

COMP3111: Software Engineering

Team Project - Course Scraper

Project Problem Statement and Activities

Activity 1 (6%) - Due: Apr 6, 2020 at 11:59pm
Submission details will be provided on Canvas

Activity 2 (24%) - Due: May 8, 2020 at 11:59pm
Submission details will be provided at Canvas

Introduction

You are going to form a team of three to work on a project. The project is about web scraping - retrieving and analysing data from some websites automatically. You will be provided with a skeleton code written in Java that is capable of fetching some data from the HKUST course website.

Codebase at GitHub: <https://github.com/namkiu2020/comp3111-project-2020s>

System Requirements

There are 6 tasks for each team to complete. Each student should complete **TWO** of the tasks. All tasks are closely related, and team members are expected to work together collaboratively in delivering a fully functional system. In most cases, no point will be awarded if a related task has not been completed. Points are given in an all-or-nothing manner.

Glossary

The following table defines and clarifies some of the keywords used throughout this document.

Keywords	Meaning
Term	A term consists of four numbers. The first two represents the academic year. For example, 18 refers to the academic year 2018-19. 17 refers to the academic year 2017-18. The third and fourth numbers can be either 10, 20, 30, 40, representing Fall term, Winter term, Spring term and Summer term, respectively.
Subject	Refers to the four-letter code prefix of a course code, e.g. COMP, ACCT, LANG, SHSS, UROP.

Keywords	Meaning
Course	A course can be identified by a unique course code. COMP3111 and COMP3111H are two different courses although they are co-listed.
Section	There is at least one section in a course. A course can have multiple sections, identified by a unique section-ID (for example, 1808 refers to "COMP 1029J L1"). A section code (L1, L2, LA1, T2...) is a unique identifier within a course. Combining the course code and section code, there will be a unique section in that semester. A code starts with 'L' followed by any characters except 'A' is lecture (e.g. COMP1022P LX is a valid lecture section). A section code starts with 'T' is a tutorial section. A section code starts with 'LA' is a lab section. Any section code does not follow these three rules can be ignored (e.g. COMP6990 R1).
Slot	A section may have zero, one, two, or three slots. A slot contains both information about time and venue. The time of a slot shall be represented by one day the week (i.e. Mo/Tu/We/Th/Fr/Sa) and a consecutive time from 9:00AM to 10:00PM. Any slot with a time that does not follow this rule is considered invalid (e.g. ACCT6900E L1, COMP6931A). A time slot contains two days (e.g. TuTh 12:00PM - 01:20PM) should be considered as two slots.
Enrolment	For simplicity, most of the enrolment rules (pre-requisite, exclusion, matching lecture-lab like COMP1022P, enrolling in multiple lecture sections of the same course, enrolling only lecture but not tutorial, etc) are not checked and they are considered as VALID. An enrolment contains exactly one section which can be a lecture, a tutorial, or a lab section.
Console	It refers to the white space on the GUI. We refer the area that the output stream <code>System.out.println</code> writes to as the system console .
Base URL	It refers to the site that host the course enrolment information. By default, it is https://w5.ab.ust.hk/wcq/cgi-bin/ . However, during the grading exercise we will test it with another URL to avoid student hard coding information.

Tasks Description

Task 0: Main

A basic search function has been implemented with the skeleton code provided.

The team should discuss among themselves on the task assignment for the implementation of the add-on features (Task 1 to Task 6).

