

School of Information Technologies
Faculty of Engineering & IT

ASSIGNMENT/PROJECT COVERSHEET - GROUP ASSESSMENT

Unit of Study: SOFT2412 Agile Software Development Practices

Assignment name: Group Project Assignment 1 – Tools for Agile Software Development

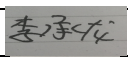
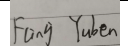
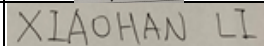
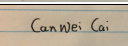
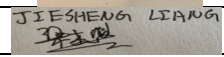
Tutorial time: Thu 6p.m. - 8p.m. R18G3 **Tutor name:** Muhit Saleh Anik

DECLARATION

We the undersigned declare that we have read and understood the *University of Sydney Student Plagiarism: Coursework Policy and Procedure*, and except where specifically acknowledged, the work contained in this assignment/project is our own work, and has not been copied from other sources or been previously submitted for award or assessment.

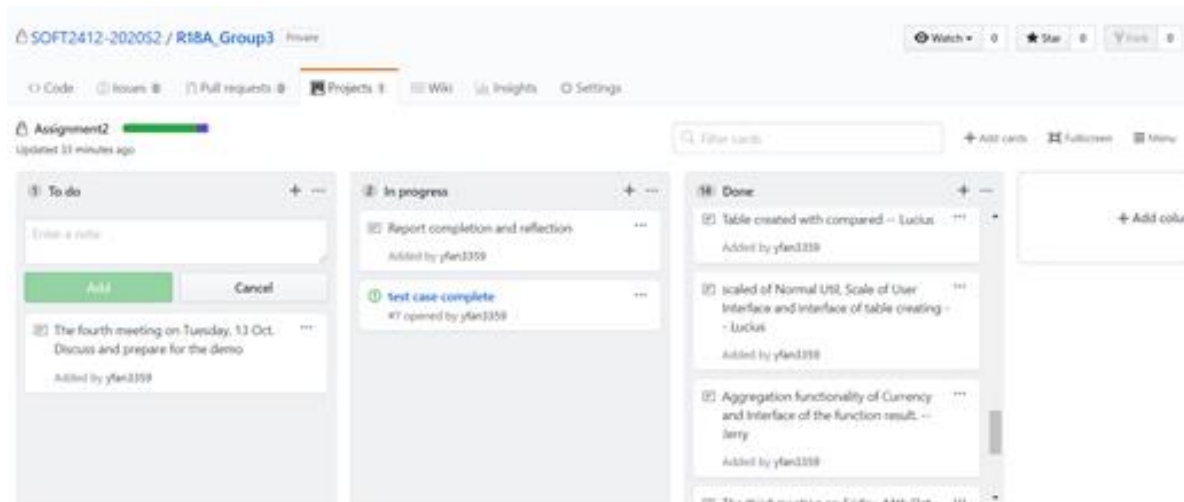
We understand that failure to comply with the *Student Plagiarism: Coursework Policy and Procedure* can lead to severe penalties as outlined under Chapter 8 of the *University of Sydney By-Law 1999* (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

We realise that we may be asked to identify those portions of the work contributed by each of us and required to demonstrate our individual knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

Project team members				
Student name	Student ID	Participated	Agree to share	Signature
1. Chenglong Li	490035344	Yes	Yes	
2. Yuben Fang	480480835	Yes	Yes	
3. Xiaohan Li	470011746	Yes	Yes	
4. Canwei Cai	490032435	Yes	Yes	
5. Jiesheng Liang	480342832	Yes	Yes	
6.		Yes / No	Yes / No	
7.		Yes / No	Yes / No	
8.		Yes / No	Yes / No	
9.		Yes / No	Yes / No	
10.		Yes / No	Yes / No	

GitHub

On GitHub, we used Kanban Board to make plans, distribute our work, record part of our meeting, which leads to an easier collaboration. It tracks the progress of our work and everyone knows what he needs to do now and what has been done by another person in which case he can continue his next part. If we have issues like someone found the coverage of the test case is not enough to fulfill the requirements which is 75%. Then, he published an issue on the card about test cases to tell collaborators to add more test cases. The usage of Kanban board and issue is shown in picture below.



P1: Kanban board

The Tutor helped us to create the remote repository on GitHub and invite us into the repository. We used the command "git clone" to clone the repository from GitHub and create a local repository with a local master branch. When we finish editing in our local branch, we use "git add" to add files to the stage area. Then, use "git commit" to record changes in the local repository. Finally, use "git push" to send the local repository to the repository on GitHub. After that, another person can use "git pull" to fetch and merge changes from the GitHub to his local repository.

When there are changes in the GitHub repository, we need to fetch and merge from GitHub to our local repository. However, we use "git pull" to combine the functionality of fetch and merge, which automatically get the latest version from GitHub and merge to our local repository.

Another situation we met was conflicts when two people were editing the same file at the same time. When the first person pushes the file he edited, the second person will get a rejection if he pushes the same file shown in P2.

```
$ git push
To https://github.sydney.edu.au/SOFT2412-2020S2/R18A_Group3
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.sydney.edu.au/SOFT2412-2020S2/R18A_Group3'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

P2: push error

In this case, he should pull first. However, there is a conflict happening which is shown in P3.

