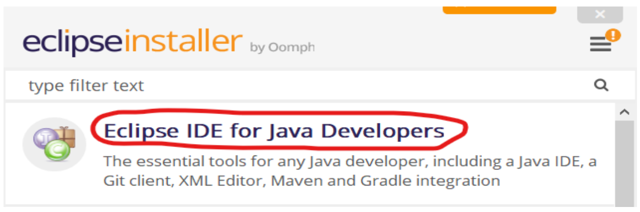
CS5351 Software Engineering
2024/2025 Semester B
Program Testing Exercise: Continuous Integration with Github Project

Overview:
1. Create a local project as the codebase to apply continuous integration (CI).
2. Link the project with Github, which provides the CI feature.
3. Modify the code locally and push it to GitHub in order to trigger CI.
4. **Please upload the result of testing and any necessary screenshots/descriptions on Discussions of Canvas until 6:30 pm, March 25.**

A. Prepare a Maven Project in Eclipse IDE

1. Download and install the newest Eclipse IDE
   a. download link: https://www.eclipse.org/downloads/
   b. Choose the option for JAVA developers.

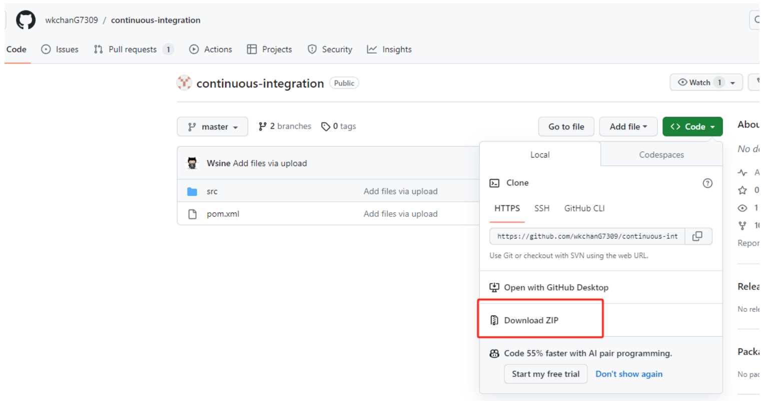


The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration

(Fig.1)

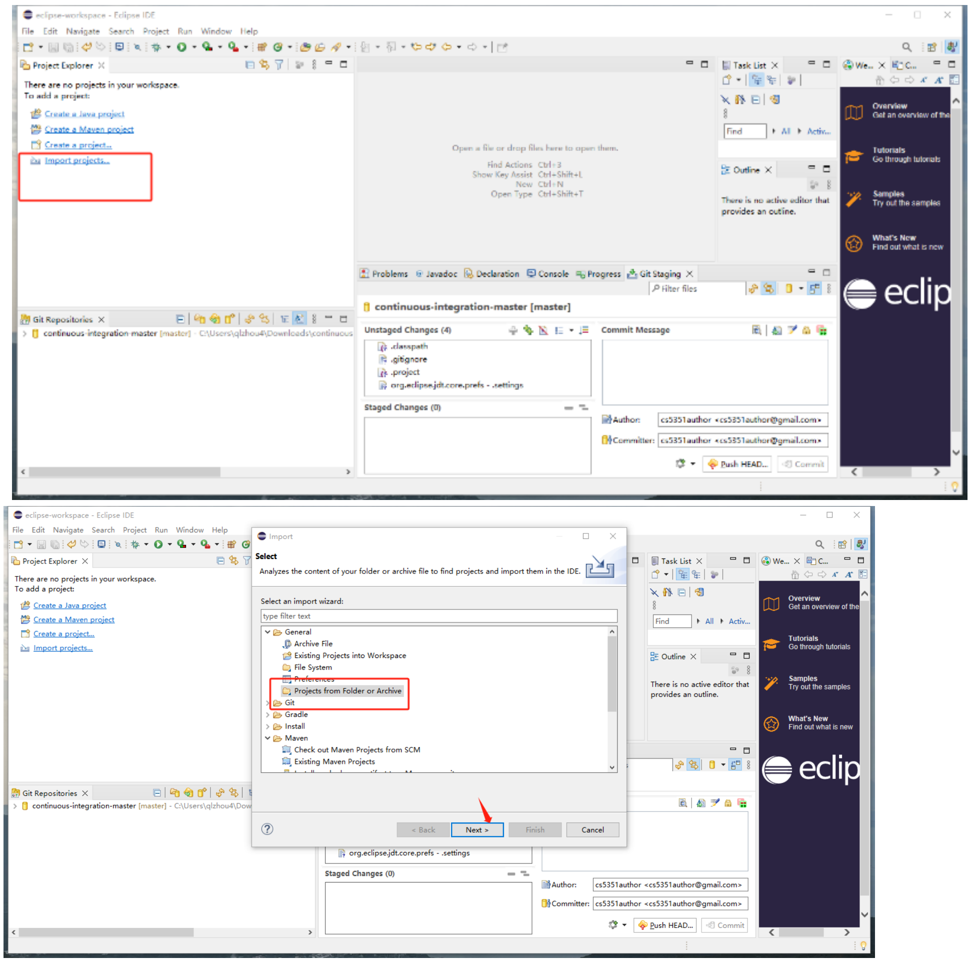
2. Create a new Maven project **on your local computer**

Import Route 1: Use this example project: wkchanG7309/continuous-integration
   a. Download the package: click the link->code->download ZIP



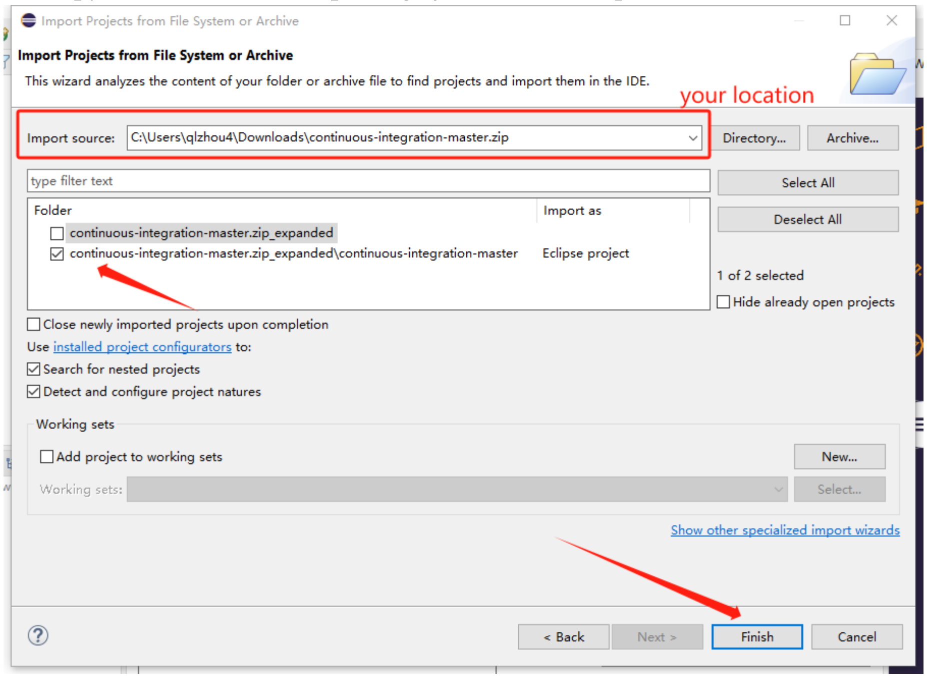
(Fig.2)

   b. Open Eclipse ->import projects


eclipse-workspace - Eclipse IDE
Project Explorer
There are no projects in your workspace.
To add a project:
Create a Java project
Create a Maven project
Create a project...
Import projects...
Git Repositories
continuous-integration-master [master]
Unstaged Changes (4)
.classpath
.gitignore
.project
org.eclipse.jdt.core.prefs - .settings
Staged Changes (0)
Commit Message
Author: cs5351author <cs5351author@gmail.com>
Committer: cs5351author <cs5351author@gmail.com>
Push HEAD...
Commit
Import
Select
Analyzes the content of your folder or archive file to find projects and import them in the IDE.
Select an import wizard:
type filter text
General
Archive File
Existing Projects into Workspace
File System
Preferences
Projects from Folder or Archive
Git
Gradle
Install
Maven
Check out Maven Projects from SCM
Existing Maven Projects
< Back
Next >
Finish
Cancel
Overview
Tutorials
Samples
What's New
eclip

(Fig.3)

c. Copy the location of the package just download, pick folder, and finish.



