SCIT

School of Computing and Information Technology
Faculty of Engineering & Information Sciences

CSIT121
Object Oriented Design and Programming
Assignment 3

INSTRUCTIONS TO CANDIDATES

1. The assignment consists of two parts. This is the part 1 of the assignment.
2. Part 2 is Moodle quiz. Should be done in class.
3. The name of the program must be **YourName_ClassListNo_A3.**java (Only one Java file); remember to replace **YourName** by your actual "shorter" name.
4. Total mark of Assignment 1 is 10 marks; 4 marks for Part II.

Your program, should begin with

// Full Name:
// Part time or Full Time
// Class List No:
// **Demo one or two tasks, by default, you demo ONLY one**
// Declaration: ...... tell me if it is your own work .... and whether you have
// passed your program to your friends.

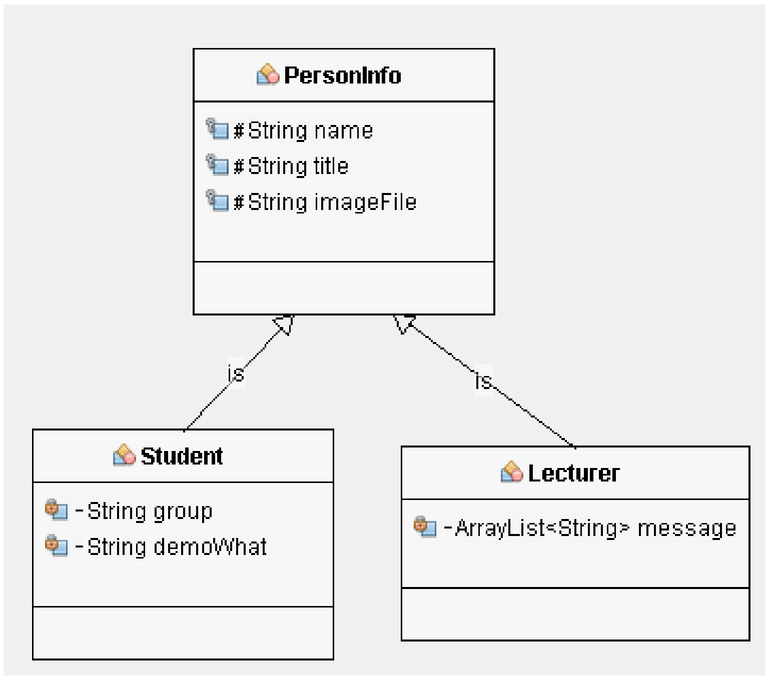
Objectives:

Practice java programming with GUI, Collections.

Task 1: (6 marks)

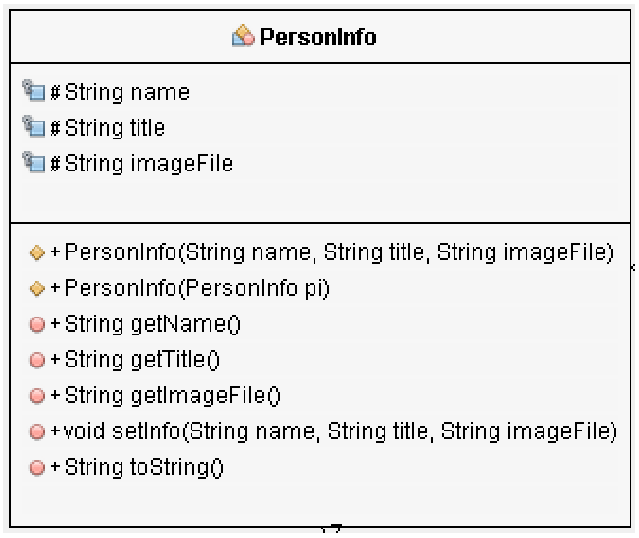
Every term, students always queue up demoing their labs and their assignments to me during the tutorial sessions. To prevent students standing too long in the queue, I would like to have a demo system. This demo system allocates, *randomly*, a student proceeds to demo his/her works to me.

Let us begin by exploring the following simple UML relationship:



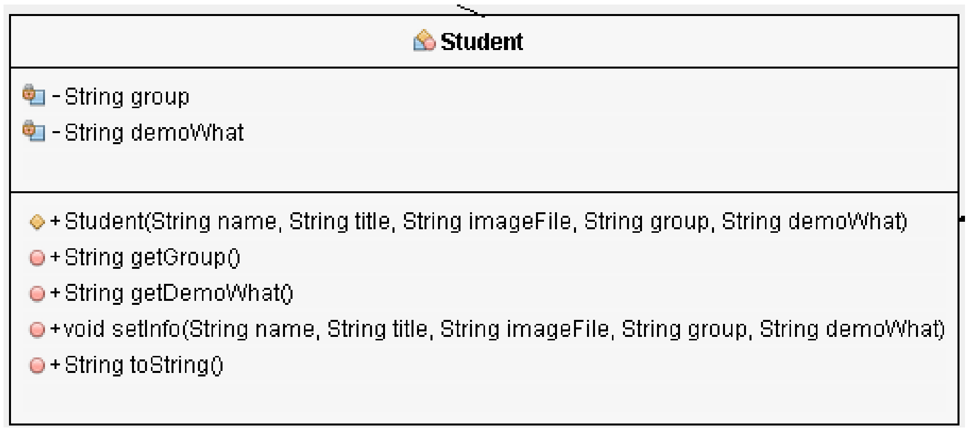
In short, `Student` and `Lecturer` classes are two subclasses of `PersonInfo` class.

We now look at the details of each class:


PersonInfo
#String name
#String title
#String imageFile
+PersonInfo(String name, String title, String imageFile)
+PersonInfo(PersonInfo pi)
+String getName()
+String getTitle()
+String getImageFile()
+void setInfo(String name, String title, String imageFile)
+String toString()

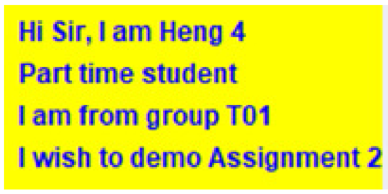
`PersonInfo` class has three instance variables, name, title, and an image file name. **Note that**, *very important*, the student image files are stored as "1.jpg", "2.jpg" etc. Lecturer image file, for (me) testing purpose, please use "heng.jpg". All image files should be placed in the main working directory, **no path**, **no subdirectory**. The title stores, for example, "Part time student", "Full time student", "Lecturer", "Tutor" etc. The `toString` method simply returns a string with the following format, for example,

Nancy Lim Lily
Part time student



```
Student
- String group
- String demoWhat
+ Student(String name, String title, String imageFile, String group, String demoWhat)
+ String getGroup()
+ String getDemoWhat()
+void setInfo(String name, String title, String imageFile, String group, String demoWhat)
+ String toString()
```

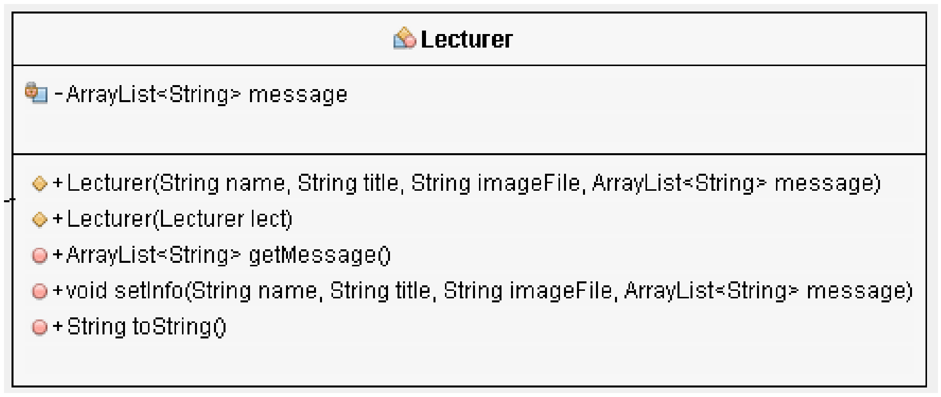
`Student` class is a subclass of `PersonInfo` class. Additional info for student, he/she has a tutorial group and wishes to demo what on the demo day; for example, Tutorial 2 or Lab 2, Assignment 3, Assignment 1 etc. The `toString` method returns a String with the following format:



```
Hi Sir, I am Heng 4
Part time student
I am from group T01
I wish to demo Assignment 2
```

this info came from the two toString's

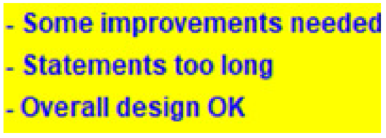
Common terms used by students to address me: "Sir", "Dr Heng", "Prof", "Mr Heng" … 老师, "cikgu", 선생님. You can randomly generate and select one of these callings in display. Most of the display info should be extracted from the two toString methods as listed in the above two classes. You can see them, later, when I show you some of the screenshots.



```
Lecturer
-ArrayList<String> message
+Lecturer(String name, String title, String imageFile, ArrayList<String> message)
+Lecturer(Lecturer lect)
+ArrayList<String> getMessage()
+void setInfo(String name, String title, String imageFile, ArrayList<String> message)
+String toString()
```