The screenshot shows a web application titled "Course Scraper" with a navigation bar containing links: Main, Backend, Filter, List, Time Table, All Subject Search, and SFQ. The main content area has three input fields: "Base URL" with the value "https://w5.ab.ust.hk/wcq/cgi-bin/", "Terms" with the value "1830", and "Subject" with the value "COMP". A blue "Search" button is positioned to the right of the "Subject" field. Below the input fields, a note states: "Please note that during grading we will save the Base URL into another site and manually edit some data. Don't hardcode". A large red word "Given" is centered on the page. At the bottom, there is a console window labeled "onsole" displaying the following text:

```
COMP 1001 - Exploring Multimedia and Internet Computing (3 units)
Slot 0:We14:00-15:50:Rm 5620, Lift 31-32 (70)
Slot 1:Tu09:00-10:50:Rm 4210, Lift 19 (67)

COMP 1021 - Introduction to Computer Science (3 units)
Slot 0:Mo14:00-14:50:Rm 2464, Lift 25-26 (122)
Slot 1:We15:30-16:20:Rm 2465, Lift 25-26 (122)
Slot 2:Fr15:30-16:20:Rm 2465, Lift 25-26 (122)
Slot 3:We16:30-17:20:Rm 2465, Lift 25-26 (122)
Slot 4:Fr16:30-17:20:Rm 2465, Lift 25-26 (122)
Slot 5:Mo15:00-16:50:Rm 4213, Lift 19 (67)
Slot 6:Tu10:00-11:50:Rm 4213, Lift 19 (67)
Slot 7:Mo09:30-11:20:Rm 4213, Lift 19 (67)
Slot 8:Tu15:30-17:20:Rm 4213, Lift 19 (67)

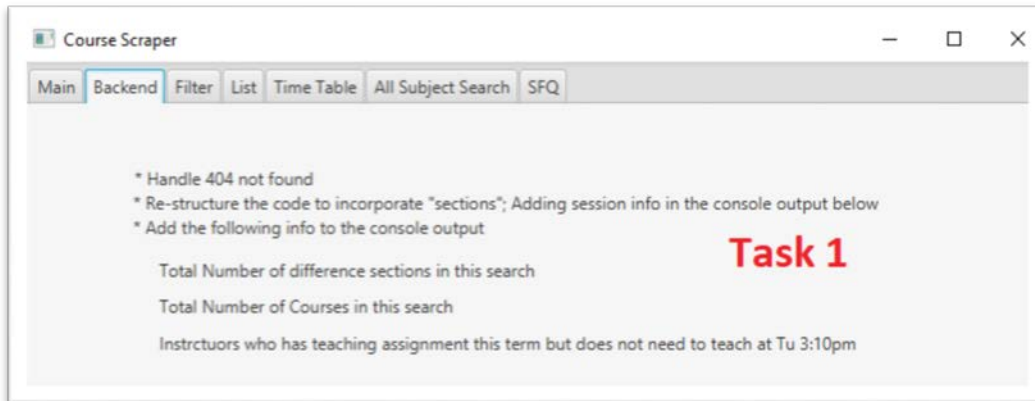
COMP 1022P - Introduction to Computing with Java (3 units)
```

Task 1: Backend [8%=20pt]

1. Properly handle the 404 page not found - display an appropriate message on the screen to notify the users and not throwing error on the system console. [4]
2. Introduce the class section that model the concept of sections. [2]
 - i. Modify the skeleton code so that when the search button is clicked, the section information is also displayed on each row. [4]
3. When the search button is clicked,
 - i. Display Total Number of difference sections in this search: NUMBER_OF_SECTIONS in console. NUMBER_OF_SECTIONS shall include all sections even if those sections contains only invalid slot. [3]
 - ii. Display Total Number of Course in this search: NUMBER_OF_COURSES in console. NUMBER_OF_COURSES shall include all courses that has at least

one lecture section, or a lab section, or a tutorial section. (Thus, COMP7990 is excluded). [3]

- iii. Display Instructors who has teaching assignment this term but does not need to teach at Tu 3:10pm: INSTRUCTOR_NAME1, INSTRUCTOR_NAME2, INSTRUCTOR_NAME3, ... where the list INSTRUCTOR_NAME are the subset of the union of the instructors' name obtained in the current search. It should contain all and only those has no teaching assignment at Tu 3:10pm. [3]
 - a. Sort the order of the instructor's name ascendingly according to the alphabetical order of their display name. Display name refers to LAST_NAME, First_name.[1]



Task 2: Filter [8%=20pt]