Import Projects from File System or Archive

Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

your location

Import source: C:\Users\qlzhou4\Downloads\continuous-integration-master.zip

Folder	Import as
☐ continuous-integration-master.zip_expanded	
☑ continuous-integration-master.zip_expanded\continuous-integration-master	Eclipse project

1 of 2 selected

☐ Hide already open projects

☐ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

☑ Detect and configure project natures

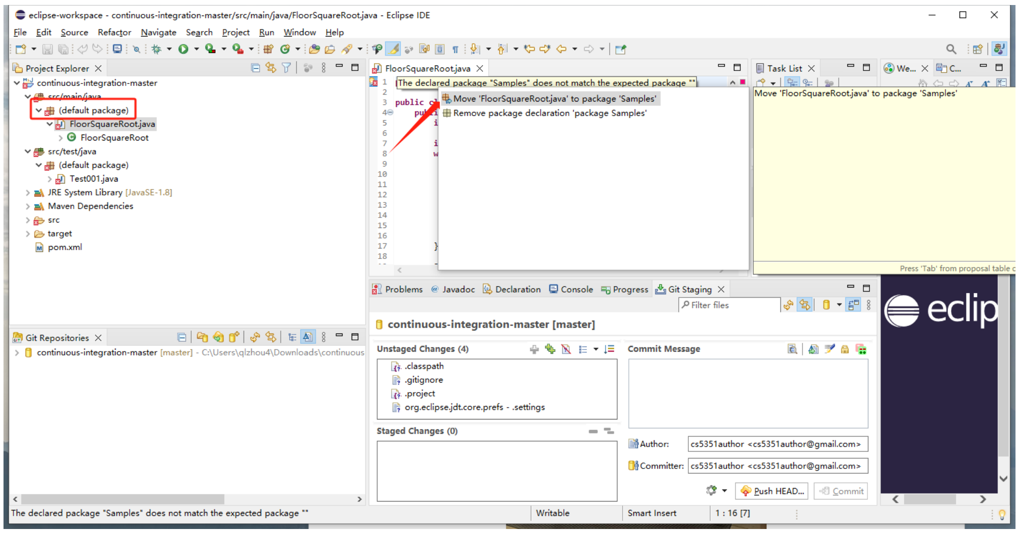
Working sets

☐ Add project to working sets

Working sets:

(Fig.4)

d. click “Move …. to package … ” for both two java files if package error appears.


eclipse-workspace - continuous-integration-master/src/main/java/FloorSquareRoot.java - Eclipse IDE

The declared package "Samples" does not match the expected package ""

Move 'FloorSquareRoot.java' to package 'Samples'

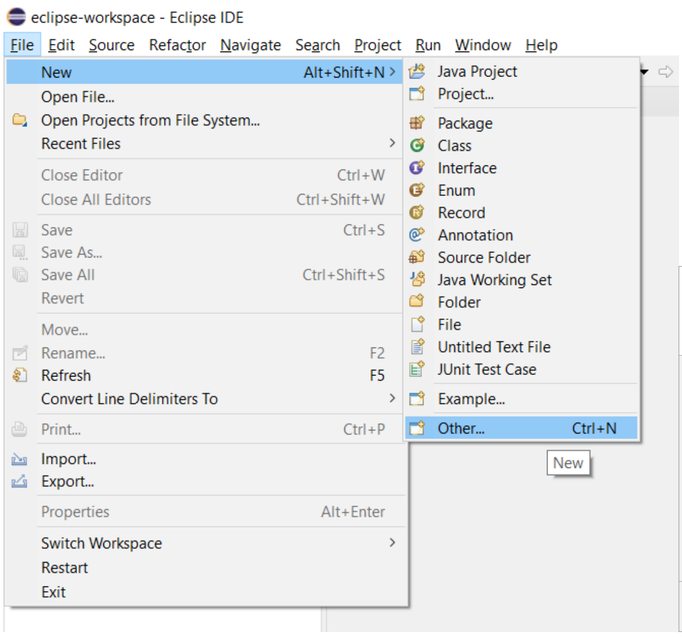
Remove package declaration 'package Samples'

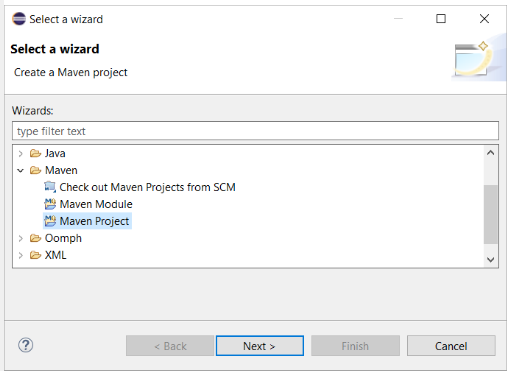
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
continuous-integration-master
src/main/java
(default package)
FloorSquareRoot.java
FloorSquareRoot
src/test/java
(default package)
Test001.java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
pom.xml
Git Repositories
continuous-integration-master [master] - C:\Users\qlzhou4\Downloads\continuous
Problems Javadoc Declaration Console Progress Git Staging
continuous-integration-master [master]
Unstaged Changes (4)
.classpath
.gitignore
.project
org.eclipse.jdt.core.prefs - .settings
Staged Changes (0)
Commit Message
Author: cs5351author <cs5351author@gmail.com>
Committer: cs5351author <cs5351author@gmail.com>
Push HEAD...
Commit
Writable
Smart Insert
1 : 16 [7]

(Fig.5)

Import Route 2:

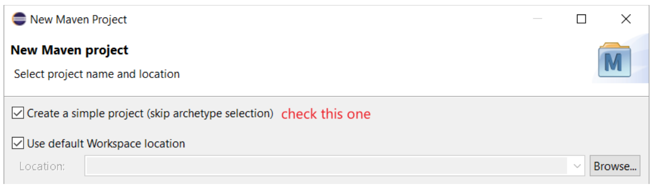
a. Create via File > New > Other > Maven > Maven Project





(Fig.6) (Fig.7)

b. Check the option “Create a simple project (skip archetype selection)”


New Maven Project

New Maven project

Select project name and location

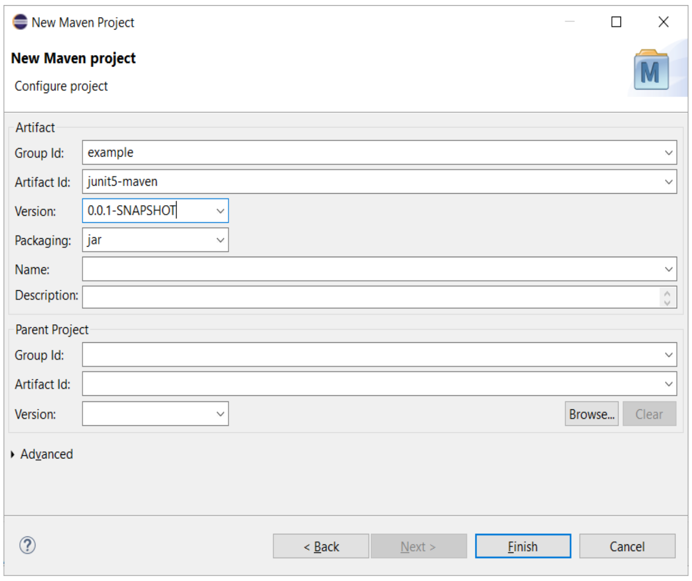
[✓] Create a simple project (skip archetype selection) check this one

[✓] Use default Workspace location

Location:

(Fig.8)

c. Set the Maven project metadata as the following.