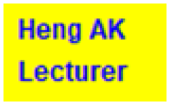
`Lecturer` class is a subclass of `PersonInfo` class. Other than the instance variables defined in the superclass, it has an instance variable which is an `ArrayList` object called `message`. Usually in grading, I aways have a set of standard remarks, for example, “Well done”, “Statements too long”, “Bad indentations” etc. This message list stores some of my comments after student demos his/her work to me.

The toString method returns a string consisting of all remarks the marker placed in the message list; of course, the marker also put down his name and his title; for example:



```
- Some improvements needed
- Statements too long
- Overall design OK
```

this info came from toString method



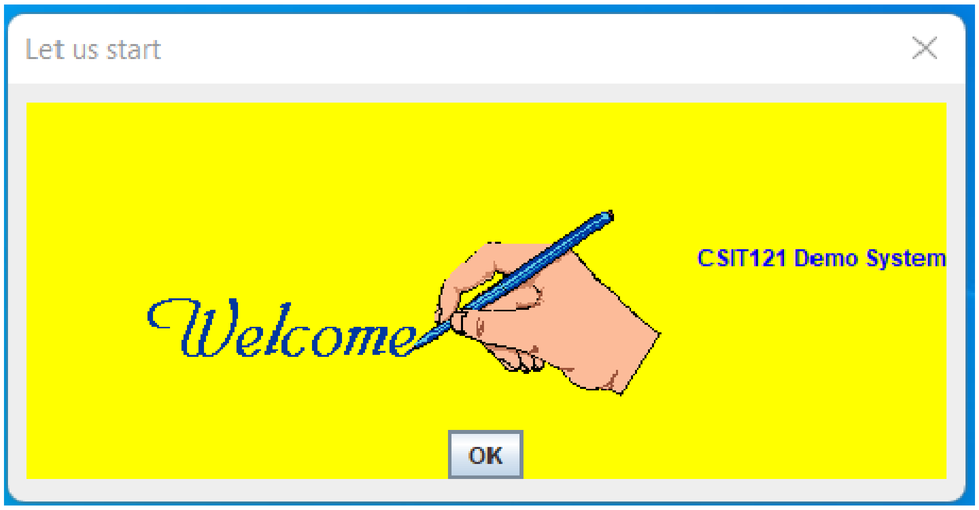
```
Heng AK
Lecturer
```

this info came from super class toString

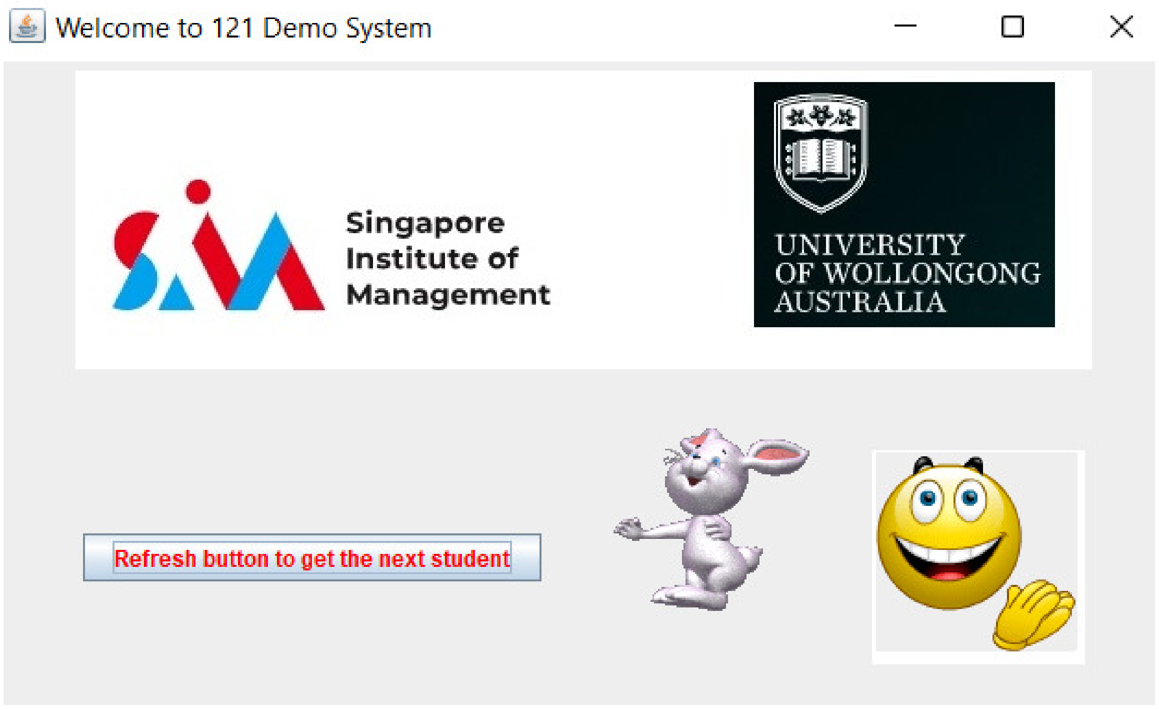
Let us explore the following interactions and displays before we talk about the designs. I propose three *possible* tasks to be implemented. **You only submit of the implementations**. You can choose your own way of organizing the GUI, just respect *“Look and feel”*!!!

Possible Task 1: Design using javax

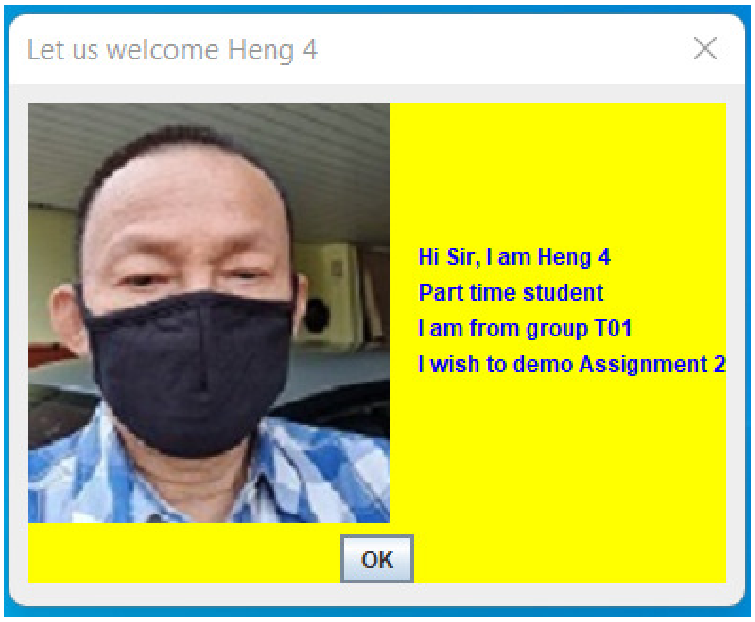
When you enter to the system, it displays



