done.
Total 60 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/jeffwang0516/JavaTest/pull/new/master
remote:
To https://github.com/jeffwang0516/JavaTest.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
jeff@jeffwang ~/eclipse-workspace/Tutorial ➤ ↗ master ➤
```

Goal 4: Create a public repo and push your work

- You're now able to see files on Github

The screenshot shows a GitHub repository page for 'jeffwang0516 / JavaTest'. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The 'Code' tab is selected, showing a list of files and folders. The files include .gradle, .settings, bin, build, gradle/wrapper, src, .classpath, .project, build.gradle, gradlew, gradlew.bat, and settings.gradle. All files were committed 'Initial commit' 4 minutes ago.

jeffwang0516 / JavaTest

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

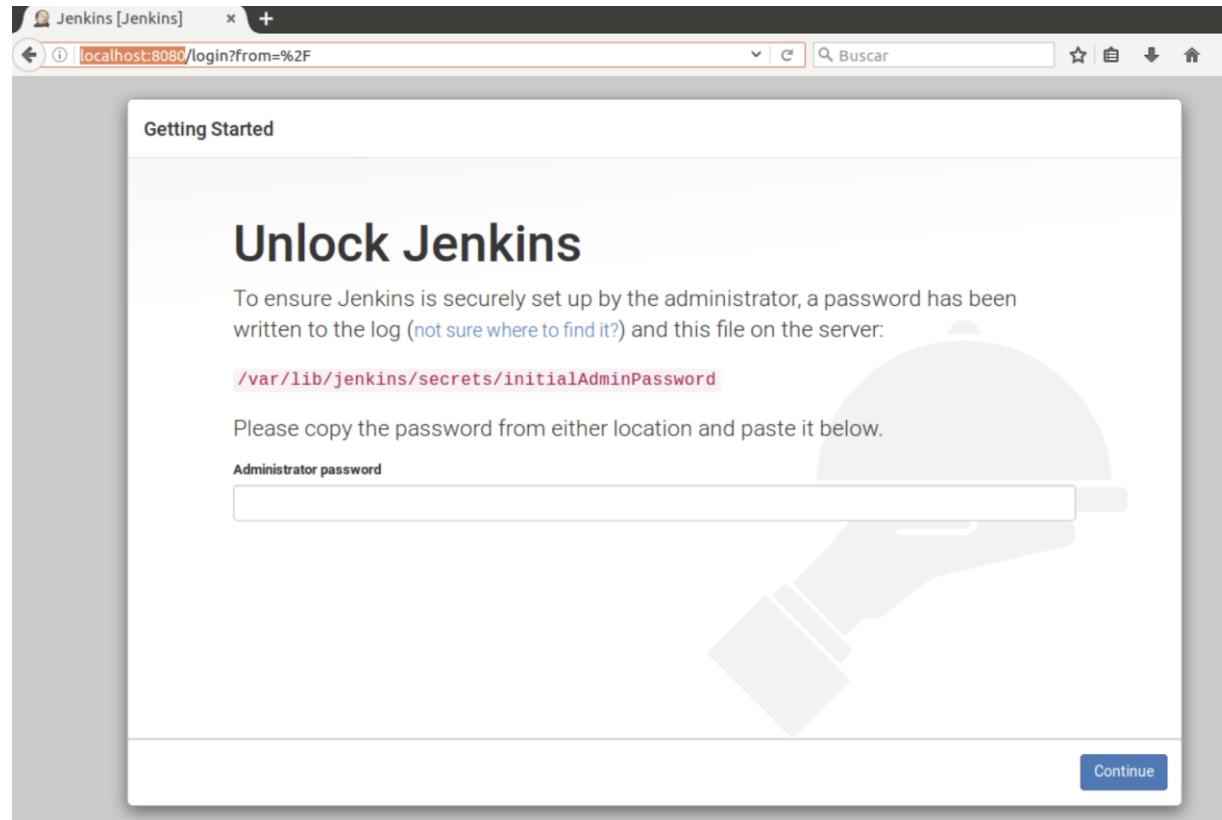
jeffwang0516 Initial commit	Latest commit 91b992e 5 minutes ago
.gradle	Initial commit 4 minutes ago
.settings	Initial commit 4 minutes ago
bin	Initial commit 4 minutes ago
build	Initial commit 4 minutes ago
gradle/wrapper	Initial commit 4 minutes ago
src	Initial commit 4 minutes ago
.classpath	Initial commit 4 minutes ago
.project	Initial commit 4 minutes ago
build.gradle	Initial commit 4 minutes ago
gradlew	Initial commit 4 minutes ago
gradlew.bat	Initial commit 4 minutes ago
settings.gradle	Initial commit 4 minutes ago

Goal 5: Setup Jenkins server

- Install Jenkins <https://jenkins.io/download/>
 - For Ubuntu, follow the steps here.