1. When Select A11 is clicked.
 - i. All boxes on this tab are checked. [2]
 - ii. The text of that button will be changed to De-select A11. [1]
 - a. When De-select A11 is clicked.
 - a. The text of that button will be change back to Select A11. [2]
 - b. All boxes on this tab are unchecked. [1]
2. When the status of any box on this tab is changed (checked or unchecked), the console will be cleared and filtered information is displayed. [2]
 - i. If the boxes AM (or PM) is checked, display only all sections of the courses which has a slot in AM (or in PM). [2]
 - a. If both AM and PM are checked, display only all sections of the courses that has a slot that starts at AM and ends at PM, or a section has a slot in AM and another slot in PM. [2]
 - ii. If any boxes of days of the week (Monday, Tuesday, ...) is clicked, display only all sections of the courses that has slots on the selected boxes. Thus, if only Monday and Tuesday are clicks, only sections that has slot on both Monday and Tuesday will be displayed. [2]

- iii. If Common Core is clicked, display only all sections of the courses that are 4Y CC. [2]
 - iv. If No Exclusion is clicked, display only all sections of the courses that does not define exclusion. [2]
 - v. If With Labs or Tutorial is clicked, display only all sections of the courses that has labs or tutorials. [2]
3. *Note: filters are combined with AND logic. Therefore, if multiple filters are applied, display all sections of the courses which simultaneously fulfils the requirements.*

The screenshot shows a web application window titled "Course Scraper". It has a navigation bar with tabs: Main, Backend, Filter (selected), List, Time Table, All Subject Search, and SFQ. Below the tabs, there is a "Select All" button and several filter options with checkboxes:

- ☐ AM
- ☐ PM
- ☐ Monday
- ☐ Tuesday
- ☐ Wednesday
- ☐ Thursday
- ☐ Friday
- ☐ Saturday
- ☐ Common Core
- ☐ No Exclusion
- ☐ With Labs or Tutorial

There are three text blocks providing instructions:

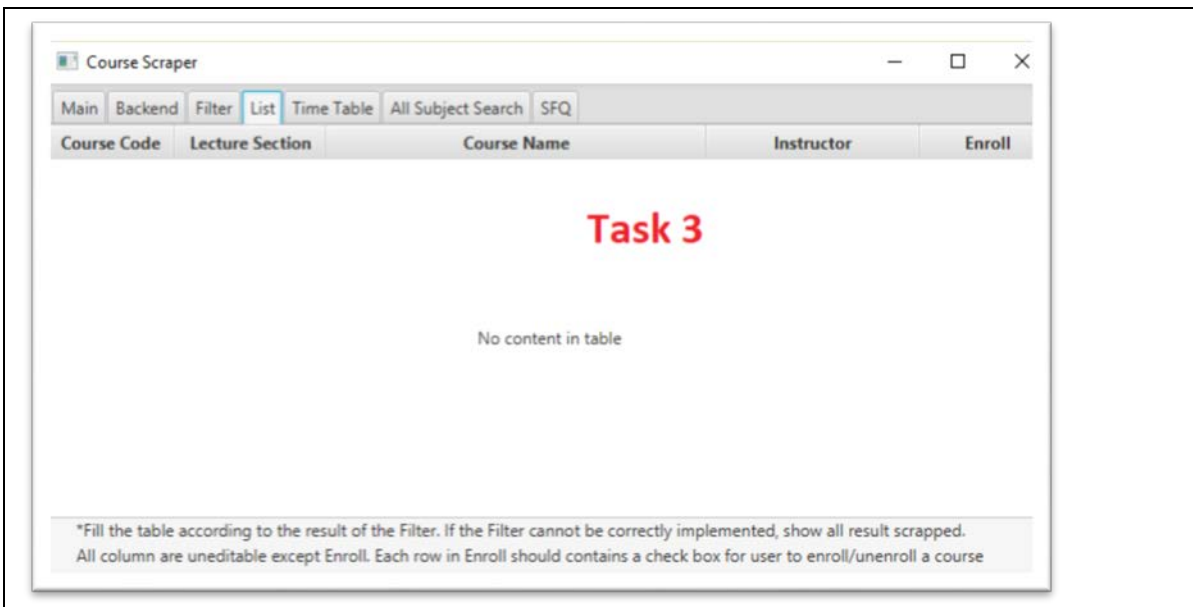
- A course has a section at that time/on that day. A section can be lecture/tutorial/lab. Click AM + Monday does not imply a course has a section on Monday morning. It means there is a section in the morning, but that section may not be on Monday. However, that course must have another Monday section.
- Common Core refers to 4Y CC; No Exclusion refers to a course does not has any exclusion. Information can be founded on the pop-up text of the "COURSE INFO"
- The section code of a Lab must begins with LA, tutorial must begins with T. Code begins with R can be ignored.