```
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 7 (delta 0), reused 4 (delta 0), pack-reused 0
Unpacking objects: 100% (7/7), 2.07 KiB | 75.00 KiB/s, done.
From https://github.sydney.edu.au/SOFT2412-2020S2/R18A_Group3
 30d81b1..c881a66  master      -> origin/master
Auto-merging src/test/java/Model/UsermenuTest.java
CONFLICT (content): Merge conflict in src/test/java/Model/UsermenuTest.java
Automatic merge failed; fix conflicts and then commit the result.
```

P3: conflict

When we see our code, the IntelliJ points out the position where conflict happens (show in P4).

```
@Test
<<<<<< HEAD
void TestSummaryfoexchange() {
    PrintStream standardOut = System.out;
    ByteArrayOutputStream outContent = new ByteArrayOutputStream();
    System.setOut(new PrintStream(outContent));
    List<Time> notimes = new ArrayList<>();
    try {
        UserMenu.conversionsummary(sdf.parse("2021-01-01"), sdf.parse("2022-01-01"), "AUD", "USD", times);
    } catch (ParseException e) {
        System.out.println(e);
    }
    assertEquals("expected: 'No exchange rate from AUD to USD between 2021-01-01 and 2022-01-01.\n", outContent.toString());
}

=====
void getEditedCurrenciesTest() {
    List<Currency> edited = UserMenu.getEditedCurrencies(currencies);
    int size = edited.size();
    assertEquals("expected: 4, size", size);
}
>>>>>> c881a66ec69796e8d75dca70afea74f4543de92_
}
```

P4: conflict in code

Here, we need to preserve two methods, above the “=====” line and below “=====” line, in this file. We just need reform the two methods followed by commit and push again (show on P5).

```
@Test
void TestSummaryNoexchanges() {
    PrintStream standardOut = System.out;
    ByteArrayOutputStream outContent = new ByteArrayOutputStream();
    System.setOut(new PrintStream(outContent));
    List<Time> notimes = new ArrayList<>();
    try {
        UserMenu.conversionsSummary(sdf.parse("2021-01-01"), sdf.parse("2022-01-01"), "AUD", "USD", times);
    } catch (ParseException e) {
        System.out.println(e);
    }
    assertEquals("expected: \"No exchange rate from AUD to USD between 2021-01-01 and 2022-01-01.\\n\", outContent.toString());
}

@Test
void getEditedCurrenciesTest(){
    List<Currency>edited = UserMenu.getEditedCurrencies(currencies);
    int size = edited.size();
    assertEquals("expected: 4,size);
}
```

P5: reform the code

Gradle

Gradle is a build automation tool for multi-language software development. It controls the development process in the tasks of compilation and packaging to testing, deployment, and publishing.

- Relevant Gradle commands used

1. Gradle init
2. Gradle build
3. Gradle clean
4. Gradle run
5. Gradle test

- Explanation about the results/outputs obtained from relevant Gradle commands

1. Gradle init

initialise a git repository in local directory.

2. Gradle build

```
JacksonLENGUs-MacBook-Pro:R18A_Group3 jacksonliang$ gradle build

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.1/userguide/command\_line\_interface.html#sec:command\_line\_warnings

BUILD SUCCESSFUL in 638ms
7 actionable tasks: 7 up-to-date
```

3. Gradle clean

```
JacksonLENGUs-MacBook-Pro:R18A_Group3 jacksonliang$ gradle clean

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.1/userguide/command\_line\_interface.html#sec:command\_line\_warnings

BUILD SUCCESSFUL in 593ms
1 actionable task: 1 executed
```

4. Gradle run

```
JacksonLENUGs-MacBook-Pro:R18A_Group3 jacksonliang$ gradle run

> Task :run
Today is 2020-10-10

make a selection
1. normal user
2. admin
3. next day
4. quit
Please enter a selection
<=====--> 75% EXECUTING [2m 17s]
> :run
```

5. Gradle test

```
JacksonLENUGs-MacBook-Pro:R18A_Group3 jacksonliang$ gradle test

> Task :test

Admin_ExchangeTest > simpleadd() PASSED
Admin_ExchangeTest > simpleadd_withreplace() PASSED
Admin_ExchangeTest > miss_both_currency() PASSED
Admin_ExchangeTest > same_currency_name_inexchange() PASSED
Admin_ExchangeTest > miss_to_currency() PASSED
Admin_ExchangeTest > input_inlegal() PASSED
Admin_ExchangeTest > miss_form_currency() PASSED
Admin_ExchangeTest > simpleadd_withsameexhchange() PASSED
Admin_ExchangeTest > simple_error_add() PASSED

ExchangeImplTest > equals() PASSED
ExchangeImplTest > testToString() PASSED
ExchangeImplTest > getFromCurrency() PASSED
ExchangeImplTest > getRate() PASSED
ExchangeImplTest > HashCode() PASSED
ExchangeImplTest > getToCurrency() PASSED

BuilderTest > buildDate() PASSED
BuilderTest > buildCurrency() PASSED
BuilderTest > buildExchange() PASSED
```