New Maven Project

New Maven project

Configure project

Artifact

Group Id: example

Artifact Id: junit5-maven

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

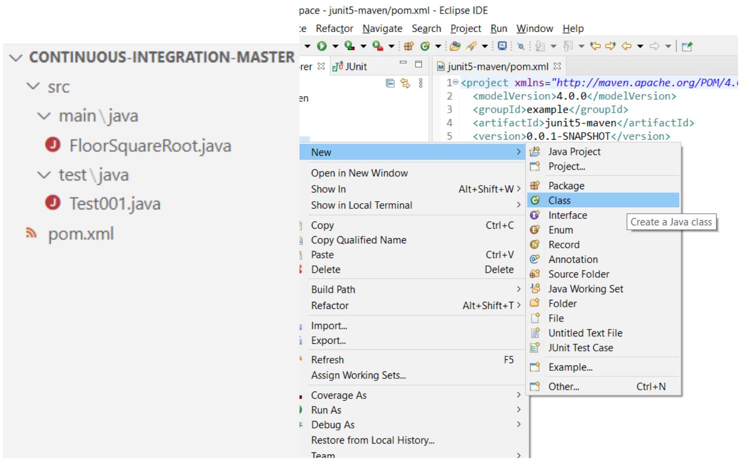
(Fig.9)

d. use the “ctrl+a” to select all the code and copy, the stucture is in fig 6. ROUTE 2 finish.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/PO
M/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XML
Schema-instance"
     xsi:schemaLocation="http://maven.apac
he.org/POM/4.0.0
     http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>example</groupId>
  <artifactId>junit5-maven</artifactId>
  <version>1.0</version>

  <properties>
    <java.version>1.8</java.version>
    <junit-jupiter.version>5.6.2</junit-
jupiter.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-
engine</artifactId>
      <version>${junit-
jupiter.version}</version>
      <scope>test</scope>
    </dependency>
    <dependency>
```

```
package Samples;

public class FloorSquareRoot {
  public static int floorSqrt(int x) {
    if (x == 0 || x == 1) return x;

    int start = 1, end = x, ans = 0;
    while (start <= end) {
      int mid = (start + end) / 2;
      if (mid * mid == x) return mid;
      if (mid * mid < x) {
        start = mid + 1;
        ans = mid;
      } else {
        end = mid - 1;
      }
    }

    System.out.println("floorSqrt(" + x +
") = " + ans);
    return ans;
  }
}
```

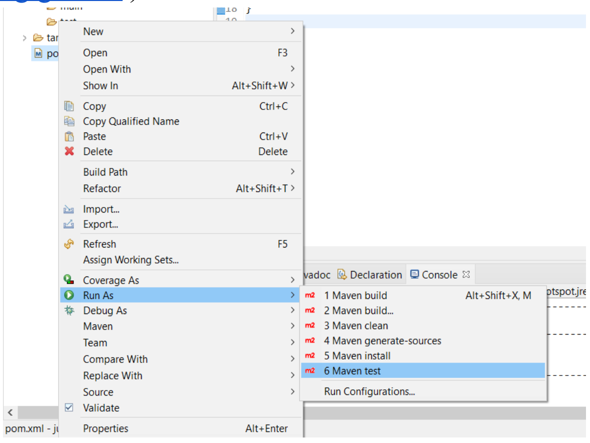
```
package testcases;

import org.junit.jupiter.api.Test
;
import static org.junit.jupiter.a
pi.Assertions.*;

import Samples.*;

public class Test001 {
  @Test
  public void testA() {
    assertEquals(3, FloorSquareRo
ot.floorSqrt(11));
  }

  @Test
  public void testB() {
    assertEquals(4, FloorSquareRo
ot.floorSqrt(20));
  }
}
```



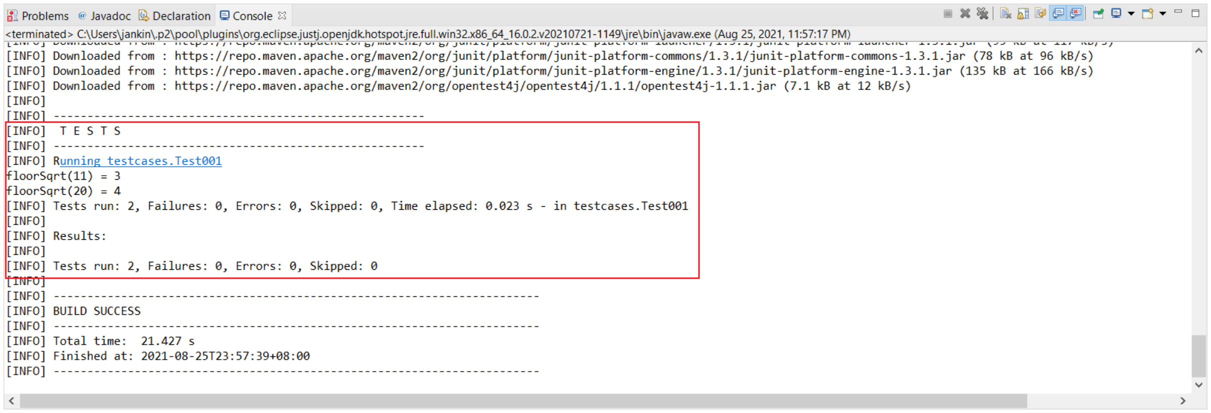
```
<project xmlns="http://maven.apache.org/POM/4.
  <modelVersion>4.0.0</modelVersion>
  <groupId>example</groupId>
  <artifactId>junit5-maven</artifactId>
  <version>0.0.1-SNAPSHOT</version>
```

(Fig.10)

Import Finish. Compile and run your code with tests. (Maven is a package manager that helps you to manage the dependencies and bootstrap the execution. More details can be found at https://maven.apache.org/guides/)



(Fig.11)

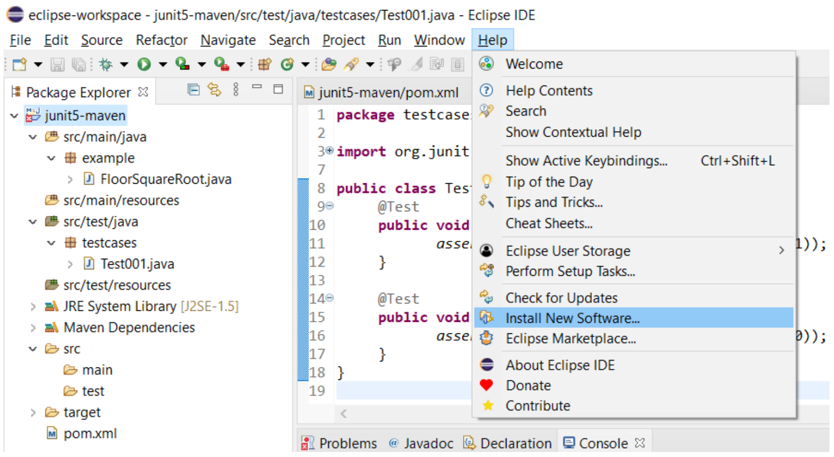


