



Report

CE/CZ2002: Object-Oriented Design & Programming

Building an OO Application

GROUP 1

BY: Khoo Chee Yang U1920289A

Kevin Liu Kai U1921021F


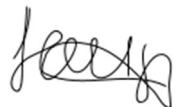


Peh Swee Sing U1922871H

Huang Neng Qi U1921454E

2020/2021 SEMESTER 1

NANYANG TECHNOLOGICAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Declaration of Original Work

Name	Course	Lab Group	Signature / Date
Khoo Chee Yang	CZ2002	SS13	25 th Nov 2020 
Kevin Liu Kai	CZ2002	SS13	25 th Nov 2020 
Peh Swee Sing	CZ2002	SS13	25 th Nov 2020 
Huang Neng Qi	CZ2002	SS13	25 th Nov 2020 

Youtube Link:

<https://youtu.be/SfLSee1KLJY>

Design Considerations

Objective: To design and develop a My Student Automated Registration System (MySTARS), a university application meant for each School's academic staff and undergraduate students of NTU. This application allows the creation of courses, adding of student records, along with the registration of courses and students.

OO Concepts:

We integrated OO concepts like Encapsulation, Abstraction, Inheritance, Polymorphism whenever suited to make the application more flexible, extensible and easy to maintain which allows reusability, extensibility and maintainability and decreased fragility of the code. This also allows future changes in the application.

We use Abstraction to select data that contains general attributes that can be shared by its subclasses. For subclass Student and Staff, since they contain similar attributes like the gender, name, id and nationality, we created an abstract Account class so that these general attributes can be inherited to them. Future accounts can also inherit from the Account class.

We use Encapsulation for all of our object classes, like Account, Courses, IndexNumber, and Lesson. Instance variables inside it are kept private and we create public accessor and mutator methods to allow other classes to get and set the information of some of the variables in the object, hence they can only take information that they need to know, and alter the information that they need to alter from the logic we implement in our getters and setters.

We used Inheritance, so that we can reuse the attributes and methods of existing classes, Student class which extends Account class, to reduce duplication of codes and make our class more specific in what they need to do.

We use Polymorphism in our Model interface. Models is implemented by all model subclasses like CourseModel, LessonModel, IndexNumberModel, StudentModel and StaffModel and override the Model's method ,populate(), to populate their respective model class. With this we do not need to come up with different method names for each similar task.

Single responsibility principle and design considerations:

Our main design pattern is based on the Entity-Boundary-Control design pattern.