```
BuilderTest > buildExchange() PASSED
MonthTest > values() PASSED
MonthTest > valueOf() PASSED
MonthTest > getValue() PASSED
TimeImplTest > getExchange() PASSED
TimeImplTest > getDate() PASSED
TimeImplTest > getYear() PASSED
TimeImplTest > deleteExchange() PASSED
TimeImplTest > addExchange() PASSED
TimeImplTest > getExchanges() PASSED
TimeImplTest > getMonth() PASSED
AddCurrencyTest > addCurrency_normalCase() PASSED
AddCurrencyTest > addCurrency_lotsOfInputsCase() PASSED
TableTest > getDateTest() PASSED
TableTest > displayWithYesterdayDataAdded() PASSED
TableTest > getYesterdayTimeTest() PASSED
TableTest > CompareTest() PASSED
TableTest > displayNormalTest() PASSED
UsermenuTest > TestConvert() PASSED
UsermenuTest > TestSummaryEmptyTime() PASSED
```

```
UsermenuTest > TestSummaryEmptytime() PASSED
UsermenuTest > TestConvertNoExchange() PASSED
UsermenuTest > TestSummary() PASSED
UsermenuTest > InitMenuDisplayNormalTest() PASSED
UsermenuTest > TestSummaryWrongperiod() PASSED
UsermenuTest > TestConvertEmptytime() PASSED
UsermenuTest > TestSummaryNoexchanegs() PASSED
CurrencyImplTest > equals() PASSED
CurrencyImplTest > testToString() PASSED
CurrencyImplTest > getName() PASSED
CurrencyImplTest > hashCode() PASSED

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.1/userguide/command\_line\_interface.html#sec:command\_line\_warnings

