Assignment 3: Desktop application development (2)

IMPORTANT: This assignment must be done individually

Read Section 1 to understand the programming requirements, Section 2 to understand the programming tasks that you need to carry out and Section 5 to know what you need to submit.

1. Description

As required from the stakeholder, you will change the CourseMan program to create a **CourseMan2** program with a graphical user interface using Java's **Swing API**. Similar to CourseMan, CourseMan2 still uses the `CourseManDemo` class to coordinate other components and to execute from the command line. This change is divided into **two phases**. In this assignment, you will continue to complete **the second phase** of CourseMan2 including:

(1) the function to **create a new enrolment**,

(2) the function to **display an initial enrolment report** & **an assessment enrolment report**

Similar to the two menus: *Student* and *Module*, *Enrolment* is for managing enrolment data objects. If *New Enrolment* menu item is chosen, then the Create new enrolment GUI will be displayed. This will allow the user to enter details about the enrolment objects. If *Intial Report* or *Assessment Report*, a tabular report dialog of enrolments without and with grades will be displayed respectively, the samples of interface design are shown in Section 4.

In addition, in this phase, we made our design decision to

(3) generalize our **data management controller classes** to increase code reusability.

Details about this generalization will be mentioned in details in Section 3.

The next section will describe what you need to do in detail in order to develop the `CourseManDemo` class and other related classes to work with the user interface.

2. Task requirements

Use the given attached file named ***studentpack.zip*** and complete the following tasks:

1. Copy & rename your (revised) `courseman2` package from previous assignment to `courseman3` as the top-level package.

(a) Copy into sub-package `courseman3.models` the following domain class from your (revised) `courseman` package: `Enrolment`.

2. Study the details of design update & class diagram of the application given in Section 3 to understand

its overall design **in this final phase**.

3. Update the code of the class `courseman3.CourseManDemo` with (1) image of the developer, (2) menus to manage enrolments. See sample GUI in Section 4.

- ***Hint***: you may use `JLable` with an image icon

4. Copy then complete the code of the partially written code `courseman3.controllers.Manager`, which is given in the attached file. Review the details about this class (Section 3 – design update) and your two classes `StudentManager` & `ModuleManager` from the previous assignment.

5. Refactor classes `StudentManager` and `ModuleManager` in package `courseman3.controllers` with new design mentioned in Section 3 – Design update. You may start with the partially written code `StudentManager`, which is given in the attached file.

6. Specify & implement these two methods, which is used for the next task (7) to validate if user input the correct student id and module code.

   (a) `StudentManager.getStudentByID(id: String): Student` – return Student object with specified id or null if not found.

   (b) `ModuleManager.getModuleByCode(code: String): Module` – return Module object with specified code or null if not found.

7. Similar to task (5), specify & implement class `courseman3.controllers.EnrolmentManager`. See sample GUI in Section 4.

- ***Hint:*** Receive `StudentManager` & `ModuleManager` objects as **attributes** so then you can use `getStudentByID()` & `getModuleByCode()` implemented before.

8. In class `courseman3.controllers.EnrolmentManager`, specify & implement the two methods.

   (a) `report(): void` – generate the initial report of all enrolment objects & display. See sample GUI in Section 4.