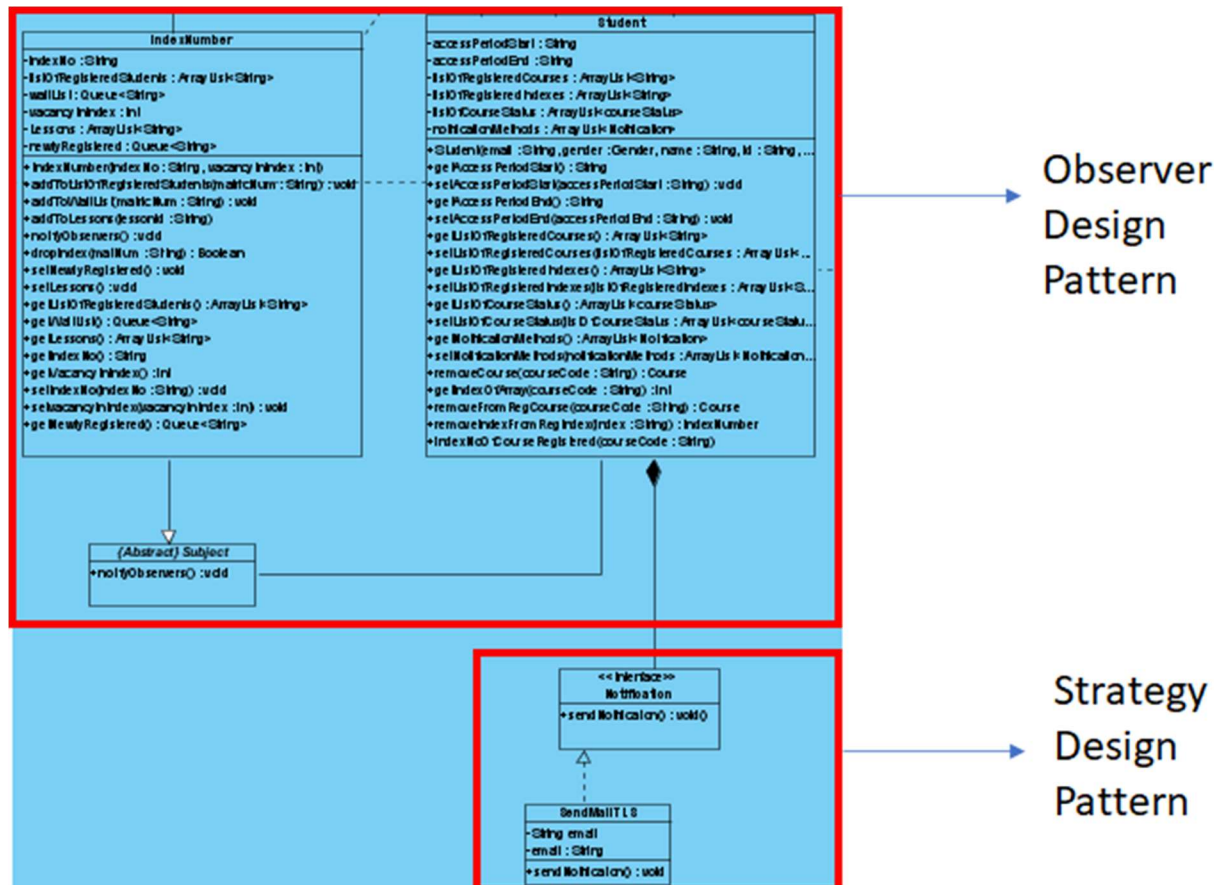
This design pattern allows us as a team to work on different parts of the application at the same time with minimal dependencies and also allows us to separate its classes into more specific classes. For example we have a StudentUI class as the boundary class for the user to interact, a StudentMngr class to control the logic flow of the application and StudentModel as the entity class to store the data. This allows us to reduce the responsibility of each class, hence applying the Single Responsibility Principle.

Open-Closed Principle

We used Strategy Pattern in our Model and Notification class, as we needed them to be interchangeable between different objects that do similar things. For Model class, it stores all the information that is read from the data, and since we have 5 different types of data, we separate them into 5 different models (CourseModel, LessonModel, IndexNumberModel, StudentModel and StaffModel) to store them. With this we can easily extend, replace, or add new subclasses, and allow behavior change at runtime. If in the future we wanted to add another type of data, like a data for professor, we can easily extend a new Model class (e.g. ProfessorModel) from the

Model class. With this implementation, we are able to implement new codes without changing the old.