A red text "Task 2" is overlaid on the right side of the window.

* When the filter is applied, display all session of courses that fulfill all of these conditions; Print them to the console below on any change of checkbox

Task 3: List[8%=20pt]

1. Fill the table correctly with the result of the Filter. If the Filter cannot be correctly implemented, show all result scrapped. [9]
 - i. Make all cell non-editable except for the column Enroll. [2]
 - ii. Each row in the column Enroll should contains a checkbox. [2]
 - a. When the status of the checkbox on this tab is changed, the console will be cleared and filtered information is displayed. [1]
 - a. In addition, it displays The following sections are enrolled: on the console, followed the list of the enrolled sections (order is not important). [2]
 - a. The enrolment status will be persistent even when another search is performed, another filter is applied, or selecting other tabs. [2]
 - b. The enrolment status will be erased only when the checkbox is unchecked. [2]



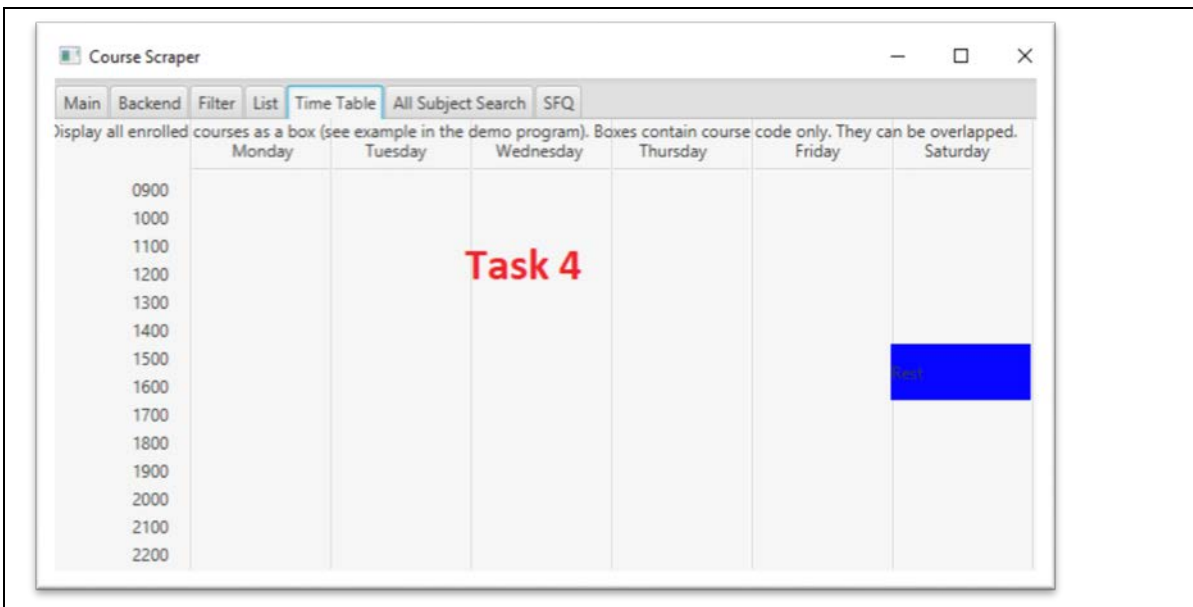
Task 4: Timetable[8%=20pt]