```
<terminated> C:\Users\jankin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Aug 25, 2021, 11:57:17 PM)
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-commons/1.3.1/junit-platform-commons-1.3.1.jar (78 kB at 96 kB/s)
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-engine/1.3.1/junit-platform-engine-1.3.1.jar (135 kB at 166 kB/s)
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/opentest4j/opentest4j/1.1.1/opentest4j-1.1.1.jar (7.1 kB at 12 kB/s)
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running testcases.Test001
floorSqrt(11) = 3
floorSqrt(20) = 4
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.023 s - in testcases.Test001
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  21.427 s
[INFO] Finished at: 2021-08-25T23:57:39+08:00
[INFO] ------------------------------------------------------------------------
```

(Fig.12)

B. Upload your project to GitHub

1. Install eGit in Eclipse IDE.
   a. Since Eclipse has no built-in git client, we need to install a git client in order to communicate with the Github service.
   b. Install via Help > Install New Software…

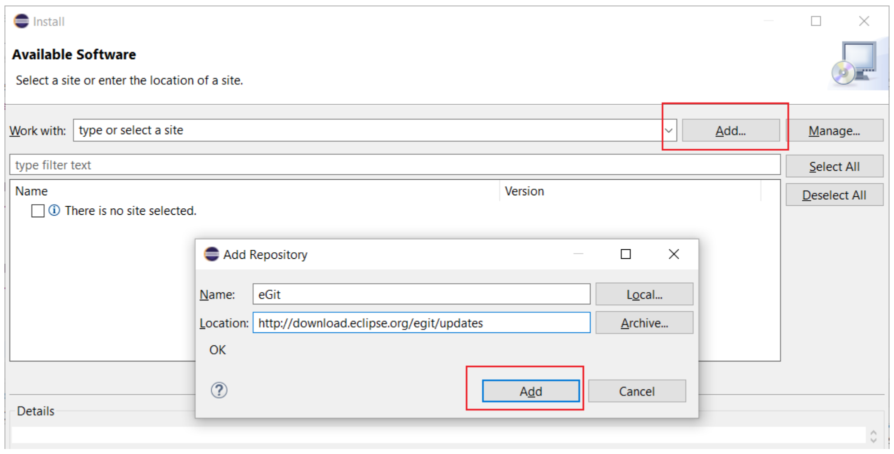


(Fig.13)

c. Click the “Add” button to add a source repository of eGit plugin.

   Name: eGit

   Location: http://download.eclipse.org/egit/updates


Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

type filter text

Name | Version

There is no site selected.

Add Repository

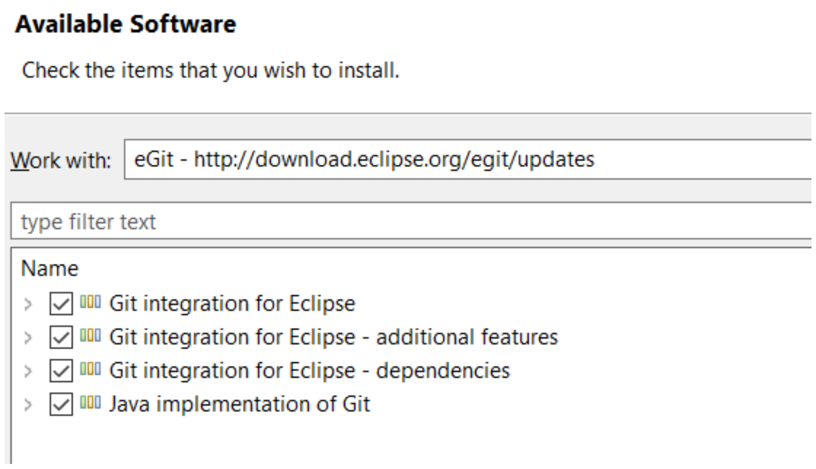
Name: eGit

Location: http://download.eclipse.org/egit/updates

OK

(Fig.14)

d. Check up all the components of eGit to install.


Available Software

Check the items that you wish to install.

Work with: eGit - http://download.eclipse.org/egit/updates

type filter text

Name

- ✓ Git integration for Eclipse
- ✓ Git integration for Eclipse - additional features
- ✓ Git integration for Eclipse - dependencies
- ✓ Java implementation of Git

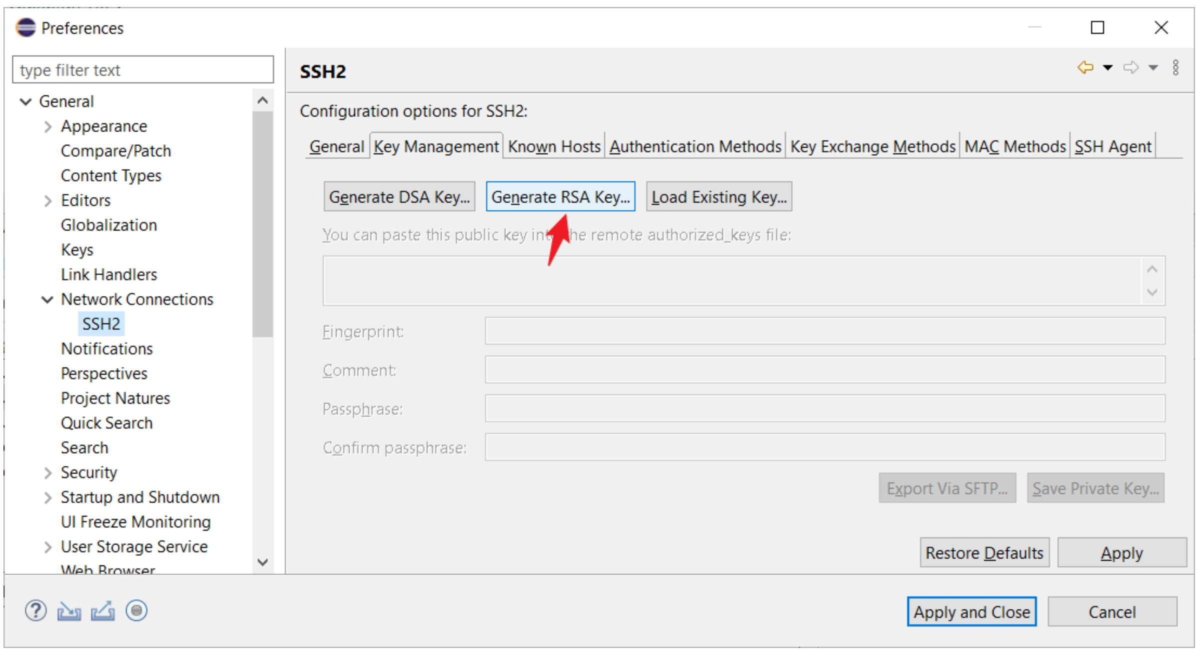
(Fig.15)

2. Setup RSA keys for your SSH connection

   (The git protocol requires connections via HTTP or SSH. For ease of entering the username and password every time pushing the code, we choose the **SSH** connection. SSH protocol requires asymmetric cryptography to communicate. Here we choose **RSA** algorithm.)

   a. Generate via Window > Preferences > General > Network Connections > SSH2 > Key Management
      Note: leave the Passphrase field to be empty to avoid entering passphrase every time you use the private key.

b. Generate RSA Key… > Copy the public key in the box > Save Private Key… > Apply and Close



(Fig.16)

c. Setup the SSH keys in your Github account

i. reference: [Adding a new SSH key to your GitHub account]()

ii. open https://github.com in your browser.

iii. click the drop-down menu of your favicon and click "Settings".

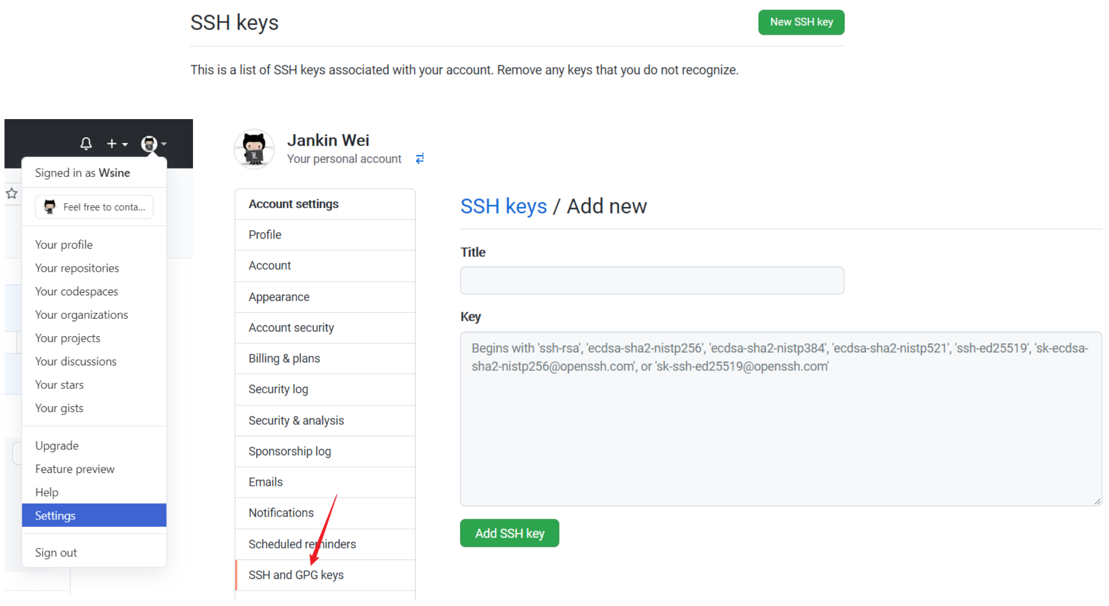
iv. click "SSH and GPG keys" on the left list.

v. click the "New SSH key" green button.

vi. fill in the title with any name that could be identified to you.

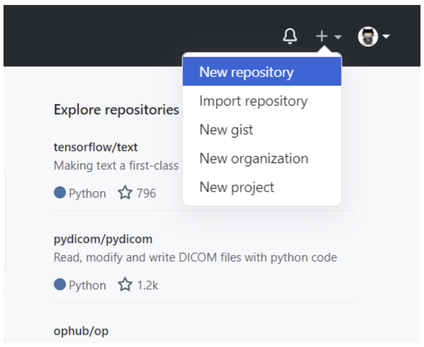
vii. fill in the key with the public that is just copied.

viii. click "Add SSH key" button.

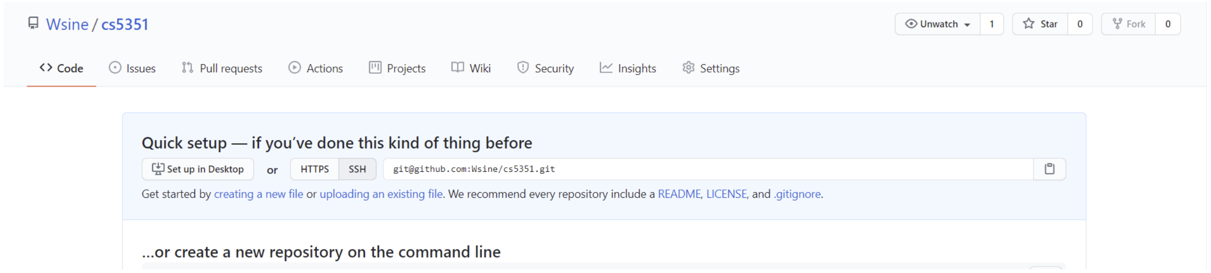


(Fig.17)

3. Use the Github account registered in Week3 and create a new repository.
   a. There are two addresses that represent your repository. We choose the **SSH one** in our tutorial. e.g., git@github.com:Wsine/cs5351.git



(Fig.18)



Quick setup — if you've done this kind of thing before

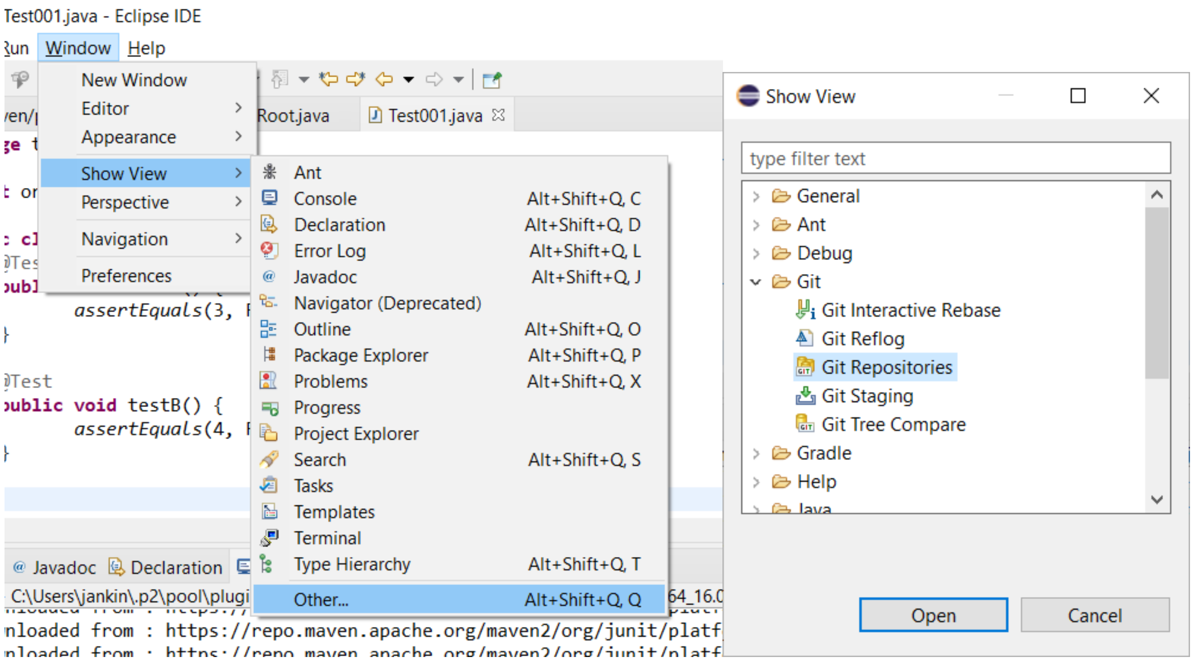
```
git@github.com:Wsine/cs5351.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