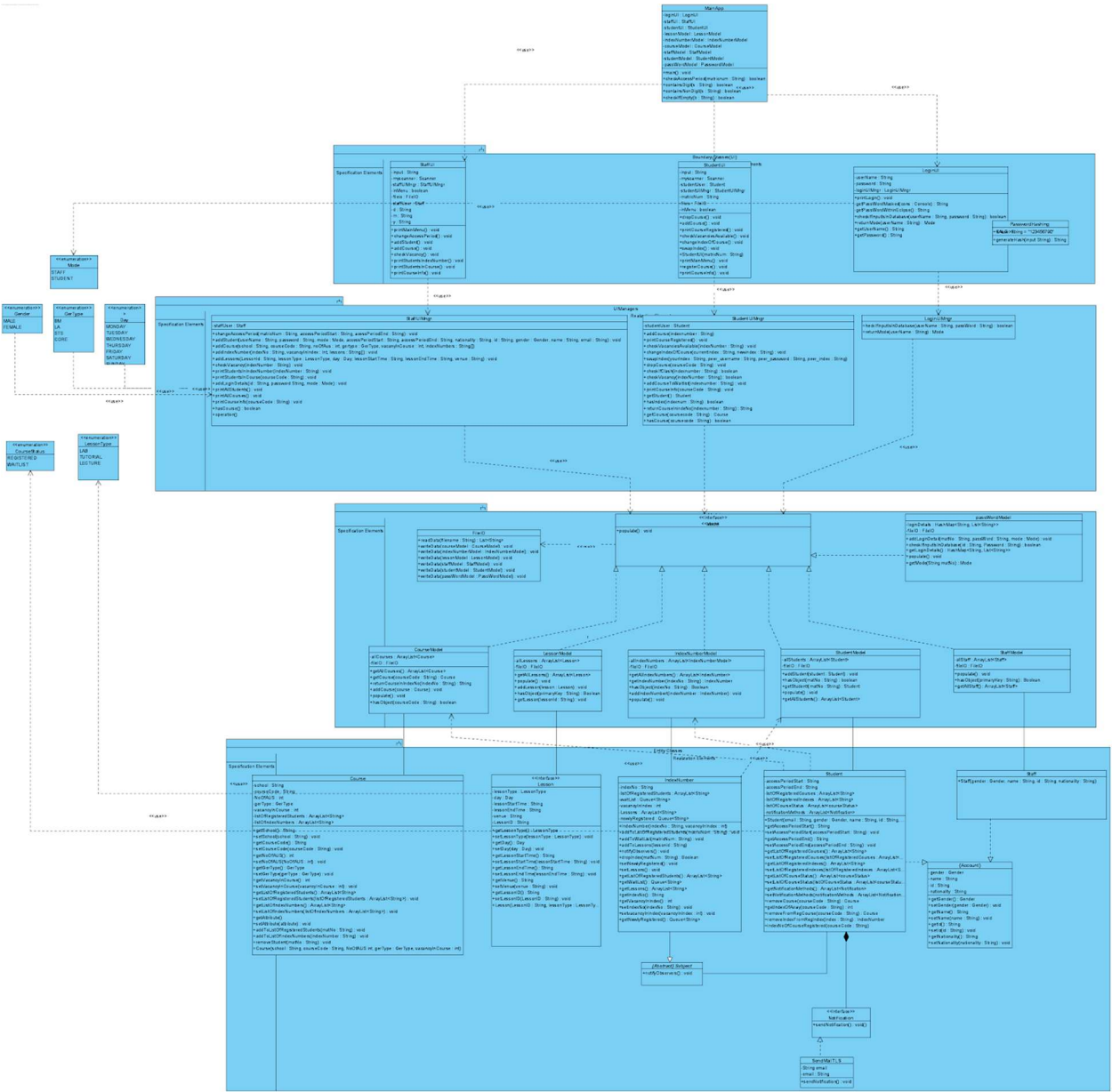
We applied Observer Design Pattern + Strategy Design Pattern for our Notification function, as we want the student to receive an update when he is successfully added to a course without the student having to constantly request to see whether he is still in the Subject class' waitlist.