BUILD SUCCESSFUL in 10s
3 actionable tasks: 3 executed
```

Build.gradle file

Plugins: plugins contain the tools we use when we create the project. In this way, we use java as the language of this project. Besides, jacoco is created to know the coverage of test cases for our project.

The application is used to run the project corresponding to the “mainClassName = ‘Model.App’” which sets the main class.

Repositories: the “mavenCentral()” ensures that “Gradle uses the Maven central repository when it resolves the dependencies of our Gradle build(Petri, 2017).

Dependencies: staff in dependencies gives an API for writing tests and extensions and an engine for running tests.

Test : configurations in test gives the possible result of “passed”, “skipped”, and “failed” separately.

Run: The “standardInput” in run allows the user to enter things through the terminal when run the project through Gradle, which corresponds to the Scanner in the project.

Junit

Table Test:

Setup() set up all objects will be used in the test including 'Currencies', 'Exchanges', and 'Times'. It also tests the creation of the table.

CompareTest() tests the value of rates in two exchanges and returns a correct format. If the first rate is larger than the second, it returns "first rate ↑". Otherwise, it returns "first rate ↓". Besides, it returns "first rate" if they are equal.

getDateTest() tests if the creation of the date achieves the format of what we want.

displayNormalTest() tests if the table can be displayed successfully.

getYesterdayTimeTest() tests if it can get the yesterday dates with the correct format as what we want.

displayWithYesterdayDateAdded() tests if the table can be displayed successfully when a yesterday time object is added.

UserMenu:

TestGetCurrentTable() tests the creation of the table with a current time shown in the system. Besides, check the date of the table created is the current date.

TestConvert() regular input of convert() method, it should calculate the converted value according to the exchange rate between the two currencies.

TestConvertEmptytime() if the input parameter "time" is null it should return the null, this test case will test if the return value is null or not.

TestSummary() regular input of summaryconversion() method. It should give the history of conversion rate between the 2 selected currencies and the average, median, max, min, standard deviation of these conversion rates, this is a void method, so I used ByteArrayOutputStream to test the standard output that is printed in the method.

Edge test cases of summaryconversion():

TestSummaryEmptytime() It tests a case when the input parameter time is null, the method should print an error message about this situation and this test case will test this output.

TestSummaryWrongperiod() It tests the situation when the entered start date is later than the end date which doesn't make sense, this method should output an error message and this test case will test it.

TestSummaryNoexchanegs() It tests the situation when there are no matching exchange objects found, this test case will test if the error message is printed or not.

getEditedCurrenciesTest() tests if it can get the correct edited currency list which are the last four currencies in the whole currency list.

CurrencyImplTest

getName() tests if we can get the correct name when we create a new currency compared with getting from “getName()” method.

Equals() tests if the two currencies can be compared through the equal method override in Currency class.

HashCode() tests if the address two currencies can be compared through the “hashCode()” method

testToString() tests if the currency object can use the “toString()” method and get the currency name of the currency object.

ExchangeImplTest

getFromCurrency() tests if we can get the correct “from currency” object when we create a new exchange compared with getting from “getFromCurrency()” method.

getToCurrency() tests if we can get the correct “to currency” object when we create a new exchange compared with getting from “getToCurrency()” method.

getRate() tests if we can get the correct rate when we create a new exchange compared with getting from the “getRate()” method.

Equals() tests if the two exchanges can be compared through the equal method override in Exchange class.

HashCode() tests if the address two exchanges can be compared through the “hashCode()” method

testToString() tests if the exchange object can use the “toString()” method and get the “from currency name”, “to currency name”, and rate of the exchange object.

TimeImplTest

setup() It creates all variables that are needed for testing and in this test file, it creates a time object and its exchange list inside.

getExchanges() tests if we can get the correct “exchanges” list when we create a Time object compared with getting “getExchanges()” method from the new Time object.

getExchange() tests if we can get the correct Exchange object in the “exchanges” list of a Time object compared with getting “getExchange()” method from the new Time object.

addExchange() test if we can add an Exchange object into the “exchanges” list of a Time object, this will use the getExchange() method to check whether the new Exchange object is added.

getDate() test. It is used to check whether getDate() will return the right date output.

deleteExchange(). It is used to check whether the exchange was deleted.

Admin_ExchangeTest:

`add_exchange()` will divide String into 3 parts by “,”. The First part is `currencyFrom` and Second part is `currencyTo`. Both of them are String. The third part is String but it can be cast into double. It is the new rate of this exchange.

“database” is the list with all the currencies which are added before.

`simple_error_add()`: The third part we need to cast it to double. This method is used to test whether the third part can be cast to double.

`simpleadd()`: This method is used to check whether the first currency and second currency is in the database. Because once we add an exchange, exchanges will add 2 exchanges. One from `currencyForm` to `currencyTo` with user rate and we will use Math method to automatically add reverse exchange. For example, Admin add (AUD,CNY,5). We will also automatically add `exchange(CNY,AUD,0.2)` ;

`simpleadd_withreplace()` .We update an exchange from (AUD,CNY,5) to (AUD,CNY,6). `simpleadd_withreplace` is used to check whether it updates.

`simpleadd_withsamexchange()`. When we update an exchange from (AUD,CNY,5) to (AUD,CNY,5). We will not add it to the database again.

`simpleadd_withreplace()`: We renew an exchange with differ rate, this method is used to check whether the new rate is double.

`same_currency_name_inexchange()`: Check whether the `currencyFrom` and `currencyTo` are the same.

`miss_both_currency()`: Check whether Both currency(from and to) are in the database.

`miss_to_currency()`: Check whether the currency(from) is in the database.

`miss_from_currency()`: Check whether the currency(to) is in the database.

`input_inlegal()`: check whether the String can be divided into 3 parts. For example, `Add_exchange(“AUD”, “CNY”, 5, 98273)` . It will be divided into 4 part.

AddCurrencyTest:

`addCurrency_normalCase()`: The method is used to test the normal case which adds a currency to the list and checks if the currency is successfully input.

`addCurrency_lotsOfInputsCase()`: The method is used to test a case which adds a lot of currencies to the list and checks if they are successfully input.

`addCurrency_usedCurrencyCase()`: The method is used to test a error case which add a same currency as the list and check if the function ignores currency.

BuilderTest:

`buildCurrency()`: tests if the builder can build a currency object correctly according to its name.

`buildExchange()`: tests if the builder can build an exchange object correctly according to its “From Name”, “To Name”, and rate.

Jenkins

0. The highest priority of Agile Principles is to satisfy the customer through early continuous delivery of software. In the agile process, welcoming changing requirements and delivering working software frequently are two major requirements we need to follow.

1. Why do we need Continuous Integration? In order to satisfy the welcoming changing requirement and delivering working software frequently. We use continuous Integration development practise to help us to minimise the duration and effort by each integration and we can deliver product versions suitable for release at any time.

2.why we need to use jenkins?Continuous Integration is a development practise.If we want to achieve the goals of Continuous Integration, we need some tools to implement it.There are version control tools , a tool to finish automated build processes , a tool to do configuration of the system build and testing processes.

The pipeline of Continuous Integration is Commit stage(compile,Unit test,Analysis,Build installers) ->Automated acceptance testing, Manual testing. And the Continuous Delivery is an extension of Continuous Integration.It make sure that you can release your new change to customer immediately

Jenkins is an automation server to automate tasks related to building,testing, and delivering or deploying software. It is a tool to integrate all the tool functions which are mentioned above to demonstrate the output to the customer.The jenkins pipeline supports implementing and integrating CD pipelines into Jenkins.

3.How Jenkins integrated with GitHub

Firstly ,with the help of ngrok ,we need to expose localhost to the internet so that every user who knows the account and password can see the jenkins website.

Secondly we need to install the Git plugin in jenkins. The GitHub plugin improves the integration ability with GitHub.And it allows us to set up a Hook. Webhook allow external services to be notified when certain events happen. Then ,we can set a webhook for one repository in GitHub. When the specified events happen in this repository (like git push origin master), GitHub will send a POST request to the Jenkins Server.

In Jenkins project setting up, we select GitHub hook trigger for GITScm polling. This function will trigger build in Jenkins when the specified events happen in GitHub, for example, git push origin master.

When Jenkins is notified , it will automate building,testing and delivering or deploying software from git repository . So, the teammate or customer can get the result easily.

4.Build and testing coverage on Jenkins

With the help of webhook and the select GitHub hook trigger for GITScm polling, The jenkins will build gradle when there is a push in git repository.The gradle build command will test all the testcases in the program with the help of Junit. The Junit already bundled with

Jenkins. If this is something wrong with the testcase, the pipeline will fail and marked as UNSTABLE(Red colour).

If we want to see the testing coverage in jenkins, We need to install JaCoCo Plugin in Jenkins. After the configuring, we can see the testing coverage of testing. The Jacoco will give us a detail about Overall Coverage Summary.

5. how to setup

This assignment is a group assignment, to allow all members to check the automated build and test result for each push, the ngrok is needed to make jenkins available on the internet. By using command “ngrok http 8080”, then a url is generated which will redirect to the local jenkins server.