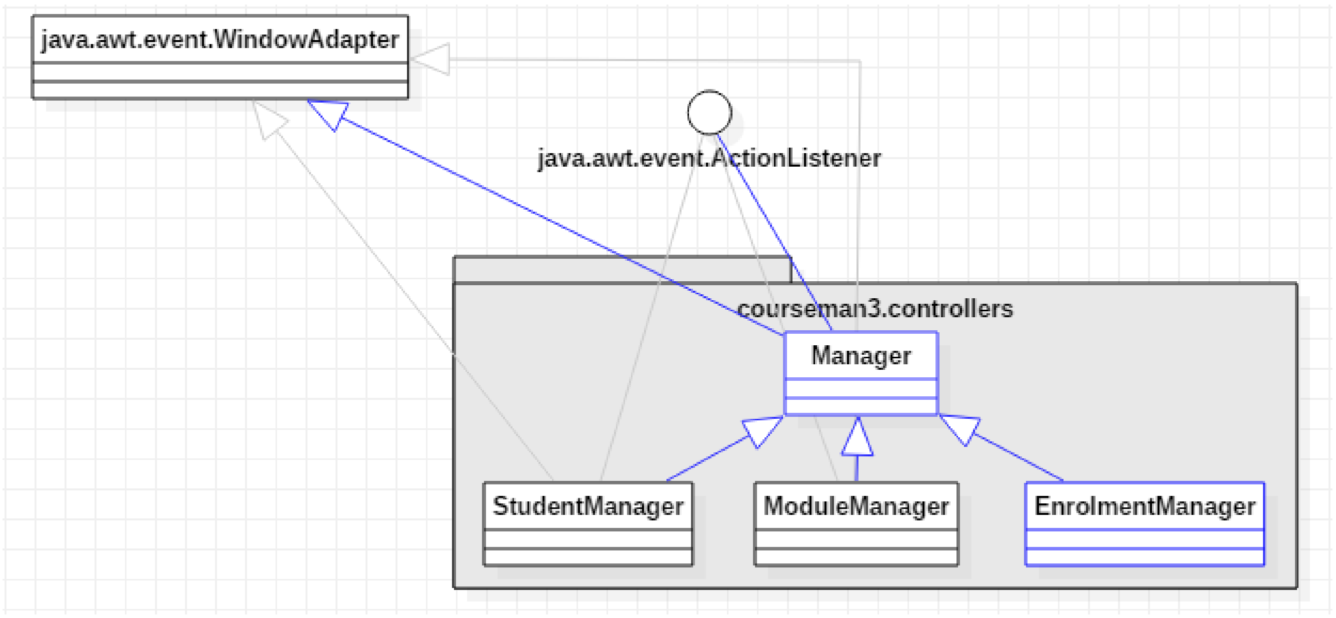
   (b) `reportAssessment(): void` - generate the initial report of all enrolment objects & display. See sample GUI in Section 4.

9. Run the `courseman3.CourseManDemo` program using the example objects that you used in CourseMan. The resulted GUIs should look like the ones shown in Section 4.

