

NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE

NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

CZ2002 OBJ ORIENTED DES & PROG, AY 20/21 SEMESTER 1

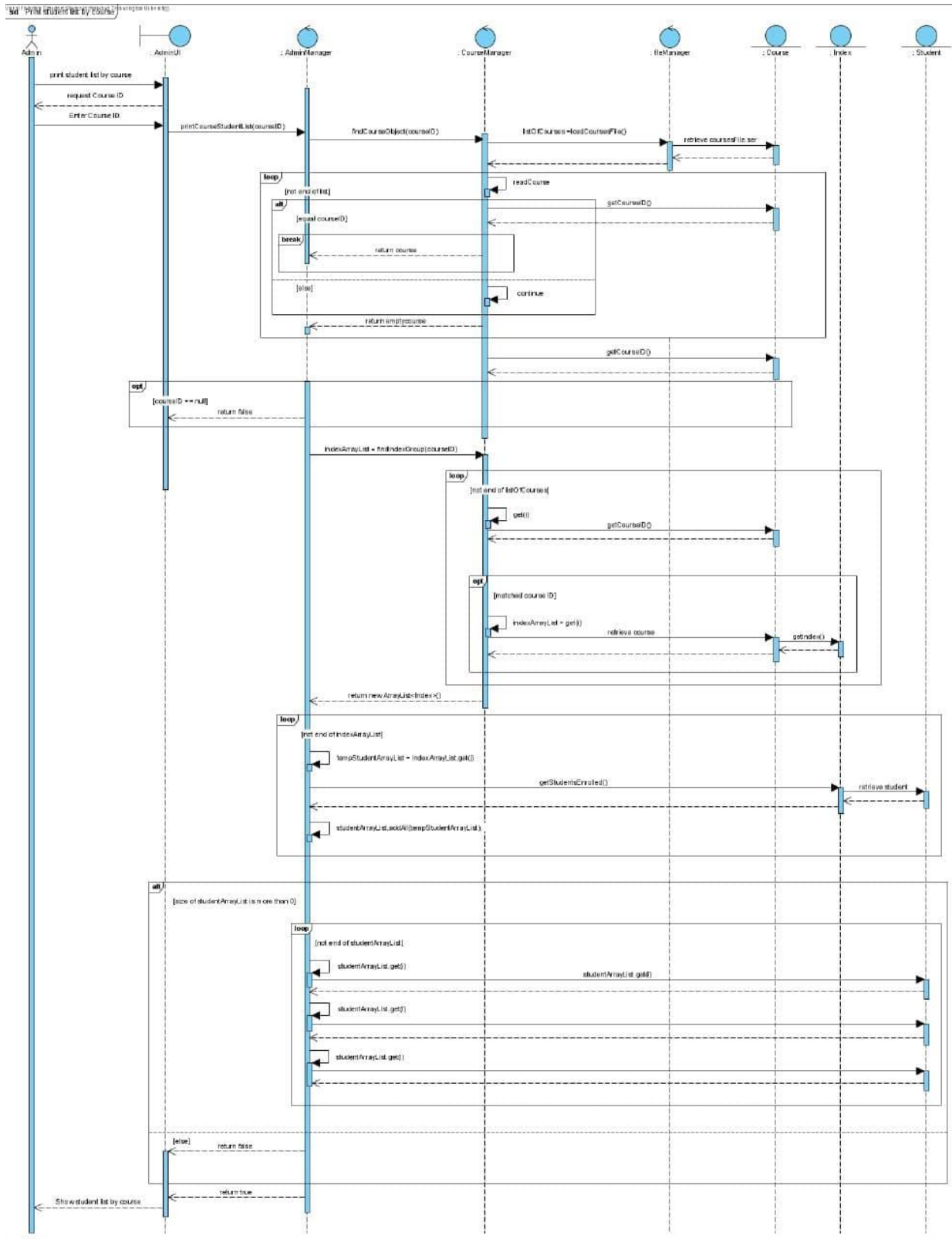
Lab Group: SS12

Group Members:

LIM BING HONG, JASPER	(U1922738B)
QUAH DIAN WEI	(U1920106C)
RAYMOND GOH KANG SHENG	(U1921489J)
TONY HO MAN TUNG	(U1922328C)
WONG CHIN HAO	(U1921712L)

Date of Submission: 25th November 2020

A detailed UML Sequence Diagram ((Refer to image in submission folder for details))



Design Considerations

To encapsulate the data in each entity, we created manager objects for the corresponding entities. This is so that the only way to access and change entities would be through these corresponding entity managers (Student manager, Course manager). File manager was created to handle the saving and loading of the multiple files generated. While a loginController handles the validation of the User's information. Through these design considerations, we ensure that there is no class that can overstep their boundaries so that they will not affect other classes to ensure that the program is modular.

To take into account real world behaviour where users may exit once making changes, we ensured that after each change to any entity records, that change is saved. Hence the programme is dynamic in the sense that it maintains the changes made to it during run time and saves the changes upon exiting the programme.