```
ngrok by @inconshreveable (Ctrl+C to quit)

Session Status      online
Account             j1279440833@gmail.com (Plan: Free)
Version             2.3.35
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://217b5786a085.ngrok.io -> http://localhost:8080
Forwarding           https://217b5786a085.ngrok.io -> http://localhost:8080

Connections          ttl      opn      rt1      rt5      p50      p90
                    484      0        0.01     0.04     0.91     7.60

HTTP Requests
-----
```

Then all members can use this url to open jenkins to check results. In addition, the member who set up the jenkins register a ngrok account so that the url will not expire in 8 hours.

As mentioned above, the jenkins need to be hooked with github. In the configuration of jenkins, we need to enter the repository url and create a credential to access its content and set the branch specifier to master branch because only changes on master branch need to be built and tested.

Source Code Management

☐ None
☒ Git

Repositories

Repository URL:

Credentials:

Branches to build

Branch Specifier (blank for 'any'):

Repository browser:

By selecting the Github hook trigger for GITScm polling on the Build Trigger section, every change on master branch(as it is set as branch specifier) will trigger a build.

☒ **GitHub Integration Plugin** 0.2.8

GitHub Integration Plugin for Jenkins

With github integration plugin installed, jenkins can receive triggers from Github. In the setting of Github, we add “/github-webhook” at the end of the url and enter it as payload url and the green tick will be shown on the left of the webhook if it is configured

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

We will also send events from this repository to your [organization webhooks](#).

✓ https://217b5786a085.ngrok.io/github-webhook/ (push)	<input type="button" value="Edit"/>	<input type="button" value="Delete"/>
--	-------------------------------------	---------------------------------------

correctly.
The gradle of version 6.6 is installed for automated build and test.

Gradle installations

Add Gradle

Gradle

name

☒ Install automatically

Install from Gradle.org

Version

Delete Installer

In the build section of configuration, we set the gradle version to the one just created and enter the task: clean and build. The task clean will clear everything from the previous build then run gradle build for this time. Every time the build is triggered by changes happening on the master branch, these tasks will be executed.

Publish JUnit test result report

X?

Test report XMLs

/*test-results//**.xml

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is [the workspace root](#).

☐ Retain long standard output/error?

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?

The Junit plugin can produce a test report which shows how many test cases are passed or failed.

	JUnit	
<input checked="" type="checkbox"/>	Allows JUnit-format test results to be published.	1.37

Publish JUnit test result report

Test report XMLs

Fileset 'includes' setting that specifies the generated raw XML report files, such as 'myproject/target/test-reports/*.xml'. Basedir of the fileset is **the workspace root**.

☐

Retain long standard output/error

Health report amplification factor

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

The JaCoCo plugin is used to generate jacoco test reports on jenkins, the coverage report is generated for every build.

JaCoCo plugin

This plugin integrates [JaCoCo code coverage reports](#) to Jenkins.

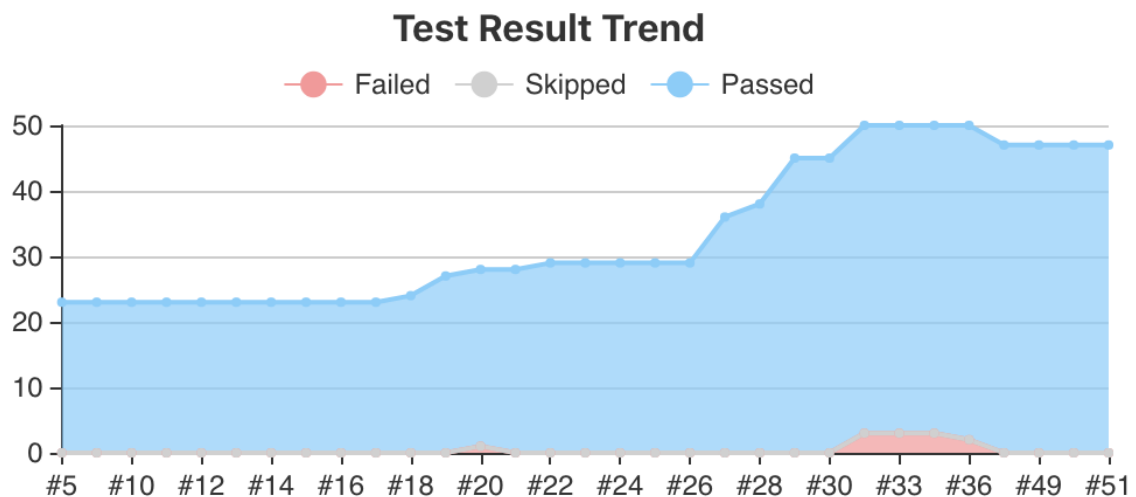
3.0.8

Post-build Actions