1. Update the table when enrollment status is changed. [2]
2. Create a block on the timetable tab for each slot of the enrolled section. [10]
 - a. Each block should contains both course code and section code in two lines. [2]
 - b. Same background color should be applied to the block of the same section while different color should be applied to block with different sections. [2]
 - c. If there are time clash happens, the boxes of the time clashed slot will be overlapped. [2]
 - d. The overlapped area will be displayed in different color. [2]

as they can be overlapped

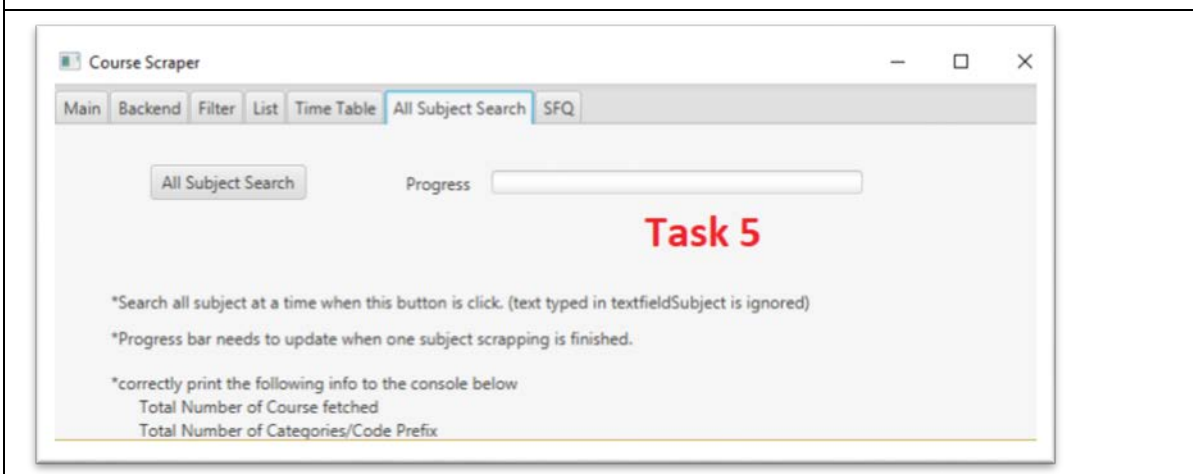
Friday	Saturday
	COMP1022 L1
	COMP1111 L2

Note: If Enrollment cannot be correctly implemented in List, enrol the first 5 sections of the scrapped data. Enrol to all sections if there are less than 5 sections from the scrapped data. In this case the table shall be changed when a search/all subject search is performed.



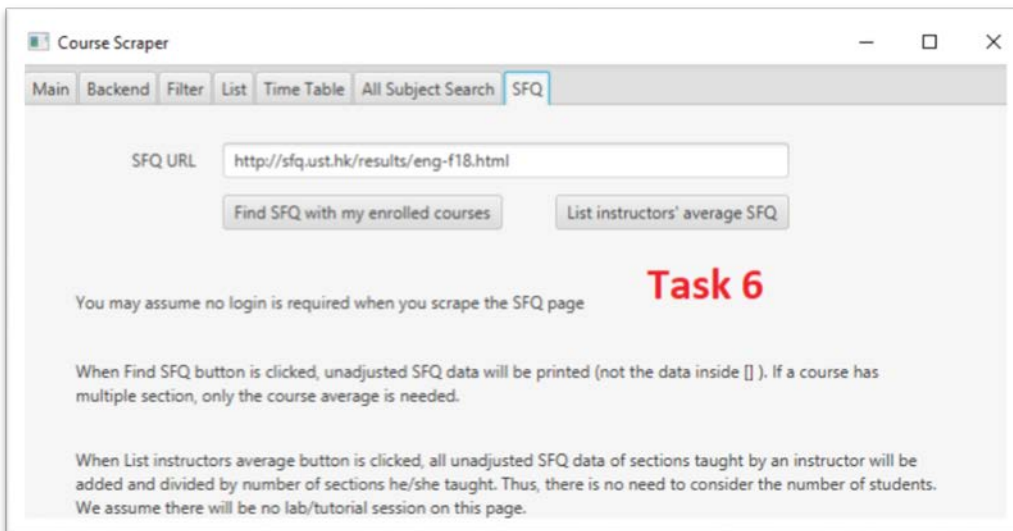
Task 5: All Subject Search [8%=20pt]