 - <https://pkg.jenkins.io/debian-stable/>
- After finished, Jenkins service will always start at boot time and will be available at <http://localhost:8080>

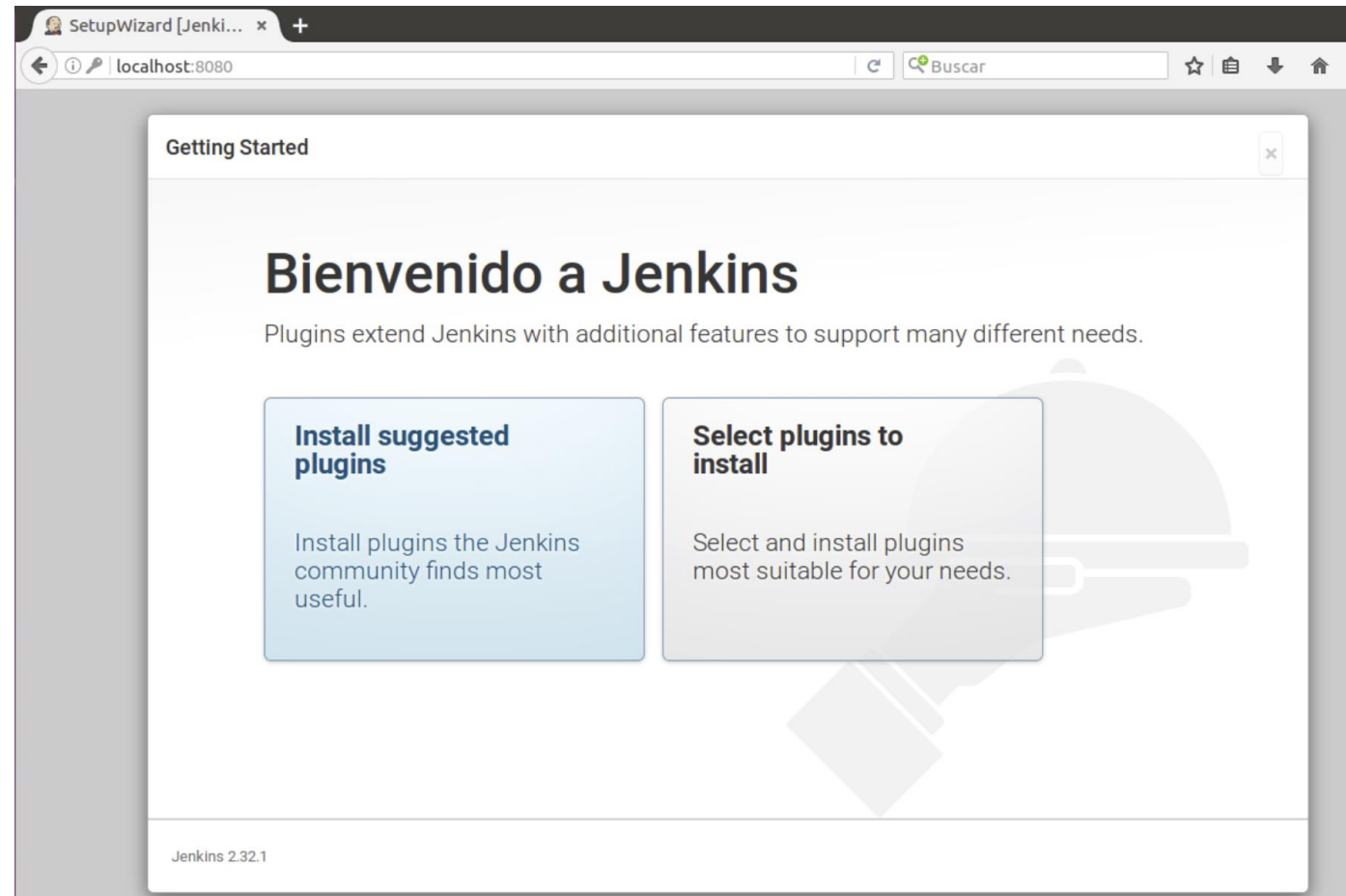
Goal 5: Setup Jenkins server

- You'll see this page for the first time
- `$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword`
- Paste the code shown



Goal 5: Setup Jenkins server

- Install suggested plugins



Goal 5: Setup Jenkins server

- Create a Jenkins job
 - Select Free-style project

Goal 5: Setup Jenkins server

- Set project url (without .git)

The screenshot shows the Jenkins 'General' configuration page. The tabs at the top are: General, Job Notifications, 原始碼管理, 建置觸發程序, 建置環境, 建置, and 建置後動作. The 'General' tab is active. Below the tabs, there is a '描述' (Description) section with a large text area. Below that, the 'GitHub project' checkbox is checked. The 'Project url' field is highlighted with a blue border and contains the text 'https://github.com/jeffwang0516/JavaTest/'. To the right of the 'Project url' field is a help icon. At the bottom right, there is a button labeled '進階...' (Advanced...).