Don't Repeat Yourself (DRY) Principle

This principle simply states that we do not repeat duplicate things in our code. There are multiple times where we applied this principle. For example, by applying abstraction, we have admin class and student class inherit from the user class. Another example is making sure we put similar functionality into a method so that those who need it can simply call it without re-coding it again and again. In addition, this is further illustrated in our application by making use of inheritance to allow us to reuse methods and fields of the parent class from the child class.

Single Responsibility Principle

For our program, we make sure that each class is only responsible for only one responsible only. Our class is designed to focus on only a single problem, piece of logic, or a single domain. For example, we separate our user interface to the admin's user interface or student's user interface so each user interface is only responsible for one domain only. we further divide our code to the boundary which is responsible for the handling of user interactions, the control which is responsible for the implementation of our logic and the entity which is responsible to contain the objects.

Future Consideration

Given that we will have more time in the project to improve our work, the group will try to enhance our project by making sure that we fulfill the SOLID principle that was taught to us in this module. We can also make use of the design pattern that was taught in CZ2006 to further enhance our project.

One such example would be for the sending of notifications. A combination of observer and strategy design pattern could have been used. Student's would have been the observers and StudentManager the subject. The different strategies would be Email, Whatsapp, SMS for example. This design would also satisfy the Open-Closed principle, we would have a notification class that would be closed for modification and open to new types of communication implementation.

Test Cases for MySTARS Application

1. Student Login

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Login before allowed period (dates)	startTime: 2020/11/30 EndTime: 2020/12/30 Current Date: 2020/11/22	Appropriate message display	"Invalid access period"	Pass
b	Login after allowed period (dates)	startTime: 2020/11/20 EndTime: 2020/11/21 Current Date: 2020/11/22	Appropriate message display	"Invalid access period"	Pass
c	Wrong password	Login ID:student Login PW: password1	Appropriate error message display	"Wrong login information"	Pass
d	Password	Login ID:student	Password will	Password is	Pass

	masking in console	Login PW: password1	be hidden when typing	hidden when typing	
--	--------------------	---------------------	-----------------------	--------------------	--

2. Add a Student

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Add a new student	Enter Student Information according to UI	the listing of all students should be displayed after the addition	Print a list of existing student + new student	Pass
b	Add an existing student	New student with same matriculation number	Appropriate error message display	“Student Exists. Returning to the main UI.”	Pass
c	Invalid data entries	Input yearOfStudy as String instead of integers	Appropriate error message display	“Invalid input. Please input integers.”	Pass

3. Add a Course

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Add a new course	Enter Course Information according to UI	listing of all courses should be displayed after the addition.	“Course record created. Returning to the main UI...” listing of all	Pass

				courses is displayed after the addition.	
b	Add an existing course	New course with same course id	Appropriate error message display	“Course Exists. Returning to the main UI...”	Pass
c	Invalid data entries	Input courseAu as String instead of integers	Appropriate error message display	“Invalid input. Please input integers.”	Pass

4. Add a Index

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Add a new Index	Enter Index Information according to UI	Index record is created.	“Index record created. Returning to the main UI....”	Pass
b	Add an existing index	New index with same index id	Appropriate error message display	"Index exists. Returning to the main UI....”	Pass
c	Add an index with non-existence course	Input courseAu as String instead of integers	Appropriate error message display	“Course ID not found. Returning to the main UI....”	Pass

5. Register student for a course

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Add a student to a course index with available vacancies.	Course ID: CZ2001 Index ID: 10125	Appropriate message display	“Adding Course ID : CZ2001 and Index ID : 10125 Successfully added student to course.”	Pass
b	Add a student to a course index with 0 vacancies in Tut / Lab.	Course ID : cz2004 Index Id : 10142	Appropriate message display	“Adding Course ID : CZ2004 and Index ID : 10142 Successfully added student to waitlist!”	Pass
c	Register the same course again	Course ID : cz2001 Index ID : 10125	Appropriate error message display	Adding Course ID : CZ2001 and Index ID : 10125 Already enrolled in this course!	Pass
d	Invalid data entries (eg wrong student ID / course code, etc)	Course ID : CZ5005 Index ID : 100505	Appropriate error message display	Course not found! Course does not exist!	Pass

6. Check available slot in a class (vacancy in a class)

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Check for vacancy in course index	Course ID : cz2001 Index ID : 10125	Appropriate message display, eg 3/10 [vacancy/total size]	“Vacancies : 4 Slots” (Total vacancies per index : 5)	Pass
b	Invalid data entries (eg course code, class code etc)	Course ID : FJDSIAF Index ID : hi	Appropriate error message display	“Course does not exist!”	Pass

7. Day/Time clash with other course

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Add a student to a course index with available vacancies.	Example of clashing courses, Student must have either one enrolled: Course ID : CZ2004 Index ID : 10142 CZ2003 Index ID : 10129	Appropriate message display, eg day/time clash	“Timetable clashes”	Pass