Path to exec files (e.g.: <code>**/target/**/*.exec,</code> <code>**/jacoco.exec)</code> <input style="width: 100%;" type="text" value="**/**/*.exec"/>	Inclusions (e.g.: <code>**/*.class)</code> <input style="width: 100%;" type="text"/>	Exclusions (e.g.: <code>**/*Test*.class)</code> <input style="width: 100%;" type="text"/>
Path to class directories (e.g.: <code>**/target/classDir, **/classes)</code> <input style="width: 100%;" type="text" value="**/classes"/>		
Path to source directories (e.g.: <code>**/mySourceFiles)</code> <input style="width: 100%;" type="text" value="**/src/main/java"/>	Inclusions (e.g.: <code>**/*.java,**/*.groovy,**/*.gs)</code> <input style="width: 100%;" type="text" value="**/*.java,**/*.groovy,**/*.kt,**"/>	Exclusions (e.g.: <code>generated/**/*.java)</code> <input style="width: 100%;" type="text"/>

6. Explanation of representative Jenkins outputs

-JUnit test report



In the main page, we can see the trend of junit test results, how the test cases are increased, when test cases failed, the answers of these questions are clearly shown in the picture.

Test Result

3 failures (+3)

50 tests (+5)

Took 7 sec.

 add description

All Failed Tests

Test Name	Duration	Age
+ Model.AddCurrencyTest.addCurrency_multiSituationCase()	15 ms	1
+ Model.AddCurrencyTest.addCurrency_usedCurrencyCase()	3 ms	1
+ Model.AddCurrencyTest.addCurrency_nonAlphabetCase()	1 ms	1

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
Model	6.9 sec	3	+3	0		47	+2	50	+5

This screenshot above is a junit test report of one fail build in the jenkins, we can see which test case failed and how many test cases failed.

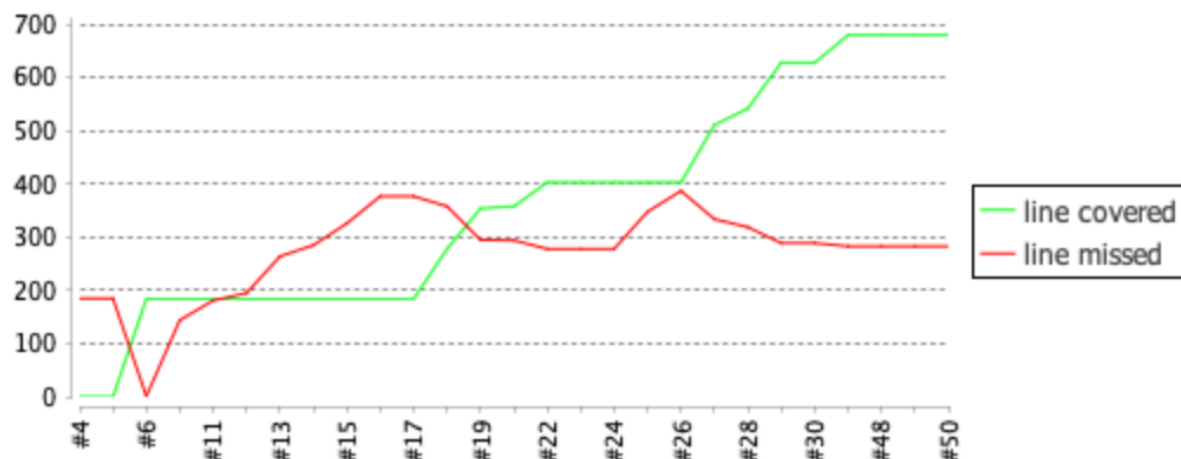
All Tests

Class	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
AddCurrencyTest	23 ms	3	+3	0		2	+2	5	+5
Admin_ExchangeTest	0.22 sec	0		0		9		9	
BuilderTest	4 ms	0		0		3		3	
CurrencyImplTest	7 ms	0		0		4		4	
ExchangeImplTest	0.13 sec	0		0		6		6	
MonthTest	3 ms	0		0		3		3	
TableTest	6.2 sec	0		0		5		5	
TimelImplTest	73 ms	0		0		7		7	
UsermenuTest	0.15 sec	0		0		8		8	

After clicking on the Model Package, we can see the passing condition of all test classes.

-JaCoCo coverage report

Code Coverage Trend



The coverage trend is shown in the main page as well, as the test cases are pushed to the master branch we can see the green curve, the line covered increased quickly.

Jacoco - Overall Coverage Summary

INSTRUCTION	85%	<div><div></div><div></div></div>
BRANCH	67%	<div><div></div><div></div></div>
COMPLEXITY	71%	<div><div></div><div></div></div>
LINE	81%	<div><div></div><div></div></div>
METHOD	90%	<div><div></div><div></div></div>
CLASS	100%	<div><div></div><div></div></div>

name	instruction	branch	complexity	line	
AddCurrencyTest	M: 11 C: 203 95% <div><div></div><div></div></div>	M: 0 C: 0 100% <div><div></div><div></div></div>	M: 1 C: 5 83% <div><div></div><div></div></div>	M: 4 C: 41 91% <div><div></div><div></div></div>	M: 83
AdminMenu	M: 265 C: 828 76% <div><div></div><div></div></div>	M: 33 C: 84 72% <div><div></div><div></div></div>	M: 29 C: 45 61% <div><div></div><div></div></div>	M: 60 C: 156 72% <div><div></div><div></div></div>	M: 92
Admin_ExchangeTest	M: 0 C: 335 100% <div><div></div><div></div></div>	M: 0 C: 0 100% <div><div></div><div></div></div>	M: 0 C: 11 100% <div><div></div><div></div></div>	M: 0 C: 64 100% <div><div></div><div></div></div>	M: 101
AdminMenuTest	M: 16 C: 356 100% <div><div></div><div></div></div>	M: 0 C: 0 100% <div><div></div><div></div></div>	M: 0 C: 11 100% <div><div></div><div></div></div>	M: 8 C: 79 100% <div><div></div><div></div></div>	M: 101

After clicking on one of the build histories, there is an overall coverage summary, including the coverage of instruction, branch, complexity, line, method and class.