Goal 5: Setup Jenkins server

- Set git repository (with .git)

原始碼管理

☐ 無
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

儲存庫瀏覽器

Goal 5: Setup Jenkins server

- Set git repository (with .git)

原始碼管理

☐ 無
☒ Git

Repositories

Repository URL

Credentials [Add](#)

[進階...](#)

[Add Repository](#)

Branches to build

Branch Specifier (blank for 'any')

[Add Branch](#)

儲存庫瀏覽器

Goal 5: Setup Jenkins server

- Go to “Build Triggers” section
 - Check Poll Scm and enter five stars with spaces (* * * * *)
 - It means “every minute” for [cron time](#) process. So our jenkins server will check github every minute, if there is any change in the repo it will trigger the job and tests will start.
- You can set trigger time however you want using cron time.

Goal 5: Setup Jenkins server

- Go to “Build Options” section
 - >> Add build step >> Invoke Gradle script
 - Select “Use Gradle Wrapper”
 - Input the script into “Tasks”

建置

Invoke Gradle script

☐ Invoke Gradle

☒ Use Gradle Wrapper

Make gradlew executable ☐

Wrapper location

Tasks

clean
test

進階...

新增建置步驟 ▾

Goal 5: Setup Jenkins server

- Go to “Post-Build Actions” section
 - Add “Publish JUnit test result report”

建置後動作

發佈 JUnit 測試結果報告

測試報告 XML

build/test-results/**/*.xml

用來指定產生的 XML 報告原始檔的 [Fileset 'includes'](#) 設定，例如 'myproject/target/test-reports/*.xml'。檔案集的基底目錄就是[工作區目錄](#)。

☐ 保留完整的標準輸出及標準錯誤內容

Health report amplification factor

1.0

1% failing tests scores as 99% health. 5% failing tests scores as 95% health

Allow empty results

☐ Do not fail the build on empty test results

Goal 6:

Automatic fetch and build your java project from Github on Jenkins

- Apply the settings and your done!!!
- See the build history and results

The screenshot shows the Jenkins web interface for a project named 'JavaTest'. The left sidebar contains navigation links: '回到儀表板', '狀態', '變更', '工作目錄', '馬上建置', '刪除專案', '組態', 'GitHub', 'Rename', and 'Embeddable Build Status'. The main content area is titled '專案 JavaTest' and includes links for '工作區', '最近變更', and '最新測試結果 (無失敗)'. Below these is a '永久連結' section. At the bottom, there is a '建置歷程' (Build History) table with columns for build number, time, and status. The table lists builds #1 through #6, all of which are successful. The build #6 is highlighted with a red box. Below the table are links for 'RSS 摘要: 全部' and 'RSS 摘要: 失敗'.