When OK button is pressed, it displays the second frame:


Welcome to 121 Demo System
Singapore Institute of Management
UNIVERSITY OF WOLLONGONG AUSTRALIA
Refresh button to get the next student

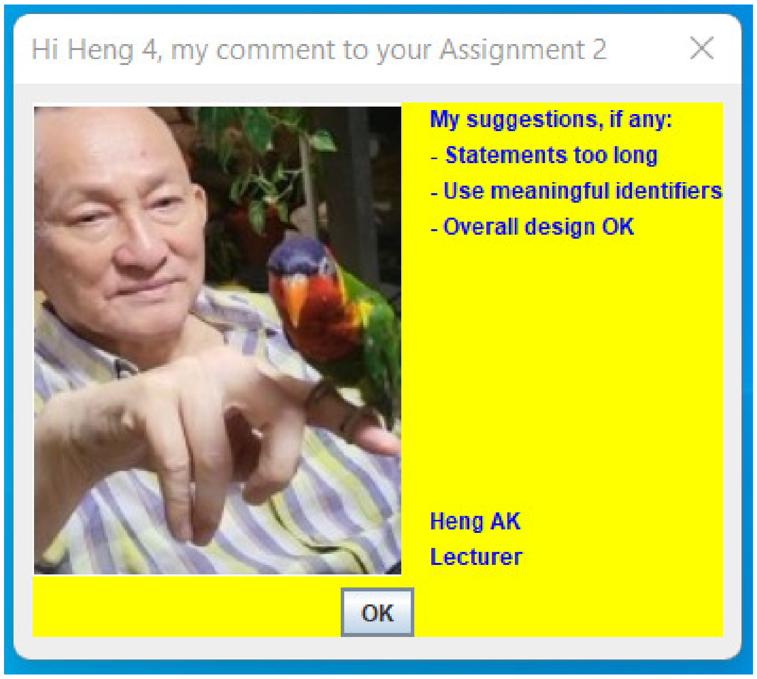
Now you can refresh (press) the button. The system ***randomly*** chooses a student to demo his/her lab/assignment to me; and the student also introduce himself/herself by telling me what he/she wants to demo and from which tutorial group.


Let us welcome Heng 4

Hi Sir, I am Heng 4
Part time student
I am from group T01
I wish to demo Assignment 2

The student called “Heng 4”. The frame displays the right welcome title.

After student completes the demo, you then press the OK button, lecturer gives comment to the demo:


Hi Heng 4, my comment to your Assignment 2

My suggestions, if any:
- Statements too long
- Use meaningful identifiers
- Overall design OK

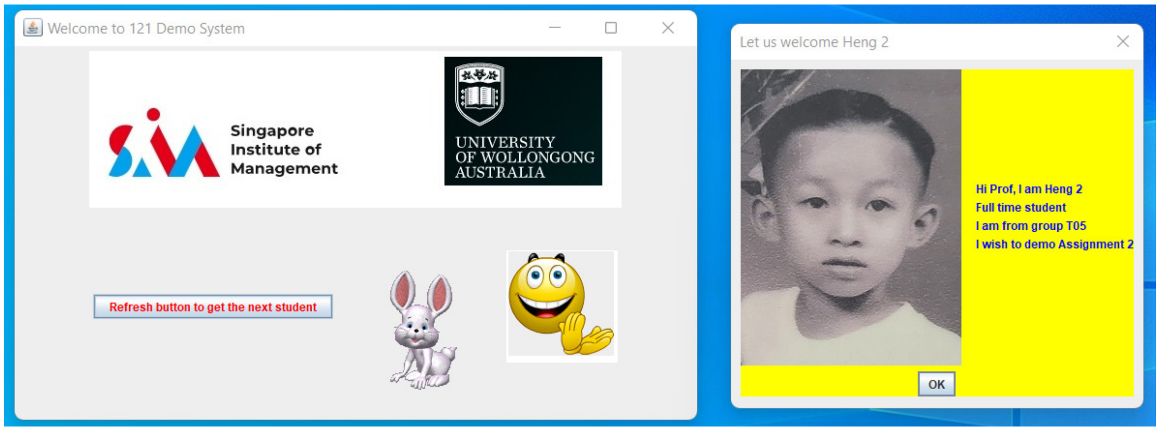
Heng AK
Lecturer

To those students who took 111 during the last semester should know my marking habits; I always have a set of marking comments to your assessments. The comments I give to students also randomly selected from a pool of “comments”, no duplication in display.

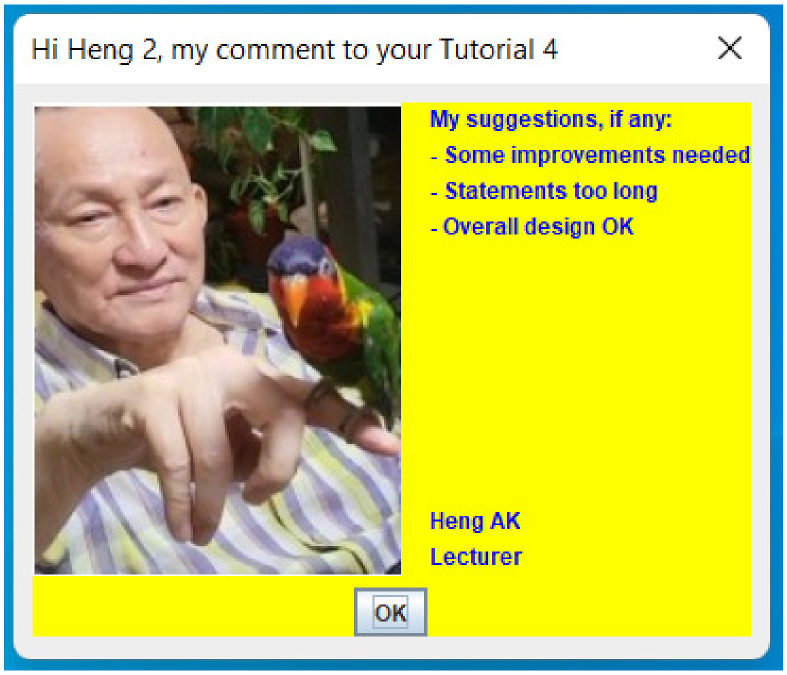
The frame also displays a friendly title 😊

If OK button is pressed again in the above frame. You return to the refresh button frame to get the next student:

Let us press one more time to the refresh button to get the 2nd student; again, if OK button is pressed, lecturer gives comments to the 2nd student:


Welcome to 121 Demo System
Singapore Institute of Management
UNIVERSITY OF WOLLONGONG AUSTRALIA
Refresh button to get the next student
Let us welcome Heng 2
OK

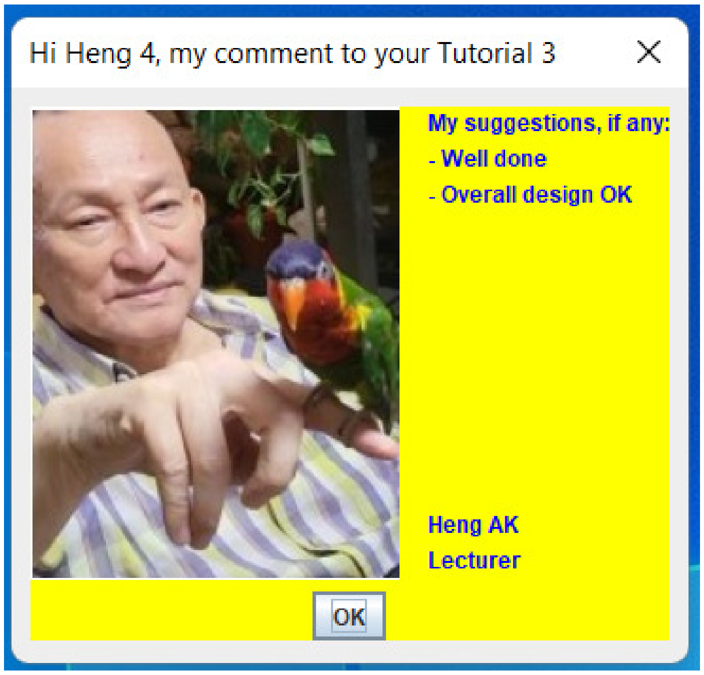
Hi Prof, I am Heng 2
Full time student
I am from group T05
I wish to demo Assignment 2