3. Design update & Concept class diagram (**final phase**)

1. Design Update

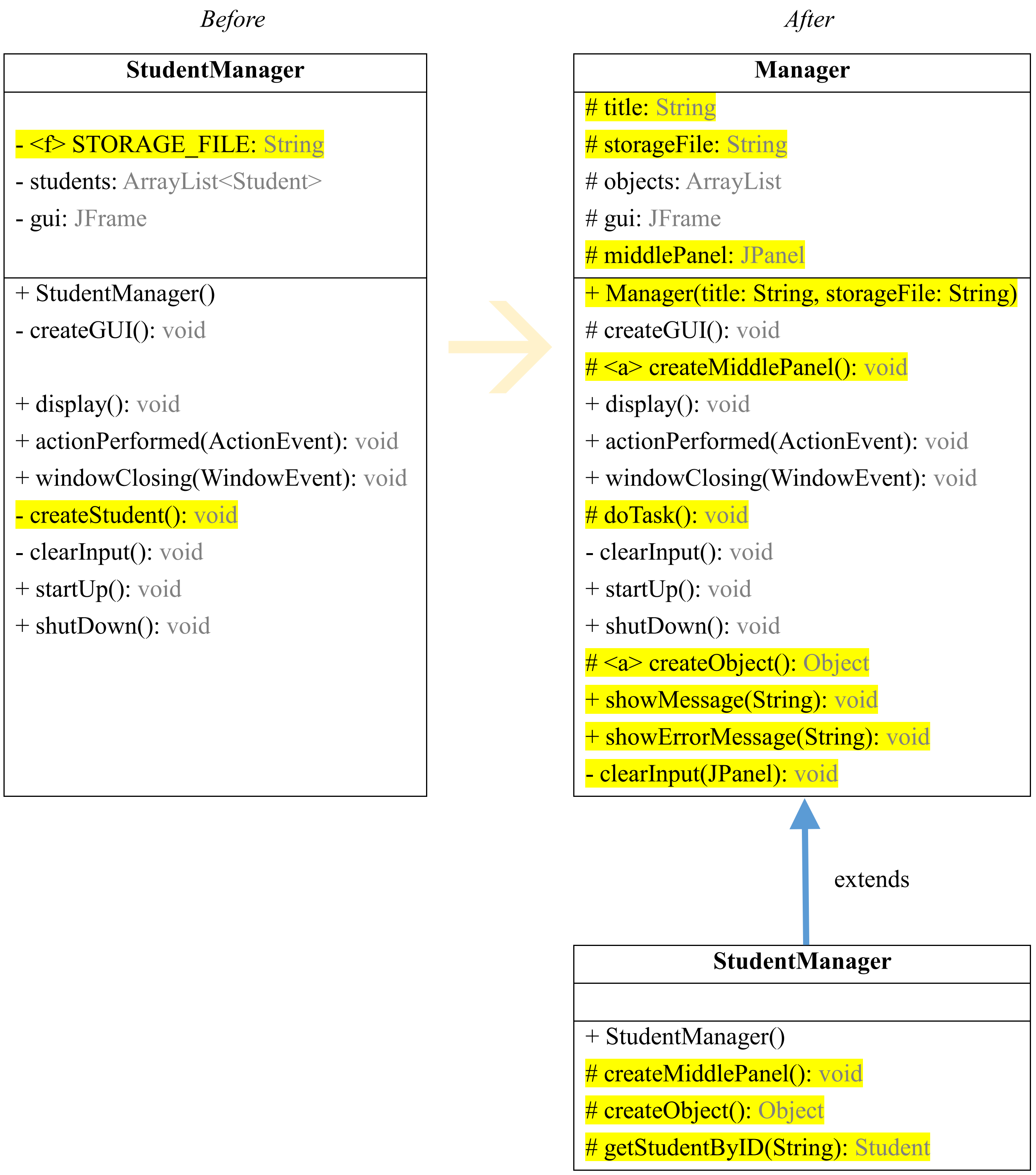
As we can see, the three data manager controllers for student, module and enrolment do the same tasks but for different domain classes, so they share many common attributes and methods. We decided to create a super class named `Manager`.



- The three managers does **not** extends `WindowAdapter` and implements `ActionListener` **directly** any more but through extending class `Manager`.

To make it clear, let’s have a look on `StudentManager` before and after adding class `Manager`.

- The shared **attributes** and **methods** are generalized to bring into `Manager` instead. Highlighted are the new or modified attributes and methods.


Before
After
StudentManager
- <f> STORAGE_FILE: String
- students: ArrayList<Student>
- gui: JFrame
+ StudentManager()
- createGUI(): void
+ display(): void
+ actionPerformed(ActionEvent): void
+ windowClosing(WindowEvent): void
- createStudent(): void
- clearInput(): void
+ startUp(): void
+ shutDown(): void
Manager
# title: String
# storageFile: String
# objects: ArrayList
# gui: JFrame
# middlePanel: JPanel
+ Manager(title: String, storageFile: String)
# createGUI(): void
# <a> createMiddlePanel(): void
+ display(): void
+ actionPerformed(ActionEvent): void
+ windowClosing(WindowEvent): void
# doTask(): void
- clearInput(): void
+ startUp(): void
+ shutDown(): void
# <a> createObject(): Object
+ showMessage(String): void
+ showErrorMessage(String): void
- clearInput(JPanel): void
extends
StudentManager
+ StudentManager()
# createMiddlePanel(): void
# createObject(): Object
# getStudentByID(String): Student