1. When Search Or All Subject Search is clicked, obtains the list of all subjects from the base_url + TERM.
 - i. Print Total Number of Categories/Code Prefix: ALL_SUBJECT_COUNT in console where ALL_SUBJECT_COUNT is the size of the list. [5]
 - ii. Search all subjects when the button All Subject Search is clicked again. [4]
 - a. After one subject is scraped. Print the SUBJECT is done on the system console (System.out.println). [3].
 - a. Update the progress bar by the fraction 1 / ALL_SUBJECT_COUNT. [4]
 - b. Print Total Number of Courses fetched:
TOTAL_NUMBER_OF_COURSES when all subjects scraped. The TOTAL_NUMBER_OF_COURSES includes all courses with course code (i.e., SCIE2500 is included). [4]
2. *Note: after all subject search is clicked, tab Filter, List, Backend shall show different result.*



Task 6: SFQ [8%=20pt]

1. Make Find SFQ with my enrolled courses disabled before search or All Subject Search is clicked. [1]
 - i. Make Find SFQ with my enrolled courses enabled after search or All Subject Search is clicked. [1]
2. When Find SFQ with my enrolled courses is clicked, scrape data from SFQ URL.
 - i. Print unadjusted SFQ data (not the data inside brackets) of the enrolled courses on console. [5]
 - a. If multiple sections are available for a course, take the average unadjusted SFQ data and print it. (simple average is needed, no need to consider the number of students). [4]
3. When List instructors' average SFQ, print all instructors' name and their unadjusted SFQ score on console. [5]. 1. If an instructor has taught more than one sections/courses, all unadjusted SFQ score of the sections taught by him/her will be added and divided by number of sections. [4]
4. *Note: If Enrolment cannot be correctly implemented in List, enroll the first 5 sections of the scrapped data. Enrol to all sections if there are less than 5 sections from the scrapped data. In this case the table shall be changed when a search/all subject search is performed.*
5. *Note: You can assume that no login is required when you scrape the SFQ page. You might want to download the page and stored in your harddisk for testing. During our grading we will host it at somewhere does not require login.*



The screenshot shows a web application window titled "Course Scraper". It has a navigation bar with tabs: Main, Backend, Filter, List, Time Table, All Subject Search, and SFQ (which is currently selected). Below the navigation bar, there is a text input field for "SFQ URL" containing the value "http://sfq.ust.hk/results/eng-f18.html". There are two buttons: "Find SFQ with my enrolled courses" and "List instructors' average SFQ". Below the buttons, the text "Task 6" is displayed in large red font. Underneath, there are three lines of instructional text: "You may assume no login is required when you scrape the SFQ page", "When Find SFQ button is clicked, unadjusted SFQ data will be printed (not the data inside []). If a course has multiple section, only the course average is needed.", and "When List instructors average button is clicked, all unadjusted SFQ data of sections taught by an instructor will be added and divided by number of sections he/she taught. Thus, there is no need to consider the number of students. We assume there will be no lab/tutorial session on this page."