My suggestions, if any:
- Some improvements needed
- Statements too long
- Overall design OK

Heng AK
Lecturer

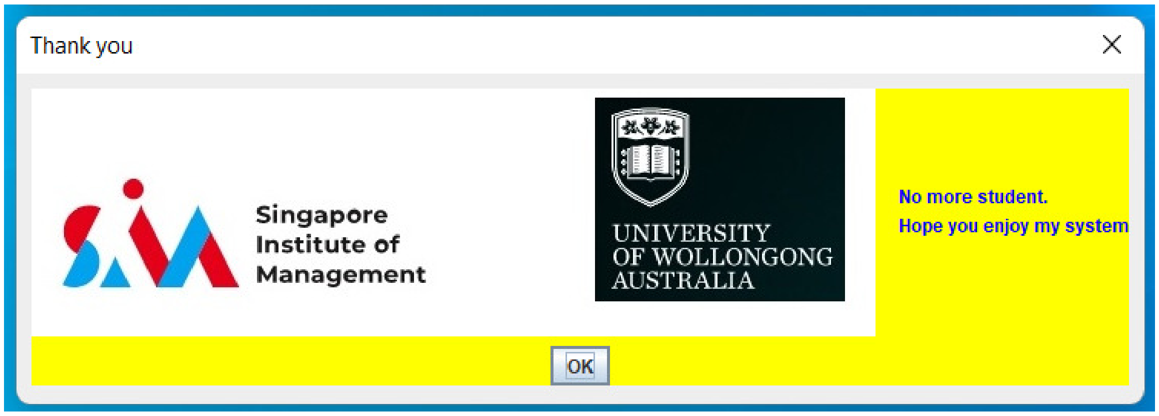
Note the following screen shot; if I say, “well done”, no additional remark is included:


Hi Heng 4, my comment to your Tutorial 3

My suggestions, if any:
- Well done
- Overall design OK

Heng AK
Lecturer

If there is no more student in the system for demoing, it displays:

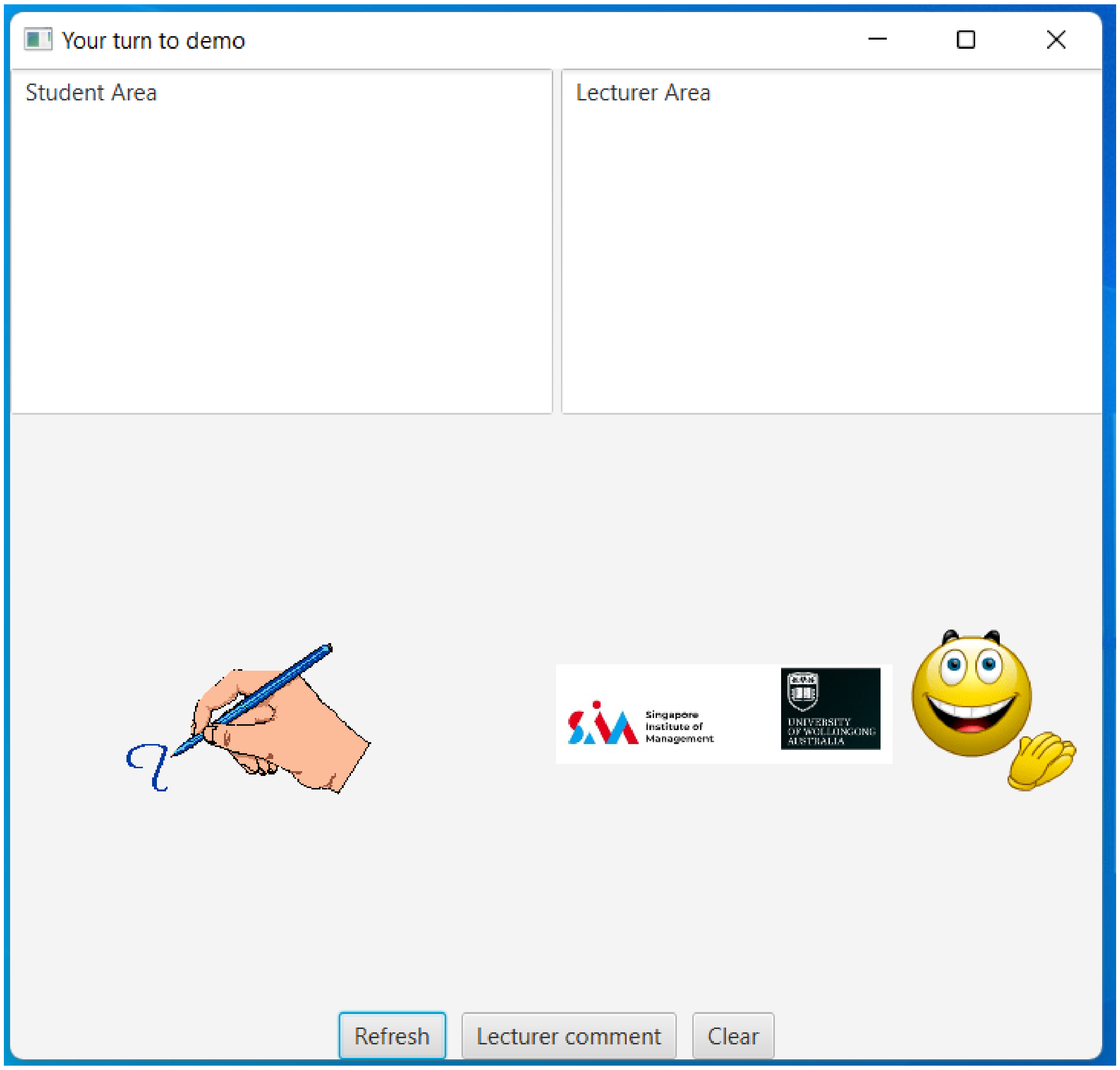

Thank you

No more student.
Hope you enjoy my system

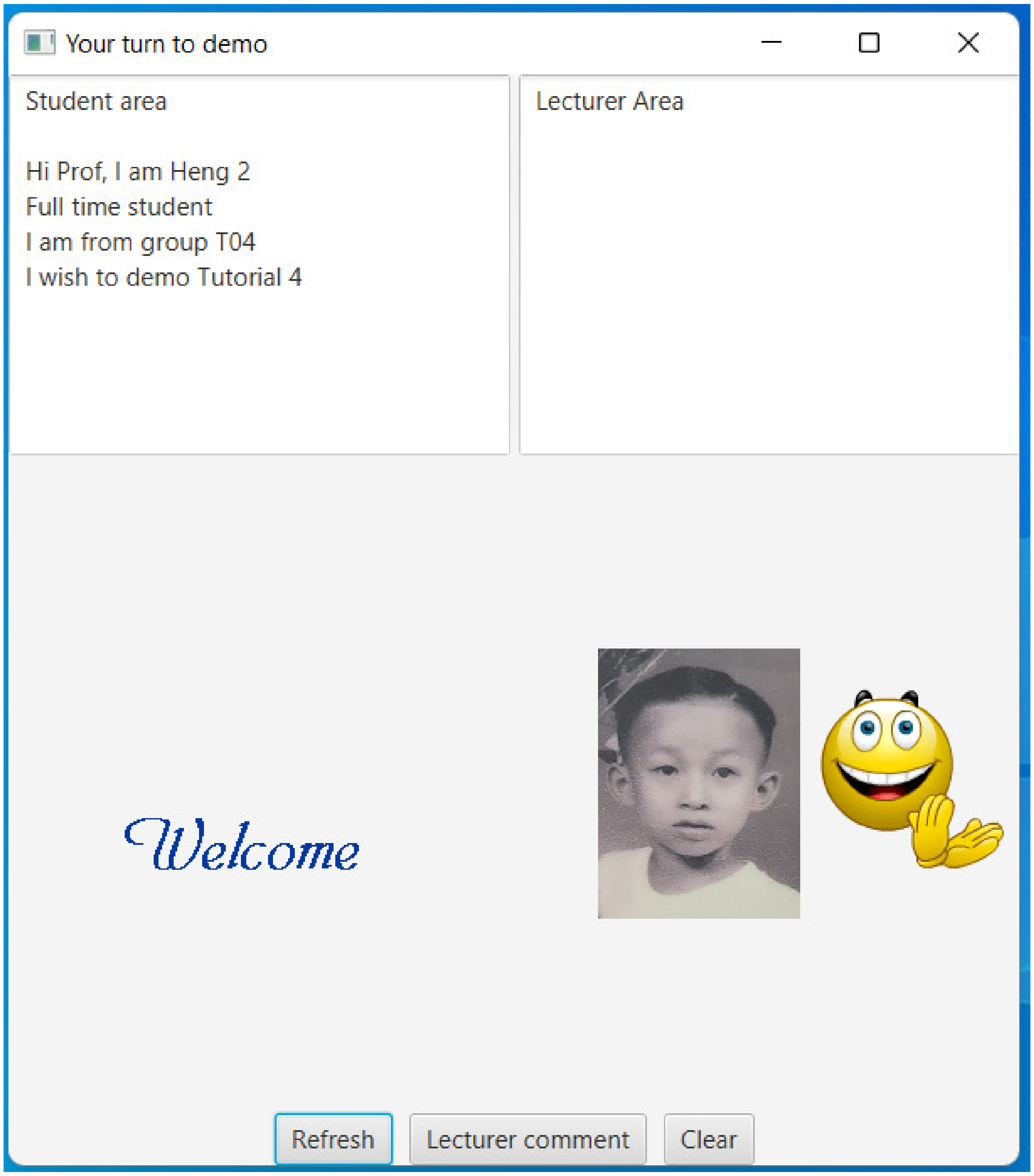
Singapore Institute of Management
UNIVERSITY OF WOLLONGONG AUSTRALIA
OK

Possible Task 2: Using javafx

When you enter to the system, it displays:



You work on one scene, but with a few possible layouts (top, middle and button). When you press the refresh button, a student comes in, i.e., some introduction to student is displayed in the student area (text area), and his/photo also displays:



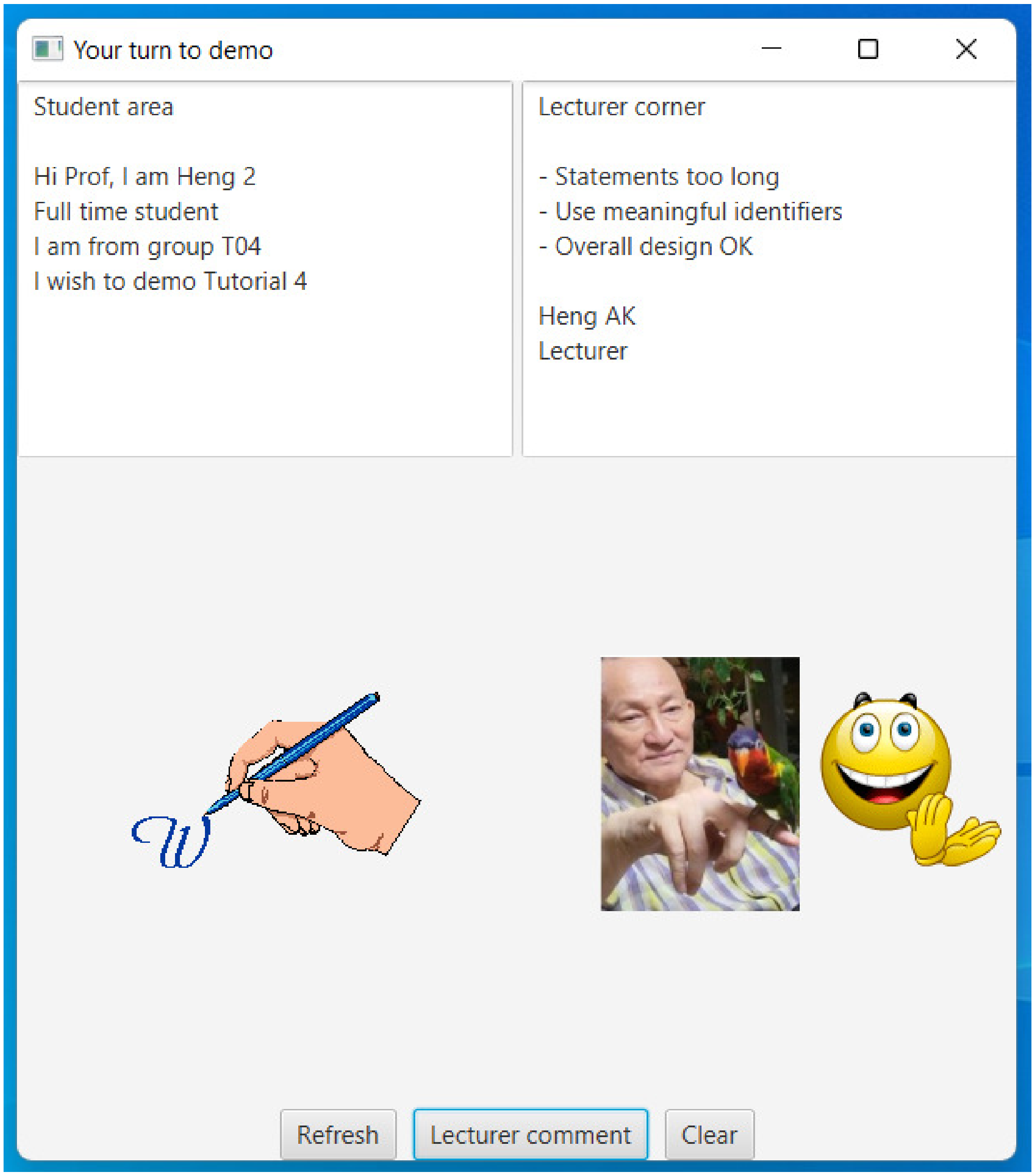
Your turn to demo

Student area

Hi Prof, I am Heng 2
Full time student
I am from group T04
I wish to demo Tutorial 4

Lecturer Area

Now you can press the lecturer comment button to see what lecturer says:



Student area

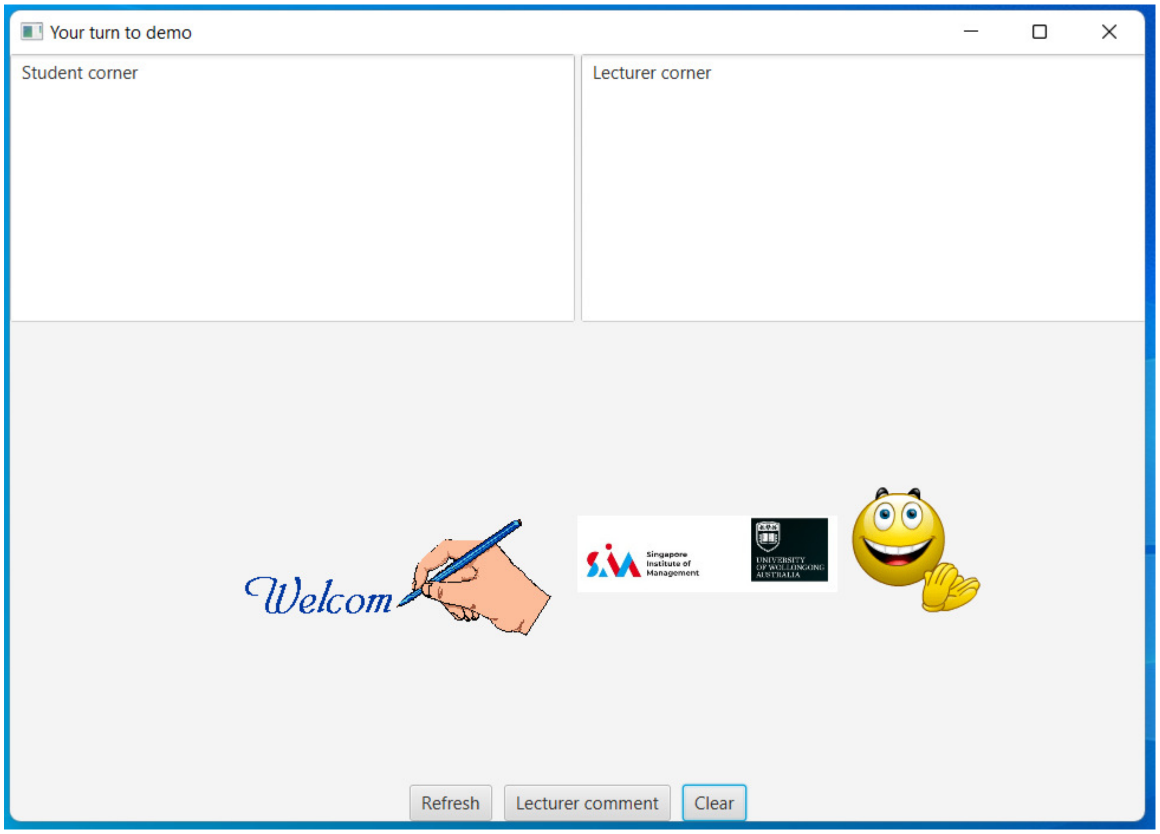
Hi Prof, I am Heng 2
Full time student
I am from group T04
I wish to demo Tutorial 4