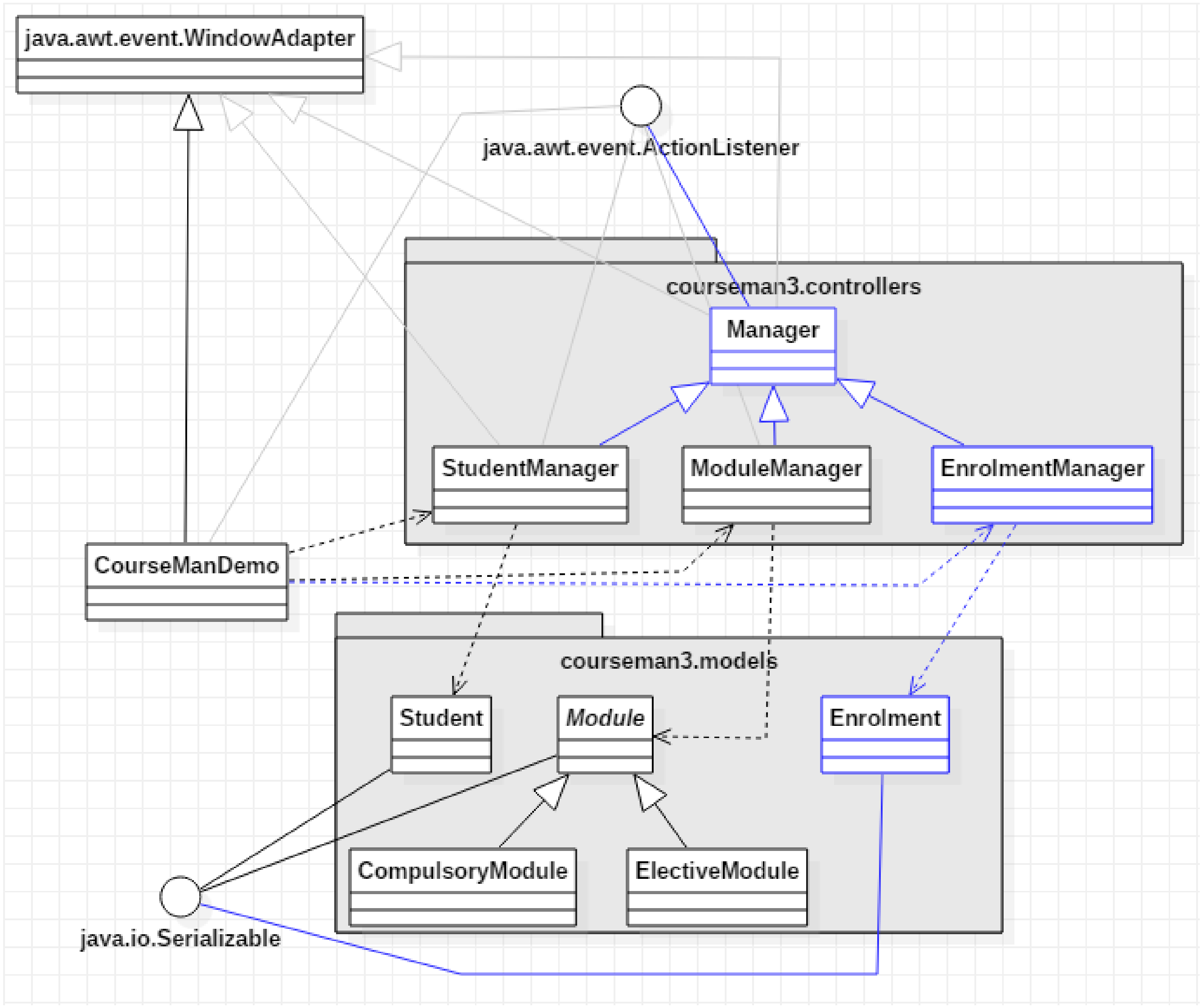
The table below gives more details from the design decision to the new attributes and methods.