```

Collections.sort(rates); // sort by the rate
if (rates.size() == 1) { // if there is only one exchange rate, this r
    System.out.println("median rate: " + String.format("%.2f", rates.g
} else if (rates.size() % 2 != 0) { // if the number of exchange rate
    System.out.println("median rate: " + String.format("%.2f", rates.g
} else { // if the number of exchange rate is even, the average of the
    System.out.println("median rate: " + String.format("%.2f", (rates.
}

```

You can see more detailed coverage on each class or method by clicking on them in the coverage report, the code covered by green means fully covered, red means missed and yellow means partly covered.

Quality of Application Development

Group collaboration:

For the tools setting, agile tools are set up including GitHub, Gradle, Junit in members' computers with the same version in case of conflicts of version. Jenkins is established on one group member's computer which has linked to the group's repository in GitHub and Gradle and he would send the link, username and password to other group members every day. The specific functionality of the above tools has already been introduced and explained in the above section.

For the group communication, WeChat is an effective tool for group members daily communication during the project, it can be used to prepare for a group meeting when a problem or situation comes up, daily progress report of every members' process and reply to some simple questions or situations. To complete the setting up, Zoom is used to discuss and solve the problems by screen sharing and voice call functionality. It is also the platform for group members to hold a regular group meeting about task allocation, progress or situation description, idea discussion, code explanation and pilot test. GitHub is a powerful and useful tool during the coding part, group members can directly and quickly share and discuss their update on code to others and GitHub can clearly show the difference between different versions so that group members can easily find the conflict and discuss it.

For completing the application, our group firstly lists the functionality together and determines the class and corresponding method the product should have. After determining the frame, one member would complete the whole basic class and methods to ensure the consistency of code with comments beside and upload them to GitHub. Then, a meeting is held on Zoom for code explanation and the remaining members of the group are divided into two groups(two people for each group), Admin Menu group and User Menu group, to complete the functionalities. Since there are differences in coding style and habit, it is better to ensure the independence of everyone's work. During the coding, group members use github to turn in their works and resolve the conflicts and Gradle runs for correctness confirmation for the functionality. Also, the Jenkins is used to monitor the whole process of the changes. Test cases would be completed after each functionality is completed to ensure the Junit testing can be above 75% to ensure the correctness of the functionalities. After all these are done, we hold a pilot test for our final code. Each member acts as a user and tests different inputs including normal cases, special cases and error cases to ensure the application can give the right answer that we expected and meets all the project requirements. Since Jenkins can monitor the whole process once codes are pushed into GitHub, all group members unify and integrate the final version of codes and check it in Jenkins to complete the application.

During the completion of the application, every group member contributes in his own tasks during coding and completes them before the deadline of milestone. All group members attend regular meetings for discussion and group issues including tool setting, task allocation, progress description, idea discussion, code explanation and pilot test.

Here is the detailed contribution for every group member in this project:

All group members: Tools setting and using - GitHub, Gradle, Junit

Determine the class and functionalities(requirements gathering)

Pilot test

Report – Junit, part of readme

Xiaohan Li: Application Code - AdminMenu(AddCurrencies), App

Test Case – AddCurrencyTest

Report - Quality of Application Development

Sub-group - Admin Menu group

Yuben Fang: Application Code – NormalUtil, Table,

Test Case – TableTest