Build Number	Time	Status
#6	2018/9/26 下午 3:49	Success
#5	2018/9/26 下午 3:48	Success
#4	2018/9/26 下午 3:47	Success
#3	2018/9/26 下午 3:46	Success
#2	2018/9/26 下午 3:45	Success
#1	2018/9/26 下午 3:44	Success

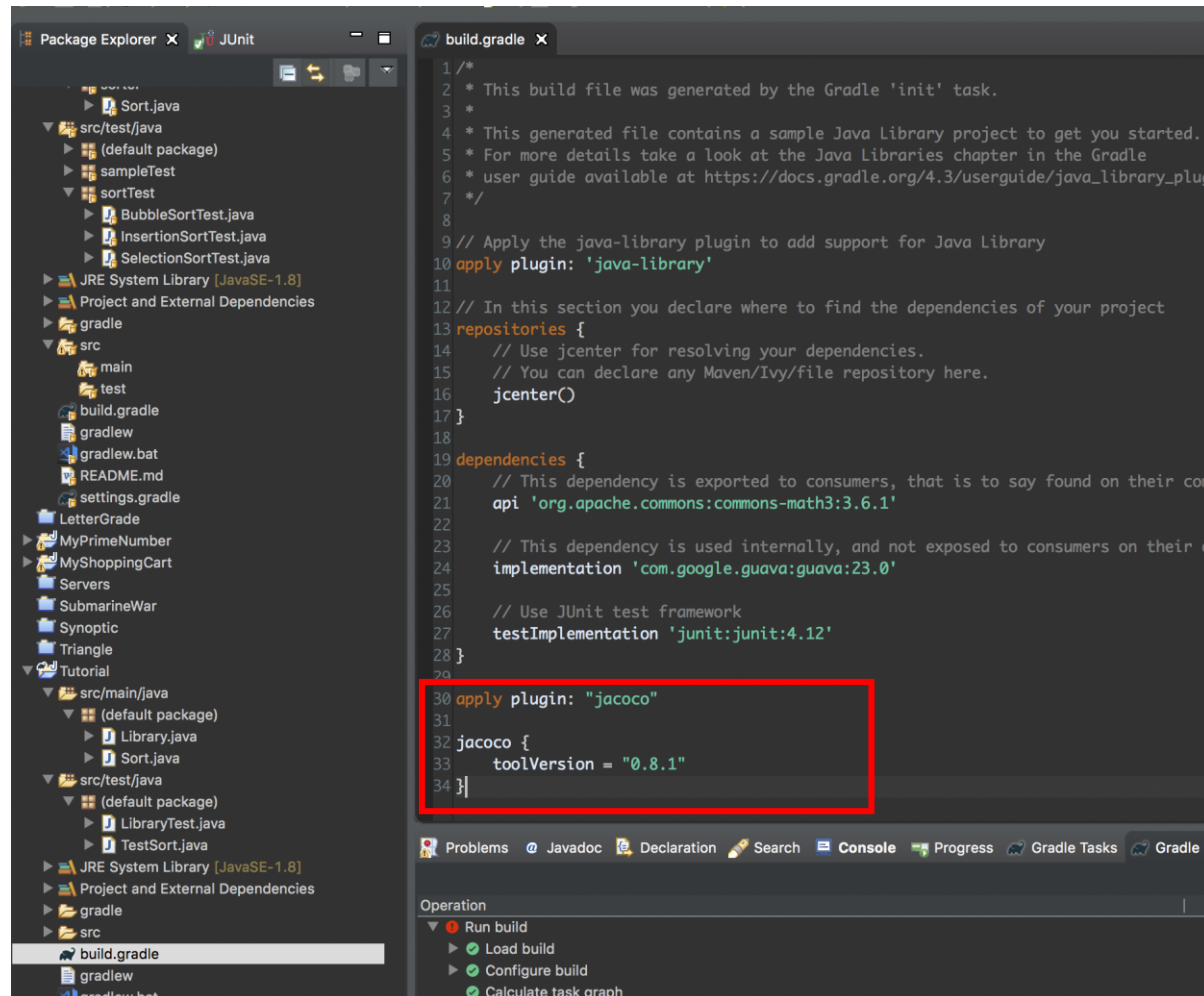
>> Click “#6” to see details

建構 #6 (2018/9/26 下午 03:49:00)

The screenshot shows the Jenkins build details page for build #6. It includes a 'No changes' message, a '由計時器啟動' (Started by timer) message, and a 'Revision' section showing the commit hash '91b992ee07663e45d885a0ea92e1dbb32981170e' and the branch 'refs/remotes/origin/master'. At the bottom, there is a link for '測試結果 (無失敗)'.

Goal 7: Coverage report with Jacoco plugin

- In your project, Edit build.gradle file (Add line 30~34)



Remember to
push to github
after you
modified your
project

Goal 7: Coverage report with Jacoco plugin

- In Jenkins, add the extension JaCoCo

The screenshot shows the Jenkins web interface. On the left sidebar, the '管理 Jenkins' (Manage Jenkins) link is highlighted with a red rectangle. Below it, the '建置佇列' (Build Queue) and '建置執行程式狀態' (Build Execution Status) sections are visible. The main content area is titled '管理 Jenkins' and lists several configuration options: '設定系統' (Configure System), '設定全域安全性' (Configure Global Security), 'Configure Credentials', 'Global Tool Configuration', and '從磁碟重新載入設定' (Reload configuration from disk). At the bottom, the '管理外掛程式' (Manage External Plugins) link is also highlighted with a red rectangle. The '系統資訊' (System Information) link is visible at the very bottom.