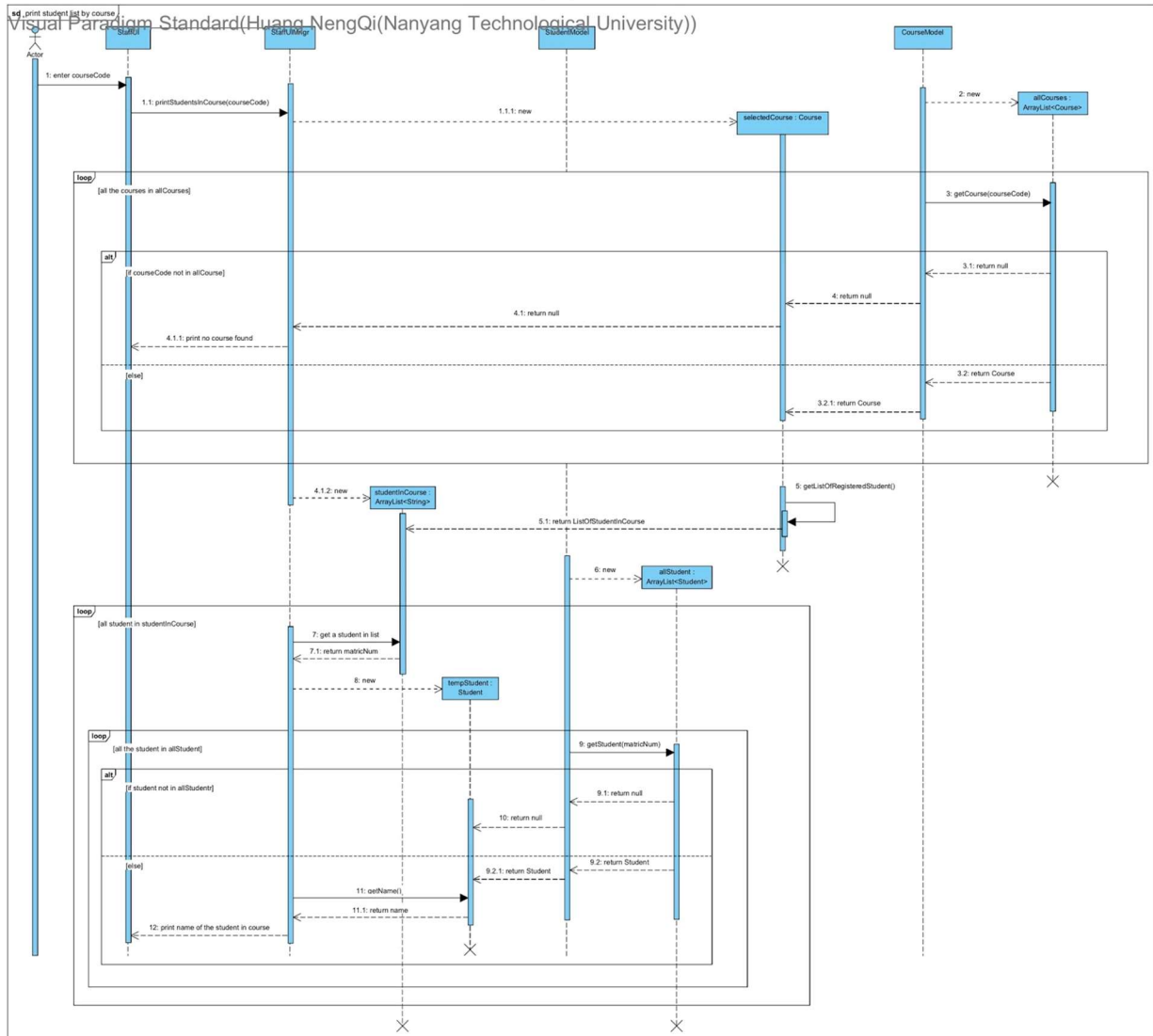
The Notification Interface class uses a strategy pattern design, which has a child class, Email. This makes the program extensible in the future for more notification types like SMS or What'sApp which can just inherit the Notification Interface.

The IndexNumber class inherits the Subject interface and will notify observers with the `notifyObserver()` method once there is a vacancy (a student in `newlyRegList`). For example, when a student drops a course he was registered for, a vacancy opens up and the first student in the wait-list queue will be popped and added to the `newlyRegList`. `NotifyObserver()` will loop through the `newlyRegList`'s students and call each of the notification Objects (e.g. `sendMailTLS`) in the Notification List of the Student. It will then call the `SendNotification()` method in the `Send-MailTLS` class to send an email to every student in the `newlyRegList`. The strategy design pattern on Notification allows extensibility to future types of notification methods. For example, a new SMS notification class can extend the Notification interface and add to the NotificationMethod list from the student. Now the student will also receive an SMS and an Email notification.