(Fig.19)

4. Configure SSH connection with eGit to your Github repository
   a. Come back to Eclipse IDE
   b. Open Git repository pane via Window > Show View > Other… > Git > Git Repositories



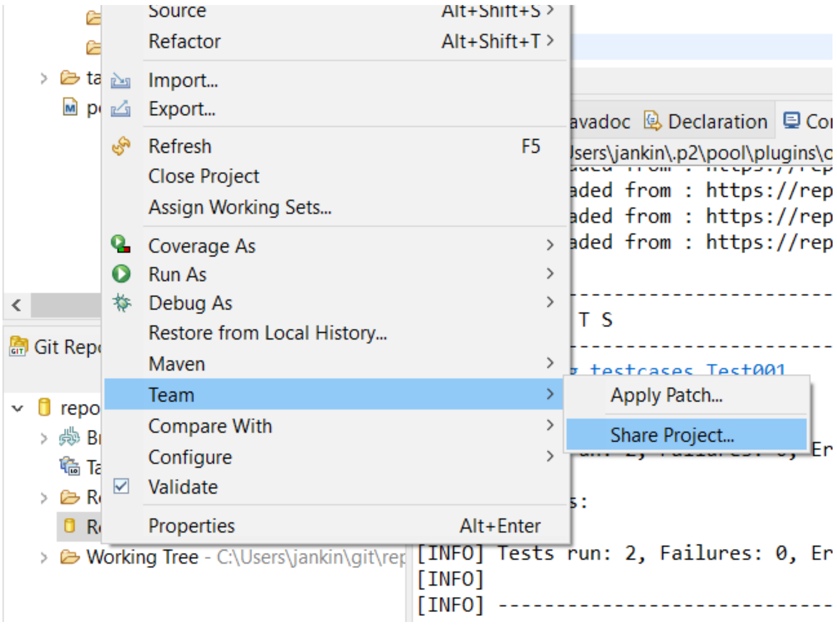
```
        assertEquals(3, 
@Test
public void testB() {
        assertEquals(4, 
}
```

```
C:\Users\jankin\.p2\pool\plugi
nloaded from : https://repo.maven.apache.org/maven2/org/junit/platf
```

(Fig.20)

c. Create a new local Git repository under your current JAVA project.

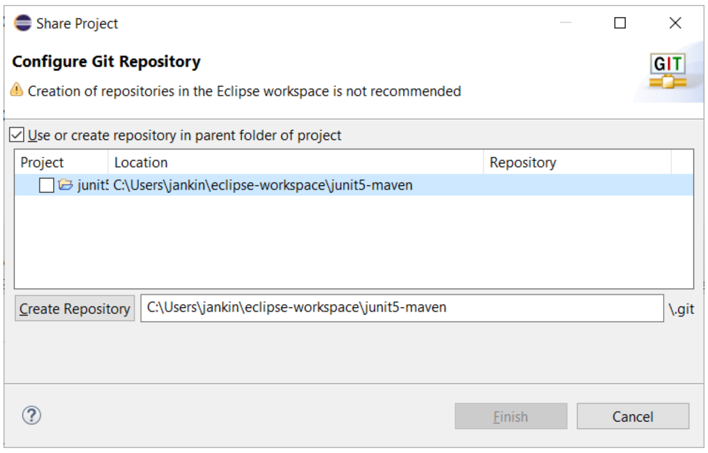
   i. Right click your project and select Team > Share Project…



```
Isers\jankin\.p2\pool\plugins\c
aded from : https://rep
aded from : https://rep
aded from : https://rep
---------------------------
T S
---------------------------
testcases.Test001
s:
[INFO] Tests run: 2, Failures: 0, Er
[INFO]
[INFO] ---------------------------
```

(Fig.21)

   ii. Check up the "Use or create repository..."option > Create Repository > Finish


Share Project

Configure Git Repository

⚠ Creation of repositories in the Eclipse workspace is not recommended

☑ Use or create repository in parent folder of project

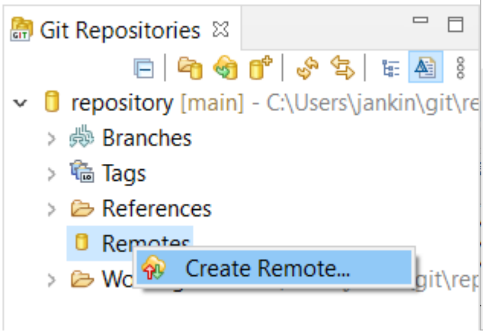
Project	Location	Repository
☐ junit!	C:\Users\jankin\eclipse-workspace\junit5-maven	