8. Waitlist notification

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Add studentA to a course index with 0 vacancies	Course ID : cz2004 Index ID : 10142	Appropriate message display to inform student on waitlist	“Adding Course ID : CZ2004 and Index ID : 10142 Successfully added student to waitlist!”	Pass
b	Drop studentB from the same course index	StudentB dropped CZ2004	StudentB successfully dropped and studentA successfully added. Simulate a notification sent	StudentB successfully dropped and studentA successfully added. An email notification sent to studentA successfully.	Pass
c	Display studentA timetable	Option “7” from StudentUI	Show the course on waitlist added.	*A very nicely compiled timetable was shown* (Please refer to our also very nicely compiled video)	Pass

9. Print student list by index number, course

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Print list by (i) Course	Course ID : CZ2001	Appropriate display	“List of students in the course CZ2001: “ *A table with student name, gender and nationality shown*”	Pass
b	Print list by (ii) index	Course ID : CZ2001 Index ID : 10125	Appropriate display	“List of students in the course CZ2001 of Index Group 10125: “ *A table with student name, gender and nationality shown*”	Pass
c	Invalid data entries (i) course code	Course ID : nihao Index ID : hao	Appropriate error message display	“Invalid CourseID or IndexID. Returning to the main UI....”	Pass
d	Invalid data entries (ii) index	Course ID : CZ2001 Index ID : hao	Appropriate error message display	“Invalid CourseID or IndexID. Returning to the main UI....”	Pass

10. Edit Student Access Period

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Edit Student Access Period	startTime: 2020/11/01 endTime: 2020/11/30	Access Period Updated	"Student Access Period Updated. Returning to the main UI...."	Pass
b	Invalid Access Period	startTime: nihao endTime: nihao	Appropriate error message display	"Invalid date format. Returning to the main UI...."	Pass

11. Update Existing Course

#	Test Cases Description	Input	Output		Status
			Expected	Actual	
a	Update Course ID	Update Course ID according to UI	Appropriate display	"Course's ID updated. Returning to the main UI...."	Pass
b	Update Course School	Update Course School according to UI	Appropriate display	"Course's School updated. Returning to the main UI...."	Pass
c	Update Index Number	Update index number according to UI	Appropriate display	"Index's ID updated. Returning to the main UI...."	Pass
d	Update vacancy	Update vacancy according to UI	Appropriate display	"Index's vacancy updated. Returning	Pass

				to the main UI....”	
e	Update an non-existence course	Course ID: nihao	Appropriate error message display	“Course ID does not exist. Returning to the main UI...”	Pass
f	Update course code with existing course code	Course ID: CZ2001 New Course ID: CZ2002	Appropriate error message display	“New Course's ID exists. Returning to the main UI...”	Pass
g	Update Index Number with existing index number	Course ID: CZ2001 New Index ID: 10125	Appropriate error message display	“index ID exists! Returning to the main UI...”	Pass

12. Change Index of a Course

#	Test Case Description	Input	Output		Status
			Expected	Actual	
a	Change Index of a Course	Course ID: CZ2001 New Index ID: 10125	Appropriate display	“Index has been changed”	Pass
b	Change Index of a Course that Student is not taking	Course ID: CZ2007	Appropriate error message display	“Invalid Course Entered!”	Pass

c	Change Index of a Course to an Index that will clash with Student's timetable	Course ID: CZ2001 New Index ID: 10900	Appropriate error message display	"Clash in timetable, change is not possible"	Pass
---	---	--	-----------------------------------	--	------

13. Swap Index of a Course

#	Test Case Description	Input	Output		Status
			Expected	Actual	
a	Swap Index of a Course with another Student	Course ID: CZ2001 Name of another Student: Cena Mareatte	Appropriate display	"Index swapped"	Pass
b	Swap Index of a Course that Student is not taking	Course ID: CZ2007	Appropriate error message display	"Invalid Course Entered!"	Pass
c	Swap Index of a Course with another Student who does not take the same course entered	Course ID: CZ2001 Name of another Student: Cena Mareatte	Appropriate error message display	"Student is not taking this course!"	Pass

d	Swap Index of a Course with another Student whose Index will clash with the current Student	Course ID: CZ2002 Name of another Student: Cena Mareatte	Appropriate error message display	“Clash in timetable, change is not possible”	Pass
---	---	--	-----------------------------------	--	------

Youtube demo link : <https://www.youtube.com/watch?v=DAUpCGZBEqY>

Declaration of Original Work






Attached a scanned copy with the report with the filled details and signatures.

Declaration of Original Work for CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course (CE2002 or CZ2002)	Lab Group	Signature /Date
Lim Bing Hong, Jasper	CZ2002	SS12	 / 221120
Quah Dian Wei	CZ2002	SS12	 / 221120
Raymond Goh Kang Sheng	CZ2002	SS12	 / 221120
Tony Ho Man Tung	CZ2002	SS12	 / 221120
Wong Chin Hao	CZ2002	SS12	 / 221120

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.