Report – part of Gradle, Github

Sub-group - User Menu group

Chenglong Li: Tool management - Jenkins

Application Code – UserMenu

Test Case – UserMenuTest

Report – part of Jenkins

Sub-group - User Menu group

Canwei Cai: Application Code - AdminMenu(AddExchange, DisplayAdminMenu)

Test Case – AddExchange, AdminMenuTest

Report – part of Jenkins

Sub-group - Admin Menu group

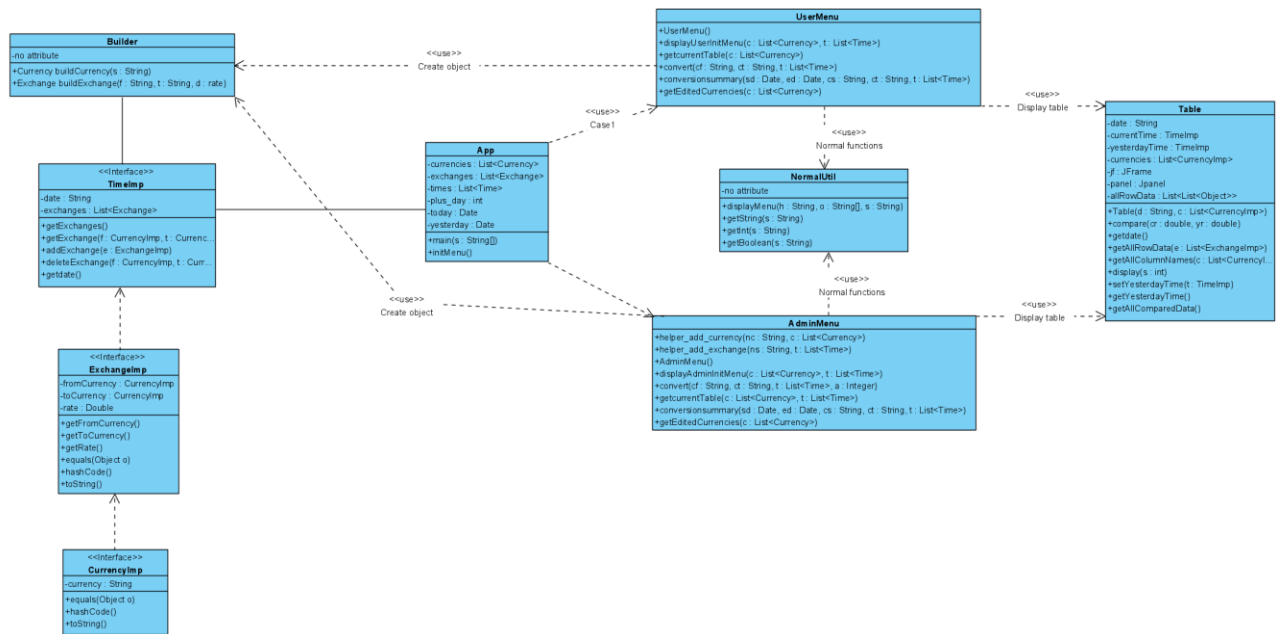
Jiesheng Liang: Application Code – Builder, Time, Exchange, Currency

Test Case – BuilderTest, TimeTest, ExchangeTest, CurrencyTest

Report – part of Gradle

Sub-group – Basic class and method completion

Following is the class diagram of the coding part:



In this application, class *Builder* is used to build objects *Time*, *Exchange* and *Currency*. Class *Time* means the exchanges and currency in a certain date which is used as the main object which every *time* contains a *date* and a list of *exchanges* and the table is displayed depending on the object. Class *Exchange* means the rate that a currency can exchange to another one which every *Exchange* contains two *currencies* and a *rate*. Class *currency* means the type of currency which is a basic class with a string name.

Class *APP* contains the *main* function which is used to run the program. Once the program runs, the *initmenu* function would initialize a table with six currencies and ask the user to check their identity. If a user chooses '1', then the *UserMenu* function would be called to perform normal user's function. If a user chooses '2', then the *AdminMenu* function would be called to perform the admin's function. If a user chooses '3', then the system would go to next day and reinitialize the table with yesterday data remaining. Class *Table* is used to display a pop-up table to show the currency and class *NormalUtil* is used to provide some basic functions which would be used in *UserMenu* class and *AdminMenu* class.

Here is the guide of our application:

About how to run the program: First, pull the whole file from git master branch, then use gradle to build and run the program. After the program runs, it will show the dates at the top, then will let you make a selection in the following opinion(1.Normal user, 2.Admin, 3.Next Day, and 4. quit).

Normal user:

- 1.Display the currency table - can let the user check the exchange result and currency rate on the table.
- 2.Convert the currency - can let the user select the currency to convert, while the user enters the amount of the currency they want to convert, the result to display on the table.(example: 1.input currency you want to convert fom 2. input amount of the currency 3. input the currency you want to convert to).

3. Show summary of conversion - the user can print a summary of the conversion rates of 2 currencies the user chose within a specific duration (start and end dates). This includes all conversion rates, average, median, maximum, minimum and standard deviation of the conversion rate of the 2 currencies during the specified start and end date.

4. Back - back to period stage.

Admin:

1. Display the table - display the current convert currencies and display the popular currencies table.

2. Add a new currency - can add new currency to the database (enter example: < new currency name >).

3. Add a new exchange - can add new exchange rates daily by entering the date and exchange rate for that date of all currencies stored in the file (enter example: < Currency from Currency_name > , < Currency to Currency_name > , < double rate >).

4. Back - back to period stage.

Next day: change the next day.

Since the test cases depend on the requirements can cover around 80% of the coding which means at least the program can have correct output with normal inputs. During the pilot test, the application still behaved correctly to our different commands based on the functional requirements. As a result, we believe that the application can have a good performance during the demonstration.

Reference

STEN P.(2020). Continuous integration vs. continuous delivery vs. continuous deployment. Retrieved from <https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>

Jenkins with github(2020). Retrieved from <https://www.jenkins.io/solutions/github/>

Petri Kainulainen(2017). JUnit5 Tutorial:Running Unit Test With Gradle. Retrieved from <https://www.petrikainulainen.net/programming/testing/junit-5-tutorial-running-unit-tests-with-gradle/>