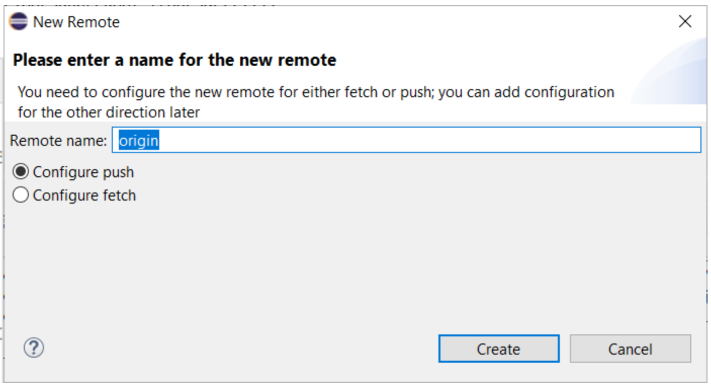
Create Repository C:\Users\jankin\eclipse-workspace\junit5-maven \.git

(Fig.22)

d. Create a new remote to link with your Github repository. (Display may be different according to your Eclipse version)

   i. Right click Remotes in the "Git Repositories" pane and click "Create Remote".
   ii. Fill in the remote name with 'origin'.
   iii. Fill in the URI that is acquired in Fig. 10. The rest fields are filled automatically.
   iv. There is no password in SSH protocol. Leave it blank.
   v. The user field is 'git' as it is the default user in Github platform.




New Remote

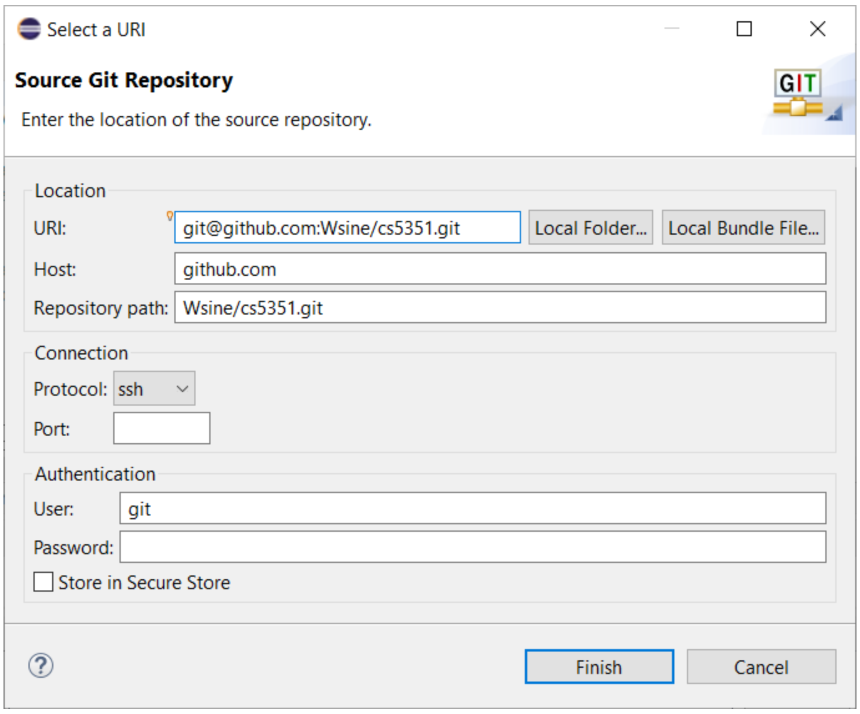
Please enter a name for the new remote

You need to configure the new remote for either fetch or push; you can add configuration for the other direction later

Remote name: origin

[✓ Configure push]
Configure fetch

(Fig.23) (Fig.24)


Select a URI

Source Git Repository

Enter the location of the source repository.

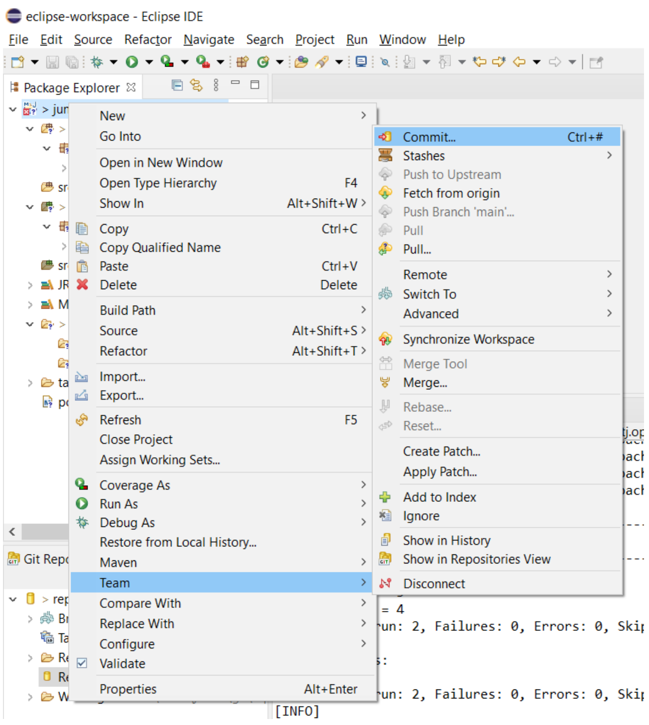
Location
URI: git@github.com:Wsine/cs5351.git
Host: github.com
Repository path: Wsine/cs5351.git

Connection
Protocol: ssh
Port:

Authentication
User: git
Password:
Store in Secure Store

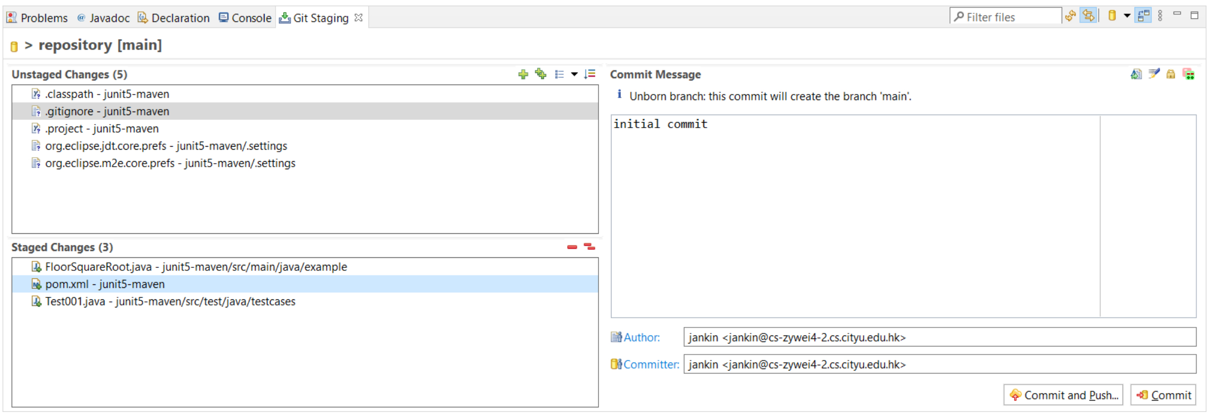
(Fig.25)

5. Upload your local code to the remote Github repository.
   a. Commit the code first, only select the necessary code files to upload, and then push



```
= 4
un: 2, Failures: 0, Errors: 0, Skip
:
un: 2, Failures: 0, Errors: 0, Skip
[INFO]
```

(Fig.26)



Unstaged Changes (5)
- .classpath - junit5-maven
- .gitignore - junit5-maven
- .project - junit5-maven
- org.eclipse.jdt.core.prefs - junit5-maven/.settings
- org.eclipse.m2e.core.prefs - junit5-maven/.settings

Staged Changes (3)
- FloorSquareRoot.java - junit5-maven/src/main/java/example
- pom.xml - junit5-maven
- Test001.java - junit5-maven/src/test/java/testcases

Commit Message

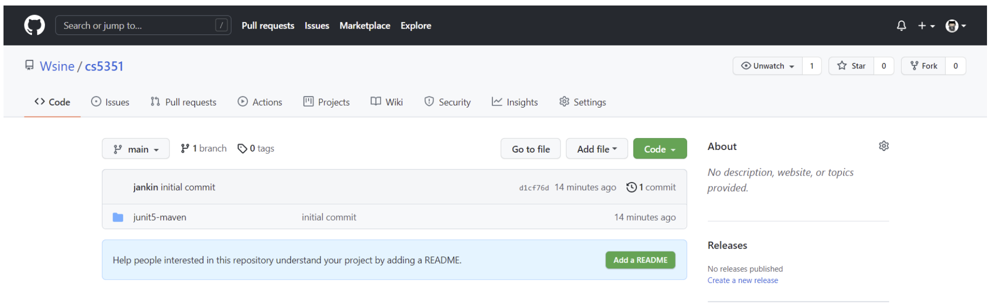
Unborn branch: this commit will create the branch 'main'.

```
initial commit
```

Author: jankin <jankin@cs-zywei4-2.cs.cityu.edu.hk>

Committer: jankin <jankin@cs-zywei4-2.cs.cityu.edu.hk>

(Fig.27)



jankin initial commit	d1cf76d 14 minutes ago	1 commit
junit5-maven	initial commit	14 minutes ago

Help people interested in this repository understand your project by adding a README.