UML Class Diagram



UML Sequence Diagram



Testing

Test Case 1: Student Login

Login before allowed period

```
Login Menu
Username: James
Password:
Access Period Start Date : 01/01/2021
Access Period End Date : 01/01/2022
Access period has not opened
```

Login after allowed period

```
Login Menu
Username: Louis
Password:
Access Period Start Date : 01/01/2019
Access Period End Date : 01/10/2020
Access period is over
```

Wrong Password entered

```
Login Menu
Username: James
Password:
Login unsuccessful.
Incorrect Username/Password, please try again.
```

Test Case 2: Add Student

Add Student

```
EnterInput: 2
Enter the following to add student:
Email, Gender, Name, Matric Num, Nationality, Access Period (Start), Access Period (End), Password

Email: Dino@gmail.com
Gender (MALE/FEMALE): MALE
Name (Cannot have digits):Dino
Id (Cannot be empty): Dino
Nationality (Cannot have digits): Singaporean
Access Period format: DD/MM/YYYY
Enter Access Period (Start):
Enter Day: 1
Enter Month: 1
Enter Year: 2020
Enter Access Period (End):
Enter Day: 1
Enter Month: 1
Enter Year: 2021
Password: password
-----All Students-----
Amy Amy
Bob Bob
Charlie Charlie
Daniel Daniel
Edward Edward
Frank Frank
Gandhi Gandhi
Helen Helen
James James
Kevin Kevin
Louis Louis
Max Max
Nathan Nathan
Oscar Oscar
Penelope Penelope
Dino Dino
-----
```


Invalid Data entries

```
EnterInput: 2
Enter the following to add student:
Email, Gender, Name, Matric Num, Nationality, Access Period (Start), Access Period (End), Password

Email: Dinogmail.com
Invalid email!
```

Add Existing Student

```
EnterInput: 2
Enter the following to add student:
Email, Gender, Name, Matric Num, Nationality, Access Period (Start), Access Period (End), Password

Email: Dino@gmail.com
Gender (MALE/FEMALE): MALE
Name (Cannot have digits):Dino
Id (Cannot be empty): Dino
Error! ID already exist.
-----All Students-----
Amy Amy
Bob Bob
Charlie Charlie
Daniel Daniel
Edward Edward
Frank Frank
Gandhi Gandhi
Helen Helen
James James
Kevin Kevin
Louis Louis
Max Max
Nathan Nathan
Oscar Oscar
Penelope Penelope
Dino Dino
-----
```

Test Case 3: Add Course

Add existing course

```
EnterInput: 3
Enter the following to add course:
School, Course Code, Number of AU, GerType, Number of Indexes

School: SCSE
Course Code: CZ2001
Sorry, course code already exist, please enter another course code!
```

Add new course

```
EnterInput: 3
Enter the following to add course:
School, Course Code, Number of AU, GerType, Number of Indexes

School: SCSE
Course Code: CZ2006
Number of AU:3
GerType (CORE/BM/LA/STS): BM
Number of Indexes: 1
Enter Index Number: 10400
-----Enter Index Number 10400 details-----
Enter number of Vacancy in Index:10
Enter number of Lessons in Index:1
-----Lesson 1-----
Enter Lesson ID:COURSE1
Enter type of Lesson (LECTURE/LAB/TUTORIAL):LAB
Day (MONDAY/TUESDAY...):TUESDAY
Lesson Start Time:1000
Lesson End Time:1400
Enter Venue:HWLAB3
-----
-----All Courses-----
CZ2001 SCSE
CZ2002 SCSE
CZ2003 SCSE
CZ2004 SCSE
CZ2005 SCSE
CZ2006 SCSE
-----
```


Invalid Data Entries

```
EnterInput: 3
Enter the following to add course:
School, Course Code, Number of AU, GerType, Number of Indexes

School: SCSE
Course Code:
Empty/invalid input!
```

Example: Course code cannot be empty

Test Case 4: Register student for course

Add student to course index with available vacancies

```
EnterInput: 1
Register Course: Enter Index Number
10480
CZ2001 10480 Available vacancies:1
This index has available vacancies!
1. Register course
2. Back to Main Menu
Enter input:1
Course added successfully!
```