Lecturer corner

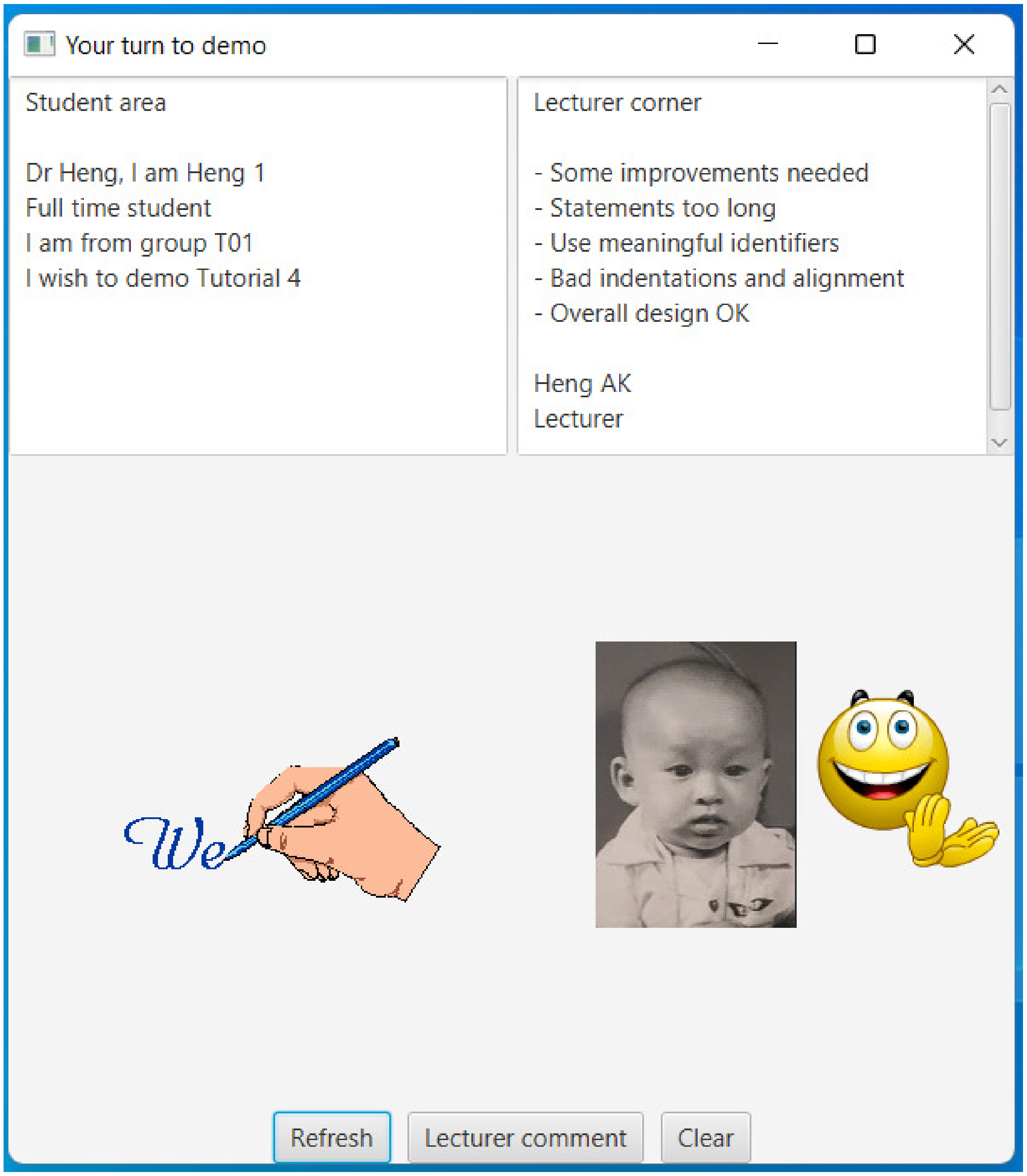
- Statements too long
- Use meaningful identifiers
- Overall design OK

Heng AK
Lecturer

You press the clear button:



Let you see one more interaction and displays:


Your turn to demo

Student area

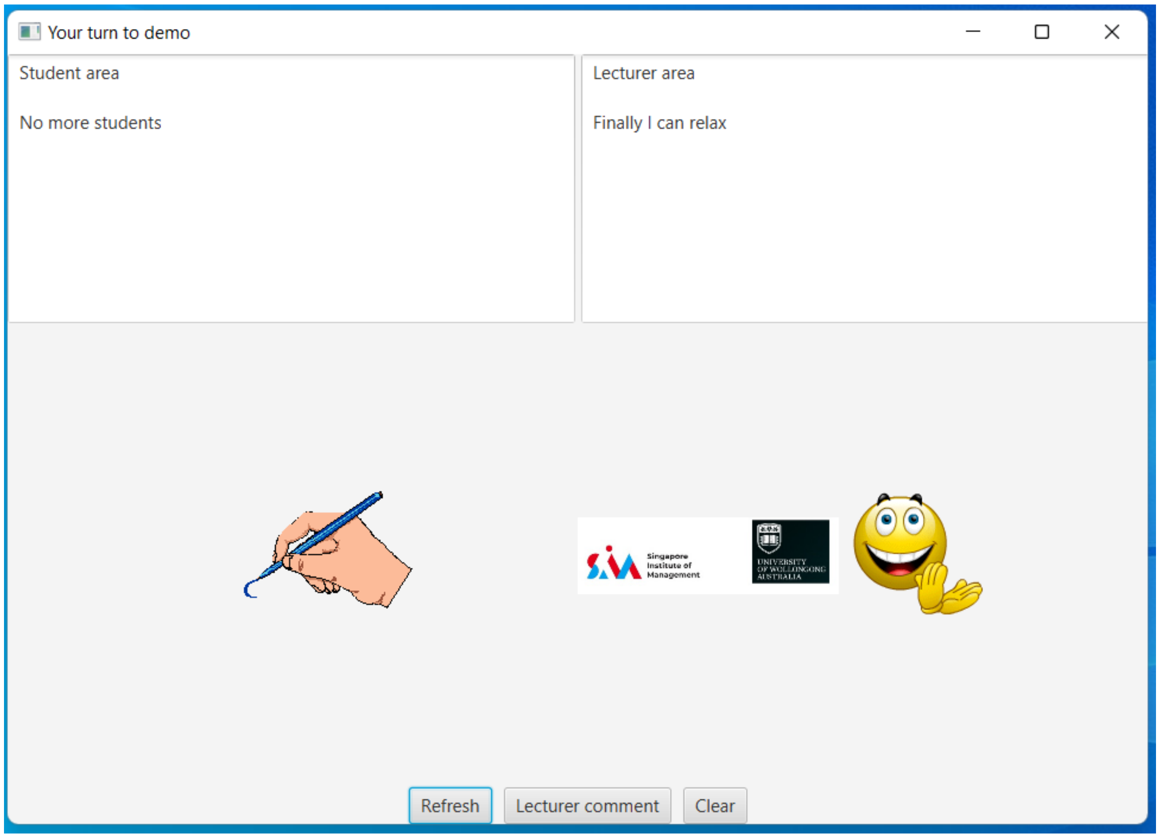
Dr Heng, I am Heng 1
Full time student
I am from group T01
I wish to demo Tutorial 4