The three managers		The super class Manager
- Each manager has a different title on the GUI and different storage file for storing data objects		- Manager: add two attributes title and storageFile o title: String o storageFile: String
- Each manager has attribute for array of data objects ~~o students: ArrayList<Student>~~ ~~o modules: ArrayList<Module>~~ ~~o enrolments: ArrayList<Enrolment>~~		- The three manager: remove these attributes - Manager: add attribute objects of more generalized data type: ArrayList o objects: ArrayList
- Each manager has the same GUI but different content in the middle panel	→	- Manager: add attribute middlePanel o middlePanel: JPanel - Manager: add abstract method to create middle panel o createMiddlePanel(): void
- Each manager is to do a specific task, in this case, creating object from user input on middle panel		- Manager: add method to do the specific task of this controller o doTask(): void - Manager: add abstract method to create object from user input on the GUI form o createObject(): Object
- Each manager needs to show user info message or error message, also clear the inputs when user click Cancel		- Manager: add utility methods to show info message, error message to user and clear user inputs inside a JPanel respectively o showMessage(String): void o showErrorMessage(String): void o clearInput(JPanel): void

2. Concept class diagram (final phase)

Below is the concept class diagram of the `CourseMan3` application, showing the associations between the classes. Refer to the design specification of each class for their internal design details. In this diagram,

- Lines with **black** color: **unchanged** connections
- **Blue** color: **new** classes and connections
- Lines with blurred **grey** color: **removed** connections



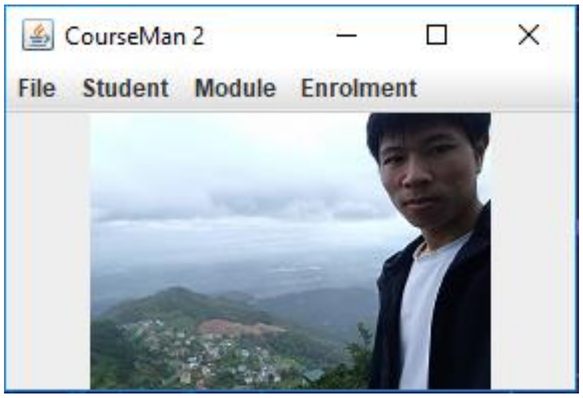
4. Sample Output & GUIs

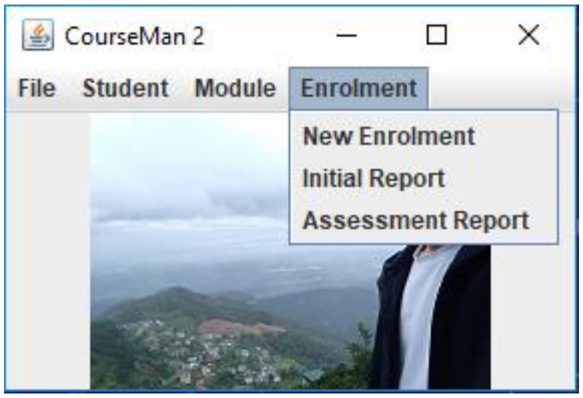
1. Console output updated with EnrolmentManager

```
Starting up...
StudentManager.loaded ...2 objects
Starting up...
ModuleManager.loaded ...3 objects
Starting up...
EnrolmentManager.loaded ...0 objects
Shutting down...
StudentManager.shutDown ...stored 2 objects
Shutting down...
ModuleManager.shutDown ...stored 3 objects
Shutting down...
EnrolmentManager.shutDown ...stored 6 objects
```

2. Main GUI with updates

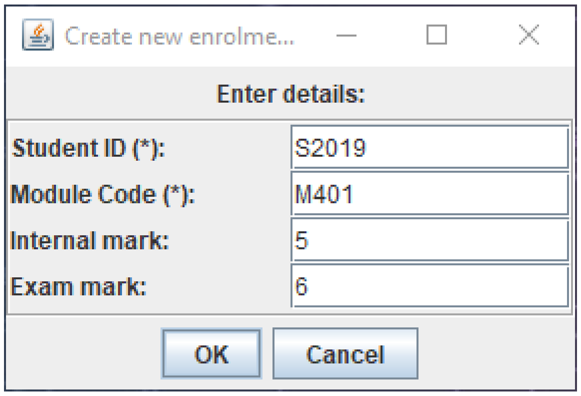
The main GUI that contains (1) an image of the creator & (2) a new menu Enrolment.