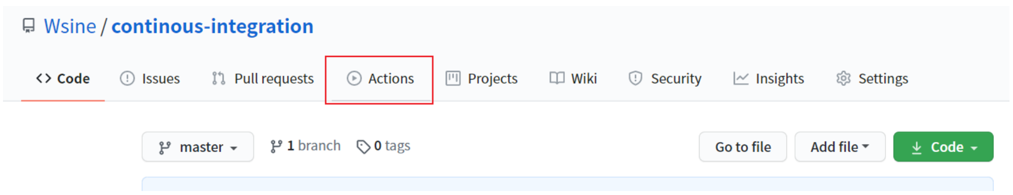
No description, website, or topics provided.

No releases published
Create a new release

(Fig.28)

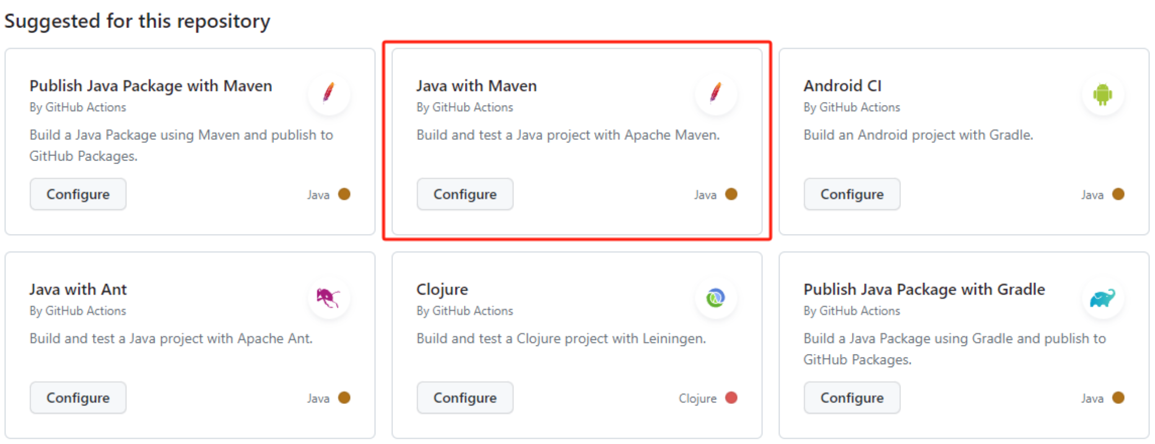
C. Create Github actions workflow (Continuous Integration)

1. Create Github Action
   a. Switch to the [Actions] menu item.



(Fig.29)

b. Press the [Set up this workflow] button with card title “Java with Maven”


Suggested for this repository

Publish Java Package with Maven
By GitHub Actions
Build a Java Package using Maven and publish to GitHub Packages.
Configure Java

Java with Maven
By GitHub Actions
Build and test a Java project with Apache Maven.
Configure Java

Android CI
By GitHub Actions
Build an Android project with Gradle.
Configure Java

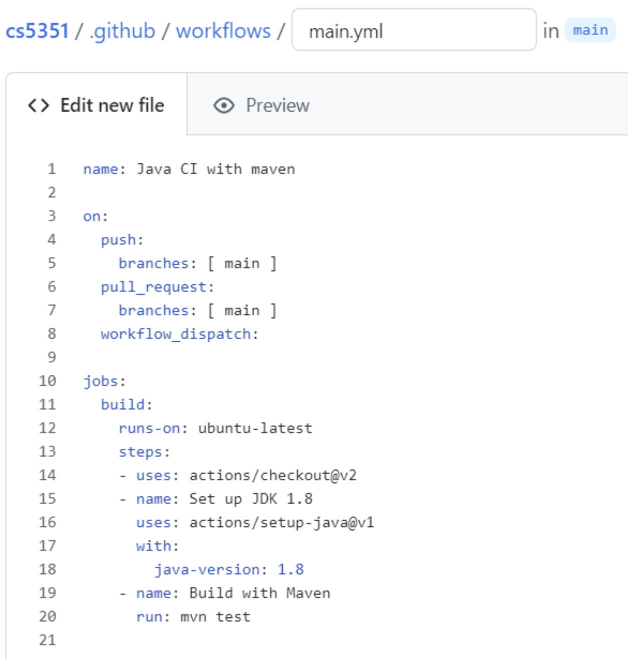
Java with Ant
By GitHub Actions
Build and test a Java project with Apache Ant.
Configure Java

Clojure
By GitHub Actions
Build and test a Clojure project with Leiningen.
Configure Clojure

Publish Java Package with Gradle
By GitHub Actions
Build a Java Package using Gradle and publish to GitHub Packages.
Configure Java

(Fig.30)

c. Modify the file content as the below image shown.

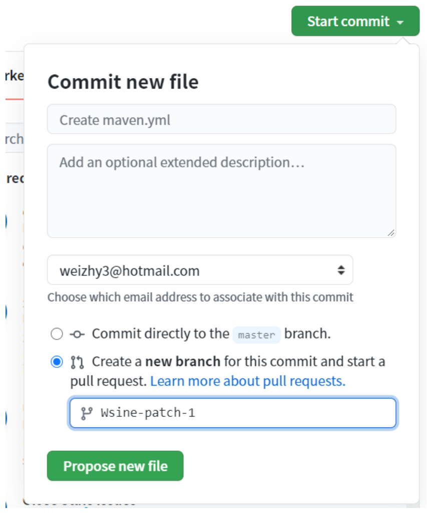


```yaml
name: Java CI with maven

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]
  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 1.8
      uses: actions/setup-java@v1
      with:
        java-version: 1.8
    - name: Build with Maven
      run: mvn test

```

```yaml
name: Java CI with maven

on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]
  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 1.8
      uses: actions/setup-java@v1
      with:
        java-version: 1.8
    - name: Build with Maven
      run: mvn test
```

(Fig.31)

d. Press the [Start commit] button, choose create a new branch option, and press [Propose new file] button.


Start commit

Commit new file

Create maven.yml

Add an optional extended description…

weizhy3@hotmail.com

Choose which email address to associate with this commit

( ) Commit directly to the `master` branch.

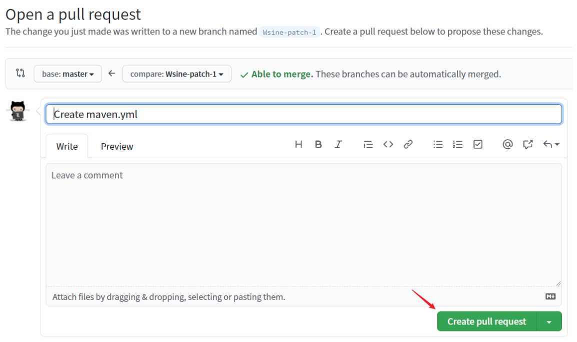
(●) Create a **new branch** for this commit and start a pull request. Learn more about pull requests.

Wsine-patch-1

Propose new file

(Fig.32)

e. Press the [Create pull request] button


Open a pull request

The change you just made was written to a new branch named `Wsine-patch-1`. Create a pull request below to propose these changes.

base: master ← compare: Wsine-patch-1 ✓ **Able to merge.** These branches can be automatically merged.