Lecturer corner

- Some improvements needed
- Statements too long
- Use meaningful identifiers
- Bad indentations and alignment
- Overall design OK

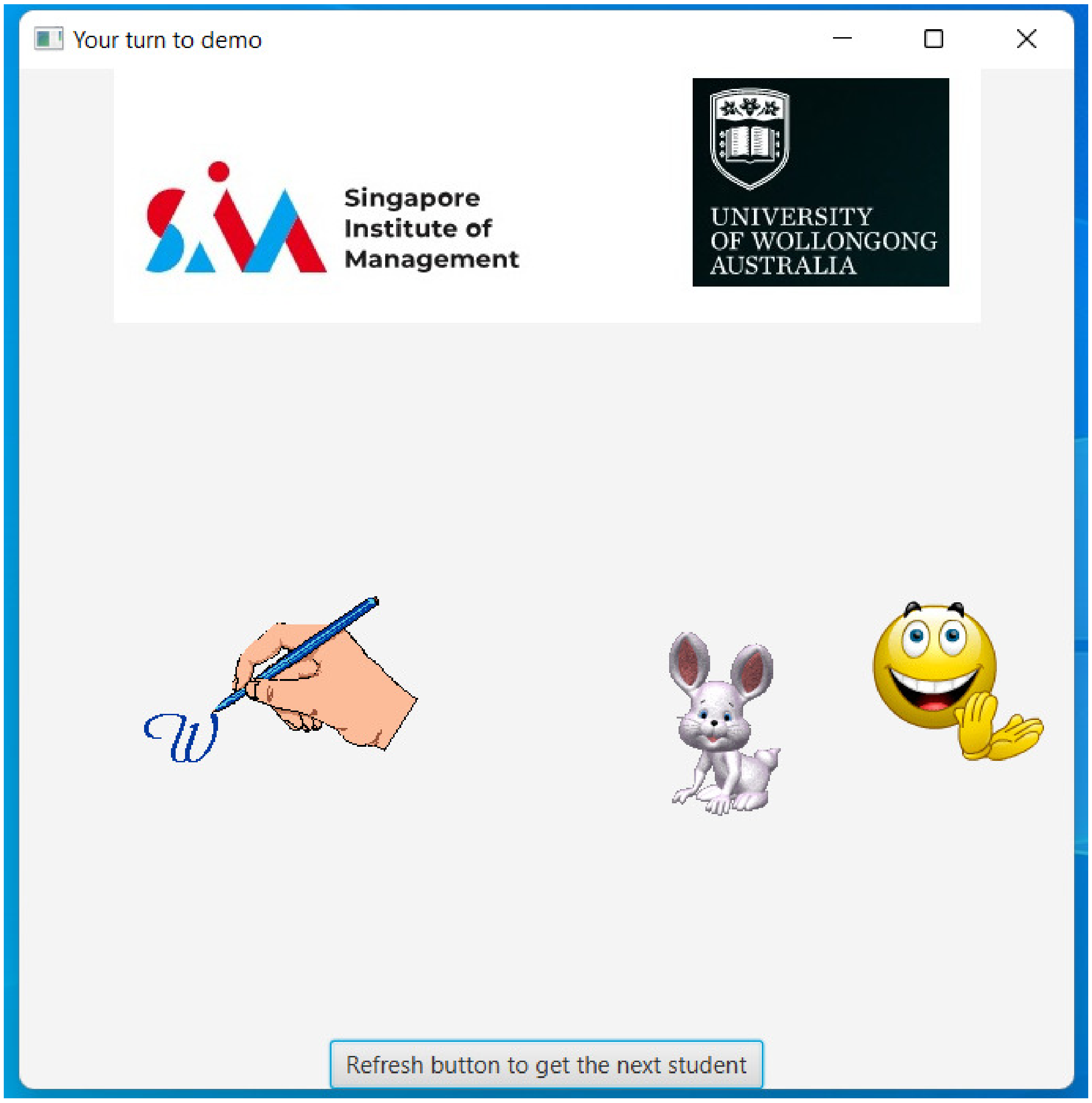
Heng AK
Lecturer

When there is no more student for demo, it displays:

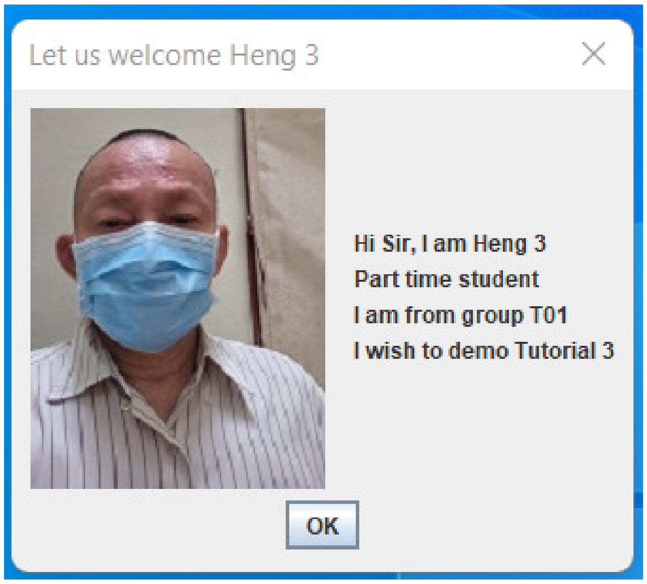


Possible Task 3: Combine javax and javafx

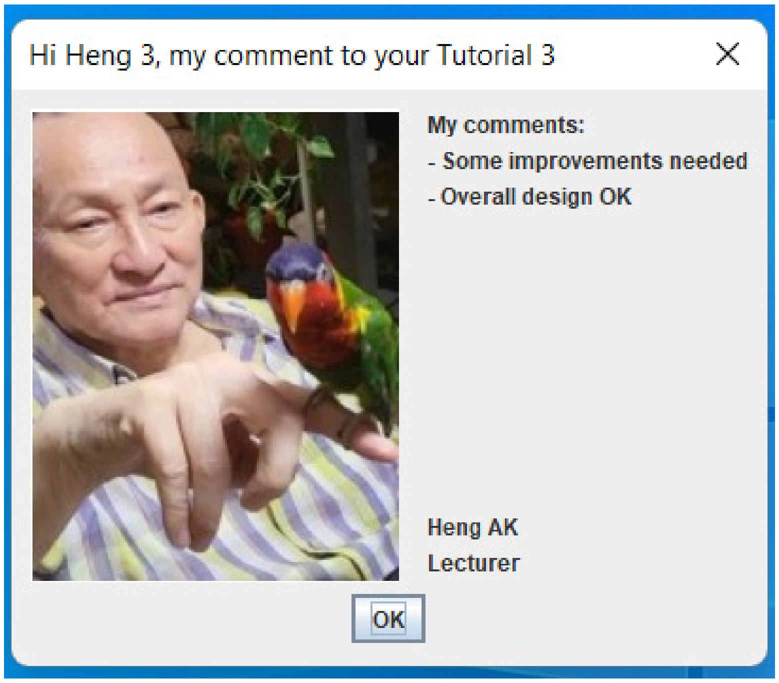
You use javafx to organize the GUI placed in the scene:


Your turn to demo
Singapore Institute of Management
UNIVERSITY OF WOLLONGONG AUSTRALIA
Refresh button to get the next student

When you refresh the button, you use JOptionPane to display the dialogs


Let us welcome Heng 3

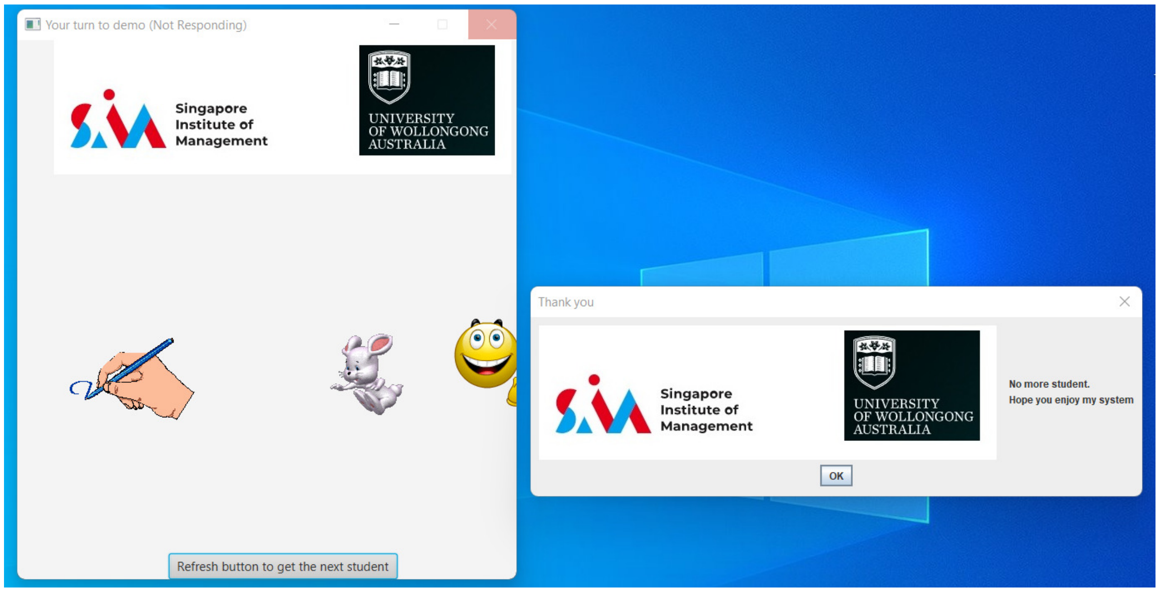
Hi Sir, I am Heng 3
Part time student
I am from group T01
I wish to demo Tutorial 3


Hi Heng 3, my comment to your Tutorial 3

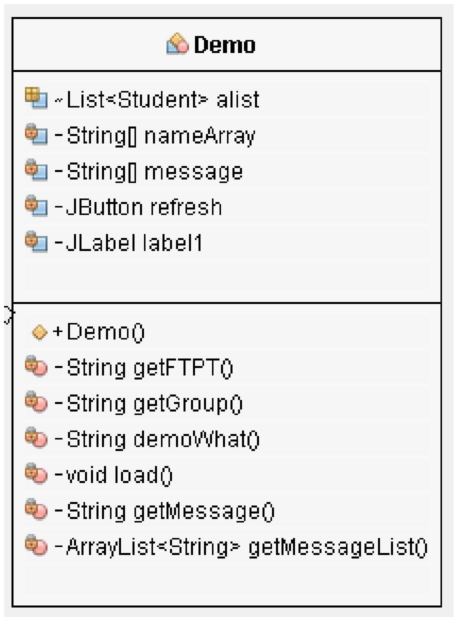
My comments:
- Some improvements needed
- Overall design OK

Heng AK
Lecturer

If no more student in the list, the following shows the final screen.

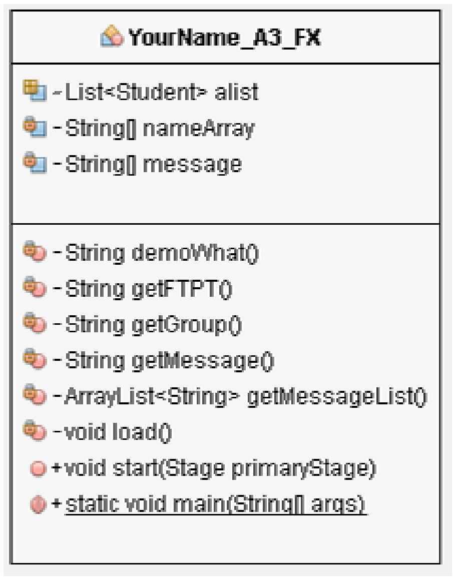

Your turn to demo (Not Responding)
Singapore Institute of Management
UNIVERSITY OF WOLLONGONG AUSTRALIA
Refresh button to get the next student
Thank you
No more student.
Hope you enjoy my system
OK

Finally, we can talk about the GUI implementations.



```
Demo
~List<Student> alist
-String[] nameArray
-String[] message
-JButton refresh
-JLabel label1

+Demo()
-String getFTPT()
-String getGroup()
-String demoWhat()
-void load()
-String getMessage()
-ArrayList<String> getMessageList()
```



```
YourName_A3_FX
~List<Student> alist
-String[] nameArray
-String[] message

-String demoWhat()
-String getFTPT()
-String getGroup()
-String getMessage()
-ArrayList<String> getMessageList()
-void load()
+void start(Stage primaryStage)
+static void main(String[] args)
```

The above UML diagrams show two different platforms, Demo class extends JFrame (design in javax environment) and YourName_A3_FX class extends Application (design in javafx environment). Both designs have some common methods:

- `getFTPT` method generates and returns either *Part time student* or *Full time student.*
- `getGroup` method generates and returns a tutorial group, *T01, T02* etc.
- `demoWhat` method generates and returns what assessment task students wish to show me, for example, *Assignment 1, Lab 3, Tutorial 2* etc.

- `getMessage` method generates and returns my usual messages to students, for example, *Statements too long, Bad indentation, Well done* etc.

A few private variables (can be final):

- An array called `nameArray` which stores some student names
- An array called `message` which stores those messages that I place in students' assessments
- A list called `alist` which store a list of `Student` objects; and `load` is a method that construct this list.

Basic Algorithm for your design: You repeats the following until the list is empty

- You shuffle the list
- You get the 1st student to demo
- You organize the GUI interactions (you can watch a video that I put in Moodle)
- You remove the 1st element from the list

For me to test your system, you can use the following image file names:

1.jpg, 2.jpg, 3.jpg to store student image files (3 students should be enough for me to test your product)

Other image file names you can use, for example, heng.jpg, sim.jpg, simuow.jpg, clab.gif, rabbit.gif, welcome.gif, thumbUp.jpg …

Note that image file names are case sensitive. You don't have to upload them. What you store in these file names are less important to me. You should assume that all image files are in the directory that you store your Java programs; for example, in javax, you say

```
Icon ic = new ImageIcon ("heng.jpg");
```

in javafx, you say

```
Image im = new Image ("heng.jpg");
```

For grading, you just upload one of the above programs (one of the three possible tasks). If you work on just one possible task, maximum marks you can get for this assignment is 5.5 / 6; to get full marks, you should do any two possible tasks.

Therefore, your program file header must begin with:

// Name:
// Class List No:
// I will demo 1 or 2 program(s)
// etc.
// Declaration:

IMPORTANT

Put all your classes in a file called **`YourName_ClassListNo_A3.java`** and make sure that this file can be compiled and can be executed. Upload **ONLY** this file to Moodle**. ALL ZIP FILE SUBMITTED WILL BE REJECTED. You don't have to upload the image files**

No re-submission will be allowed after grading.

In the declaration:

// Tell me if it is your own work, and whether you have passed your
// program to your friends etc etc etc
// and willing to accept whatever penalty given to you.

- **Wrong file name: -0.5 mark**
- **No declaration, no name etc: -0.5 mark**
- **Failing to demo: -1 marks**