Technical Requirements

1. The program must be written in Java.
 2. The project must use Gradle to manage.
 3. The program must use JavaFX as its only GUI framework. No Swing or AWT should be allowed.
 4. You may choose your own IDE. We recommend Eclipse IDE.
 5. JUnit 4.12 as your testing suite
 6. Jacoco as your test coverage measurement
 7. A Private GitHub repository for source control
 - i. Students can apply for [GitHub Education](#) to enjoy the benefits of creating private repositories with unlimited collaborators at GitHub
 - ii. In your private repository of team repo at GitHub, add all team members and the TA account "**MyProgrammingLab**" as collaborators.
-

Team Requirements

1. Each team **MUST** be formed by three students enrolled in COMP3111. Cross-section teams are allowed (i.e. L1 students can form a team with L2 students). Students not signing up for teams will be assigned to a random team of three (or, in some rare cases, a team of four) at the discretion of the teaching team.
2. Each team member is expected to accomplish two tasks (implement two functional features) from the system requirements.
 - i. No two students in the same group can implement the same feature. Students are expected to negotiate task allocation among themselves.
 - ii. The member implementing the task is also required to manage the Git commit log messages, testing, debugging, and documentation of that particular task.
 - iii. Two developers may need to work on the same Java method concurrently. Team discussion meetings and version control using Git are essential to minimize conflicts during the software development process.

Activities to submit

Activity 1 - System Requirement Specification [6%]

Based on the above project requirements, prepare and submit

1. Team work
 - i. A data model diagram (1%)
 - ii. A use-case diagram (1%)
 - iii. The workload distribution among your team (1%)
 - iv. A GitHub link properly setup (visible to TA) (1%)
2. Individual work
 - i. A use-case specification of each of your tasks (2%)

Activity 2 - Software Implementation and Testing [24%]

1. Team work - Submit via canvas [2%]
 - i. At least 3 meeting minutes by the end of the project (1%)
 - ii. A Gantt chart and a burn down chart (1%)
2. Team work, but mark individually - Submit via GitHub [22%]
 - i. A README file stating the name of team members and their tasks assigned
 - ii. A running program and the source code (8% for each task; Max 16%)
 - iii. Unit testing of your implemented tasks. (100% pass: 2%; otherwise: 0%)
Even if you are failed to implement all tasks, the unit testing is still required.
 - iv. Coverage test (>65% branch coverage: 2%; 45-65% branch coverage: 1%; otherwise: 0%)
Even if you are failed to implement all tasks, the coverage test is still required.
 - v. Git commit log at GitHub: At least three "non-trivial" commits and one "non-trivial" pull request (per student) (1%)
 - vi. Documenting your implemented features using JavaDoc (Full mark: 1%; -0.25% for each missing argument, return value, exception, class description): If a function is responsible by more than one student, all of them will receive the penalty.

We expect most of the students will finish the project with reasonable efforts. They will most likely receive full marks.

- No bonus.
- No late submit.
- Testing on JavaFX may require additional framework (TestFX) which will not be covered in the course. It is optional for you to use it or not.

Grading process

The TA will perform the following in a lab machine.

```
git clone <YOUR_URL> .  
git log # to make sure the last update is before the project due.  
gradlew build  
gradlew jacocoTestReport  
gradlew javadoc  
gradlew run
```

You will be invited to appeal at a later time if your code does not work.

Course Scraper

Compile

We configure the project with Gradle. Gradle can be considered as Makefile like tools that streamline the compilation for you. In this project you are not required to learn how to write Gradle file. If you are interested to know how, please check out [this link](#).

Compile with Windows Command Prompt

Navigate to the project folder. Type "gradlew run" will build and run the project.

If you want to just re-run the project without rebuilding it, you can go to the folder build\jar\ there should be a jar file (e.g. coursescraper-0.1.0.jar). Try double click on it (yes, you need a GUI screen to run it).

Compile with Mac/Linux terminal

Navigate to the project folder. Type "./gradlew build" to build and run the project.

If you want to just rerun the project without rebuilding it, you can go to the folder build/jar/ there should be a jar file (e.g. coursescraper-0.1.0.jar). Try double click on it (yes, you need a GUI screen to run it).

Compile with Eclipse

You are recommended to download and install [Eclipse IDE for Java Developers \(2019-12\)](#). This version of eclipse supports Gradle out of the box. Create a new project > Gradle Project and drag everything from the given base code to the project. Refer to Lab 1 to bring up the "Gradle Tasks" windows and double click on the "application > run" to launch the application.