Create maven.yml

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

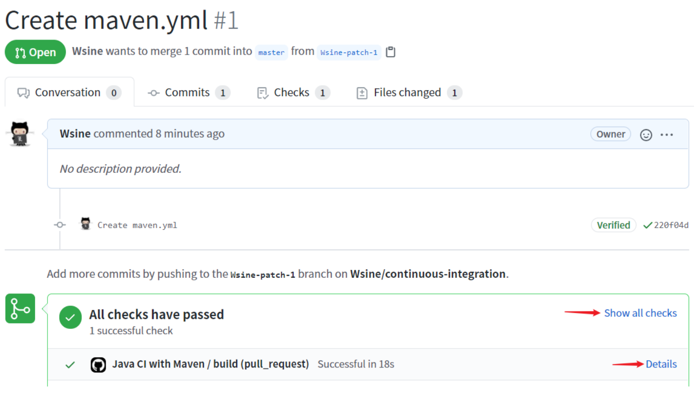
Create pull request

(Fig.33)

2. Inspect the process of Continuous Integration

a. Wait a minute to let the CI finish.

b. Click [Show all checks] link and then [Details] link.


Create maven.yml #1

Open **Wsine** wants to merge 1 commit into `master` from `Wsine-patch-1`

Conversation 0 | Commits 1 | Checks 1 | Files changed 1

Wsine commented 8 minutes ago — Owner

No description provided.

Create maven.yml — Verified ✓220f04d

Add more commits by pushing to the **Wsine-patch-1** branch on **Wsine/continuous-integration**.

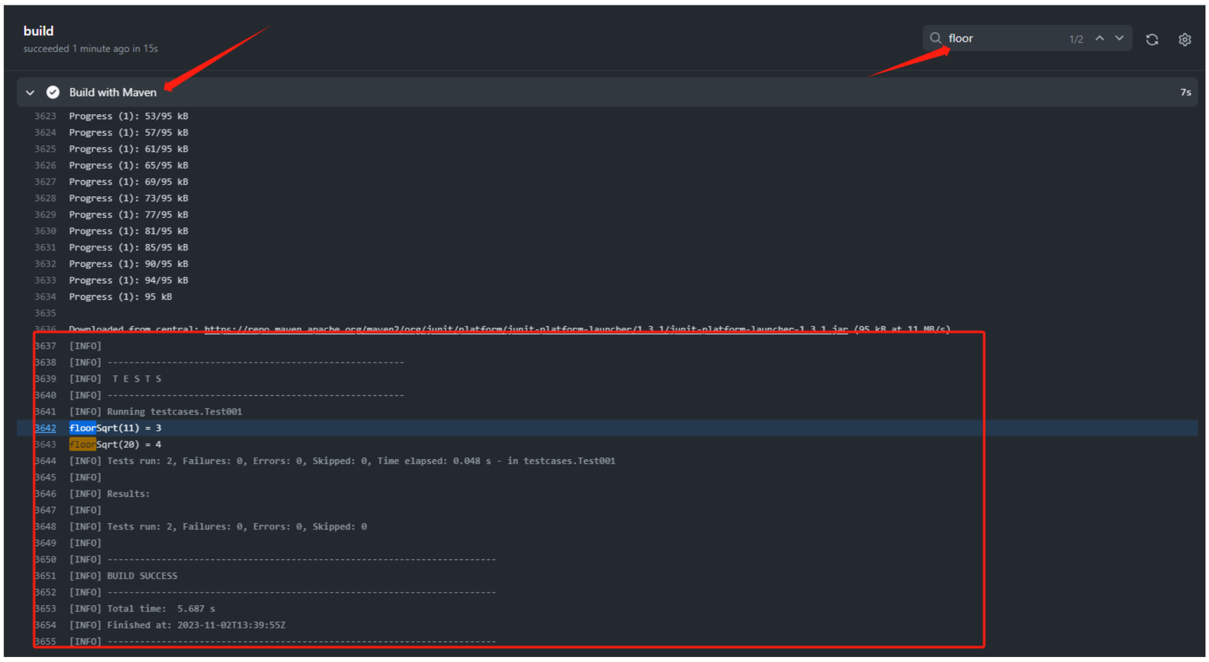
All checks have passed

1 successful check — Show all checks

✓ **Java CI with Maven / build (pull_request)** Successful in 18s — Details

(Fig.34)

c. Check the result of testing

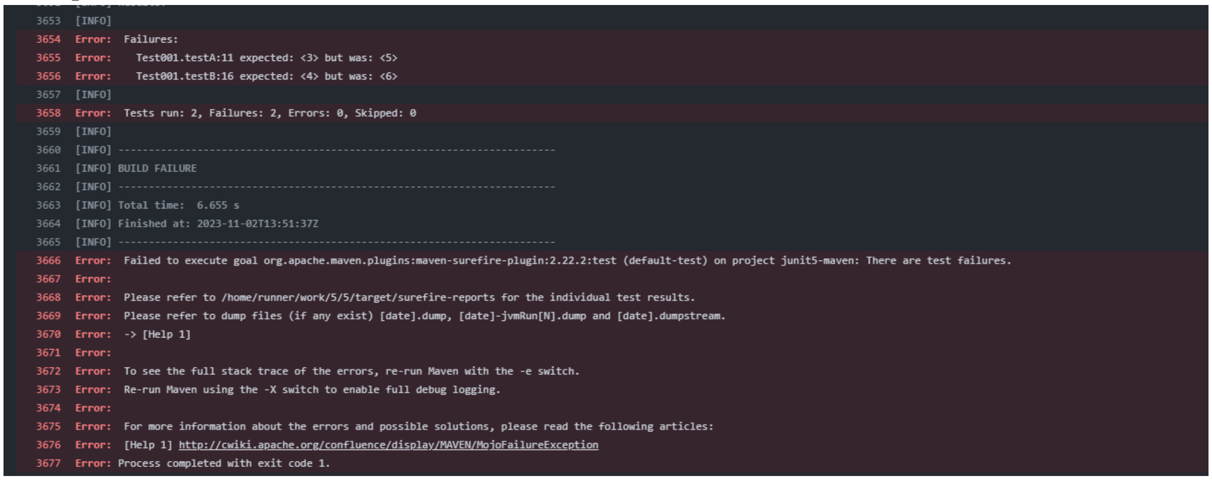


```
Progress (1): 53/95 kB
Progress (1): 57/95 kB
Progress (1): 61/95 kB
Progress (1): 65/95 kB
Progress (1): 69/95 kB
Progress (1): 73/95 kB
Progress (1): 77/95 kB
Progress (1): 81/95 kB
Progress (1): 85/95 kB
Progress (1): 90/95 kB
Progress (1): 94/95 kB
Progress (1): 95 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-launcher/1.3.1/junit-platform-launcher-1.3.1.jar (95 kB at 11 MB/s)
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running testcases.Test001
floorSqrt(11) = 3
floorSqrt(20) = 4
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.048 s - in testcases.Test001
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.687 s
[INFO] Finished at: 2023-11-02T13:39:55Z
[INFO] ------------------------------------------------------------------------
```

(Fig.35)

d. Merge this pull request as what we did in Week 3.
e. Thereafter, every time you commit new changes, it will automatically trigger the CI.

3. Exercise with Continuous Integration.
   a. try to alter the code with a bug that would fail the test cases in the Github project, submit the code to a **new branch**, make a pull request, and inspect the CI.
   Example:



```
[INFO]
Error:  Failures:
Error:    Test001.testA:11 expected: <3> but was: <5>
Error:    Test001.testB:16 expected: <4> but was: <6>
[INFO]
Error:  Tests run: 2, Failures: 2, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.655 s
[INFO] Finished at: 2023-11-02T13:51:37Z
[INFO] ------------------------------------------------------------------------
Error:  Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.22.2:test (default-test) on project junit5-maven: There are test failures.
Error:
Error:  Please refer to /home/runner/work/5/5/target/surefire-reports for the individual test results.
Error:  Please refer to dump files (if any exist) [date].dump, [date]-jvmRun[N].dump and [date].dumpstream.
Error:  -> [Help 1]
Error:
Error:  To see the full stack trace of the errors, re-run Maven with the -e switch.
Error:  Re-run Maven using the -X switch to enable full debug logging.
Error:
Error:  For more information about the errors and possible solutions, please read the following articles:
Error:  [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
Error: Process completed with exit code 1.
```

(Fig.36)

      i. Do not merge the erroneous pull request.
   b. try to fix the bug on the buggy branch and inspect the retriggered CI.
      i. If all the tests are passed, merge the pull request.

*** END ***