Add student to course index with 0 vacancies in Tutorial/Lab

```
EnterInput: 1
Register Course: Enter Index Number
10485
CZ2003 10485 Available vacancies:0
This index has no available vacancies
1. Continue to register (will be placed in waitlist)
2. Back to Main Menu
Enter input:1
You are being put on the waitlist! A notification will be sent to you if you are added to the course.
```

```
EnterInput: 3
-----
CZ2001 CORE 10480 REGISTERED

TYPE/VENUE/DAY/StartTime/EndTime
LECTURE LT2 MONDAY 1000 1200
TUTORIAL TR17 TUESDAY 1400 1600
-----
CZ2003 CORE 10485 WAITLIST

TYPE/VENUE/DAY/StartTime/EndTime
TUTORIAL TR19 THURSDAY 1600 1800
-----
Student Name
```

Registering for the same course again

For James, where he has been registered for CZ2001 already.

```
EnterInput: 1
Register Course: Enter Index Number
10480
Unable to register: Course already registered or in waitlist.
```

Invalid Data Entries

```
EnterInput: 1
Register Course: Enter Index Number
12345
Sorry! This Index Number is not found!
```

Test Case 5: Check available slot in class

Check for vacancy in course index

```
EnterInput: 4
Index number of course you wish to check vacancies for:10480
Number of vacancies in 10480: 1/3[vacancy/total size]
Number of people in waitlist: 0
```

Invalid data entries

Index 12345 do not exist

```
EnterInput: 4
Index number of course you wish to check vacancies for:12345
Error: Index number not found.
```

Test Case 6: Day/Time clashes with other courses

Day/Time clashes with other courses

```
EnterInput: 1
Register Course: Enter Index Number
10488
Unable to register: There are clashes in timeslots.
```

Test Case 7: Waitlist Notification

Waitlist Notification, James on waitlist for CZ2003

```
EnterInput: 3
-----
CZ2003  CORE    10485  WAITLIST

TYPE/VENUE/DAY/StartTime/EndTime
TUTORIAL      TR19    THURSDAY    1600    1800
-----
CZ2001  CORE    10480  REGISTERED


TYPE/VENUE/DAY/StartTime/EndTime
LECTURE LT2    MONDAY  1000    1200
TUTORIAL      TR17    TUESDAY 1400    1600
```

Max has CZ2003 registered and proceeds to drop

```
EnterInput: 2
-----
CZ2005 CORE 10488 WAITLIST
TYPE/VENUE/DAY/StartTime/EndTime
LECTURE LT2 MONDAY 1100 1300
-----
CZ2004 CORE 10487 REGISTERED
TYPE/VENUE/DAY/StartTime/EndTime
LAB HPL TUESDAY 1400 1600
-----
CZ2003 CORE 10485 REGISTERED
TYPE/VENUE/DAY/StartTime/EndTime
TUTORIAL TR19 THURSDAY 1600 1800
-----
Drop course: Enter the Course code
CZ2003
Done
Course successfully dropped.
```

James will receive the email from system to notify him

Testing Subject

 sweesing1@gmail.com <sweesing1@gmail.com>
4:25 PM

To: #PEH SWEE SING#

You Got Your Course!!!

James now have the course registered

```
EnterInput: 3
-----
CZ2003 CORE 10485 REGISTERED
TYPE/VENUE/DAY/StartTime/EndTime
TUTORIAL TR19 THURSDAY 1600 1800
-----
CZ2001 CORE 10480 REGISTERED
TYPE/VENUE/DAY/StartTime/EndTime
LECTURE LT2 MONDAY 1000 1200
TUTORIAL TR17 TUESDAY 1400 1600
-----
```

Test Case 8: Print student list by index number, course

Print list by index

```
EnterInput: 5
Enter Index Number: 10480
Matric numbers of students in 10480:
Amy
Charlie
```

Print list by course

```
EnterInput: 6
Enter Course Code:
CZ2001
Students in CZ2001:
Amy FEMALE Singaporean
Bob MALE Singaporean
Charlie MALE Singaporean
Daniel MALE Singaporean
Nathan MALE Singaporean
```

Invalid Data Entries

For 5, when invalid index number entered.

```
EnterInput: 5
Enter Index Number: 12345
Error: Index number not found.
```