Jenkins

新增作業
使用者
建置歷程
管理 Jenkins
我的視景
Credentials
New View

建置佇列
佇列中沒有建置作業。

建置執行程式狀態
1 閒置
2 閒置

管理 Jenkins

- 設定系統**
設定全域設定及路徑。
- 設定全域安全性**
保護 Jenkins，定義誰可以存取或是使用系統。
- Configure Credentials**
Configure the credential providers and types
- Global Tool Configuration**
Configure tools, their locations and automatic installers.
- 從磁碟重新載入設定**
放棄所有在記憶體裡的資料，全部由檔案系統重新載入。如果您直接修改過設定檔，這個功能就應該蠻有用的。
- 管理外掛程式**
外掛程式是能擴充 Jenkins 功能的程式，您可以在這裡新增、移除、停用或是啟用它們。
- 系統資訊**

Goal 7: Coverage report with Jacoco plugin

- In Jenkins, add the extension JaCoCo



Goal 7: Coverage report with Jacoco plugin

- In Jenkins project
 - Go to “Post-Build” section
 - Add “Record JaCoCo coverage report”
 - Apply

The screenshot shows the 'Record JaCoCo coverage report' configuration page in Jenkins. The page is titled 'Record JaCoCo coverage report' and has a red 'X' icon in the top right corner. It contains several input fields for configuring the coverage report.

Path to exec files (e.g.: **/target//*.exec, **/jacoco.exec)**
Input field: `**/*.exec`

Inclusions (e.g.: **/*.class)
Input field: (empty)

Exclusions (e.g.: **/*Test*.class)
Input field: (empty)

Path to class directories (e.g.: **/target/classDir, **/classes)
Input field: `**/classes`

Path to source directories (e.g.: **/mySourceFiles)
Input field: `**/src/main/java`

Inclusions (e.g.: **/*.java, **/*.groovy, **/*.gs)
Input field: `**/*.java`

Exclusions (e.g.: generated//*.java)**
Input field: (empty)

☐ Disable display of source files for coverage

☐ Change build status according to the thresholds

	Instruction	% Branch	% Complexity	% Line	% Method	% Class
	0	0	0	0	0	0
	0	0	0	0	0	0

☐ Fail the build if coverage degrades more than the delta thresholds

	Instruction	% Branch	% Complexity	% Line	% Method	% Class
	0	0	0	0	0	0

新增建置後動作 ▼

儲存 Apply

Goal 7: Coverage report with Jacoco plugin

- In the next build
 - Extra coverage report is shown!

 建構 #17 (2018/9/26 下午 04:18:56)



No changes.



由使用者 [Jeff Wang](#) 啟動



Revision: 55a78d0675ac658dbc0341756631f076650cb069

- refs/remotes/origin/master



[測試結果](#) (無失敗)



Jacoco - Overall Coverage Summary

INSTRUCTION	63%	
BRANCH	33%	
COMPLEXITY	62%	
LINE	56%	
METHOD	83%	
CLASS	100%	

Final Goal :

Find an external library and design your testcases

1. Find a library (ex. the Sort we use in the demo)
2. Create a Gradle project and write testcases
3. Make sure to put it on github
4. Automatically build and test on Jenkins!

Git tutorial: <https://lee-w.github.io/git-tutorial/>

Score

- $\text{rawScore}(\text{you}) = \text{\#bugs-found} + \text{line-coverage} * \log_2(\text{\#total lines of the library SUT})$
- $\text{projectScore}(\text{you}) = 15 * \text{rawScore}(\text{you}) / (\max_{\text{anyStudent } s} \text{rawScore}(s))$

Submission

- Put everything in one folder named by your student id
- The folder should include:
 1. Your Junit project Github repository URL
 2. Screenshot of your JaCoCo coverage report
 3. rawScore calculate yourself
- Compress to .zip format
- Upload to Ceiba

End