3. Manage Enrolment GUI

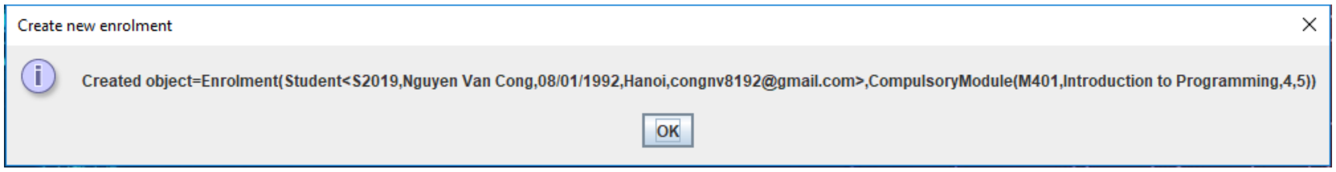
The create Enrolment GUI & message dialogs



Create new enrolme...

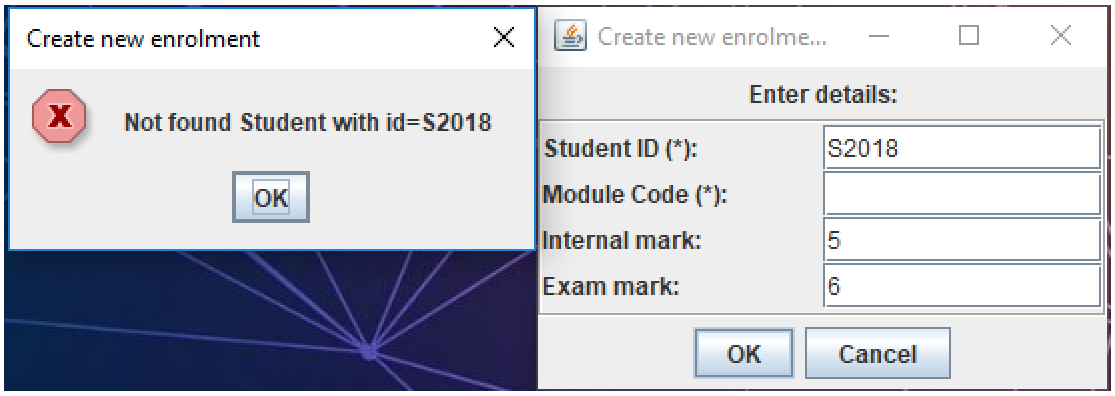
Enter details:

Student ID (*):	S2019
Module Code (*):	M401
Internal mark:	5
Exam mark:	6



Create new enrolment

Created object=Enrolment(Student<S2019,Nguyen Van Cong,08/01/1992,Hanoi,congnv8192@gmail.com>,CompulsoryModule(M401,Introduction to Programming,4,5))


Create new enrolment

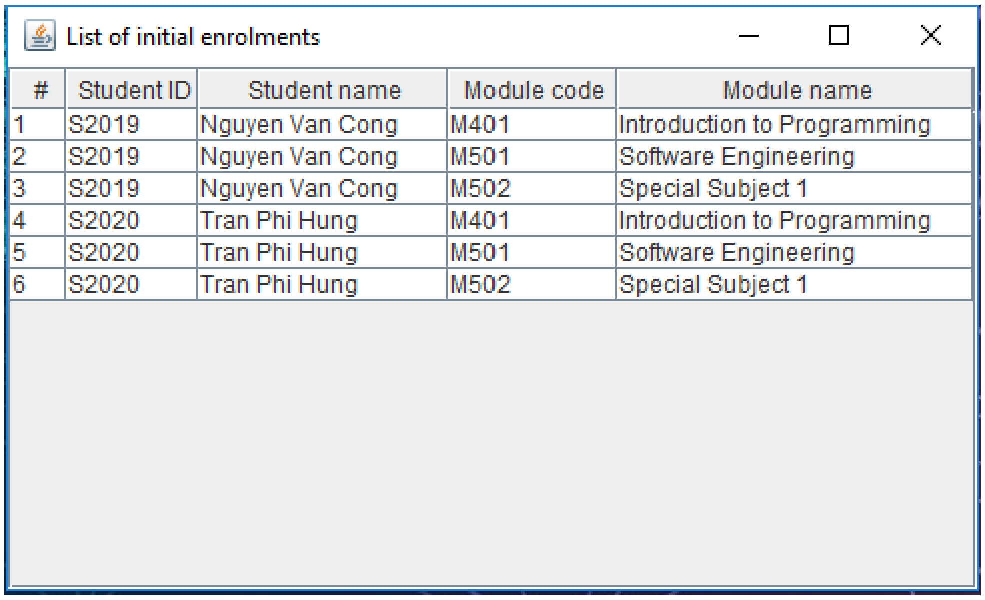
Not found Student with id=S2018

Create new enrolme...