For 6, when invalid course code entered.

```
EnterInput: 6
Enter Course Code:
CZ1234
Error: Course Code not found.
```

Test Case 9: Drop Course

When a student drop a registered course and waitlist is empty, Example: James

```
EnterInput: 2
-----
CZ2001 CORE 10480 REGISTERED
TYPE/VENUE/DAY/StartTime/EndTime
LECTURE LT2 MONDAY 1000 1200
TUTORIAL TR17 TUESDAY 1400 1600
-----
CZ2003 CORE 10485 WAITLIST
TYPE/VENUE/DAY/StartTime/EndTime
TUTORIAL TR19 THURSDAY 1600 1800
```

when James drops CZ2001

```
EnterInput: 3
-----
CZ2003 CORE 10485 WAITLIST
TYPE/VENUE/DAY/StartTime/EndTime
TUTORIAL TR19 THURSDAY 1600 1800
```

When James drops the course that is on waitlist

```
EnterInput: 3
There are no courses registered.
```

Test Case 10: Change Access Period

Staff updates James access period

```
EnterInput: 1
Change Access Period:
Enter Matric Number of student: James
Access Period format: DD/MM/YYYY
Enter NEW Access Period (Start):
Enter Day:1
Enter Month:1
Enter Year:2020
Enter NEW Access Period (End):
Enter Day:1
Enter Month:1
Enter Year:2021
Access Period successfully changed!
```

```
Login Menu
Username: James
Password:
Access Period Start Date : 01/01/2020
Access Period End Date : 01/01/2021
Login successfully!
```

(Previously James was not able to login)

Test Case 11: Change Index (Student)

Same course but no vacancy

```
EnterInput: 5
Enter CURRENT Index Number:10480
Enter NEW Index Number:10481
Error: New index does not have enough vacancies!
```

Different course (regardless of vacancy)

```
EnterInput: 5
Enter CURRENT Index Number:10480
Enter NEW Index Number:10482
Error: Current index and New index do not share the same course code
```

Successful change

```
EnterInput: 5
Enter CURRENT Index Number:10480
Enter NEW Index Number:10481
Index number is changed successfully! Press 3 to check registered courses!
```

```
EnterInput: 3
-----
CZ2003 CORE 10485 WAITLIST
TYPE/VENUE/DAY/StartTime/EndTime
TUTORIAL TR19 THURSDAY 1600 1800
-----
CZ2001 CORE 10481 REGISTERED
TYPE/VENUE/DAY/StartTime/EndTime
LECTURE LT2A THURSDAY 1000 1200
LAB SPL FRIDAY 1300 1500
```

Test Case 12: Swap Index with peer

Invalid Index number input

```
EnterInput: 6
Enter your Index Number:
abc
Invalid input!
```

Wrong username and wrong password:

```
EnterInput: 6
Enter your Index Number:
10480
Enter Peer's username:
abcd
Enter Peer's password:
password
Peer's index:
10481
Username does not exist.
```

```
EnterInput: 6
Enter your Index Number:
10480
Enter Peer's username:
Amy
Enter Peer's password:
1234
Peer's index:
10481
Peer's password is not correct
```

All valid, but index are from different courses

```
EnterInput: 6
Enter your Index Number:
10480
Enter Peer's username:
Amy
Enter Peer's password:
password
Peer's index:
10483
Error: Your index and peer index do not share the same course code
```

Valid inputs (Swap successful)

```
EnterInput: 6
Enter your Index Number:
10480
Enter Peer's username:
Amy
Enter Peer's password:
password
Peer's index:
10481
Swap Success!
```

James:

```
EnterInput: 3
-----
CZ2003  CORE    10485  REGISTERED
TYPE/VENUE/DAY/StartTime/EndTime
TUTORIAL      TR19   THURSDAY    1600    1800
-----
CZ2001  CORE    10481  REGISTERED
TYPE/VENUE/DAY/StartTime/EndTime
LECTURE LT2A   THURSDAY    1000    1200
LAB      SPL     FRIDAY    1300    1500
```