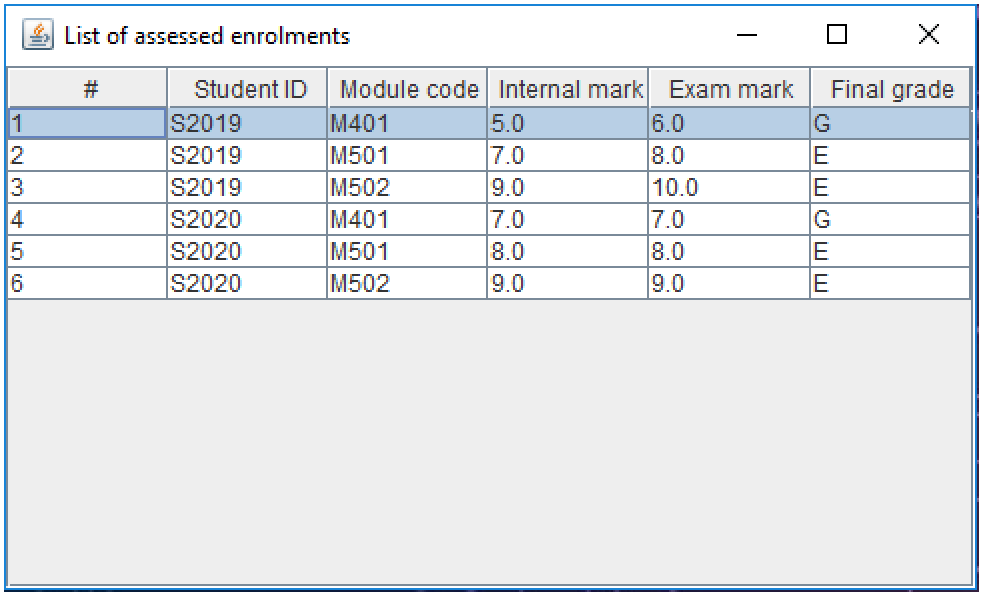
Enter details:

Student ID (*):	S2018
Module Code (*):	
Internal mark:	5
Exam mark:	6

The Enrolment initial report & assessment report


List of initial enrolments

#	Student ID	Student name	Module code	Module name
1	S2019	Nguyen Van Cong	M401	Introduction to Programming
2	S2019	Nguyen Van Cong	M501	Software Engineering
3	S2019	Nguyen Van Cong	M502	Special Subject 1
4	S2020	Tran Phi Hung	M401	Introduction to Programming
5	S2020	Tran Phi Hung	M501	Software Engineering
6	S2020	Tran Phi Hung	M502	Special Subject 1


List of assessed enrolments

#	Student ID	Module code	Internal mark	Exam mark	Final grade
1	S2019	M401	5.0	6.0	G
2	S2019	M501	7.0	8.0	E
3	S2019	M502	9.0	10.0	E
4	S2020	M401	7.0	7.0	G
5	S2020	M501	8.0	8.0	E
6	S2020	M502	9.0	9.0	E

5. Submission requirements

You must submit a single zip file, which contains the `courseman3` package specified in Section 1.

The zip file name must be of the form `a3_Sid.zip`, where *`Sid`* is your student identifiers (the remaining parts of the file name must not be changed!). For example, if your student id is 1601040001 then your zip file must be named `a3_1601040001.zip`.

IMPORTANT: failure to submit the file as described will result in no marks being given!

NO PLAGIARISM, if plagiarism, 0 mark will be given!