CS3343 Software Engineering Practice

BSC4 Project Group 6

Analysis and Design Report

Group Member:

CHAN Ho Man 53078369

CHEUK Yik Sum 53012319

CHAN Ho Man 53091240

LEE Man To 53075299

HO Sui Cheong Jonathan 53117178

CHEUNG Chi Ngai 53013494

Tutorial Section: LB2

Last Edited by Cheuk Yik Sum on 25th November, 2014

# **Contents**

[**Contents** 2](#_Toc404727755)

[**Introduction** 3](#_Toc404727756)

[Design 4](#_Toc404727757)

[Class Description 4](#_Toc404727758)

[Work Flow 5](#_Toc404727759)

[Class Diagram 6](#_Toc404727760)

[Sequence Diagram 7](#_Toc404727761)

[Terms 7](#_Toc404727762)

[Conflict Number 7](#_Toc404727763)

[Priority 9](#_Toc404727764)

[Our algorithm 10](#_Toc404727765)

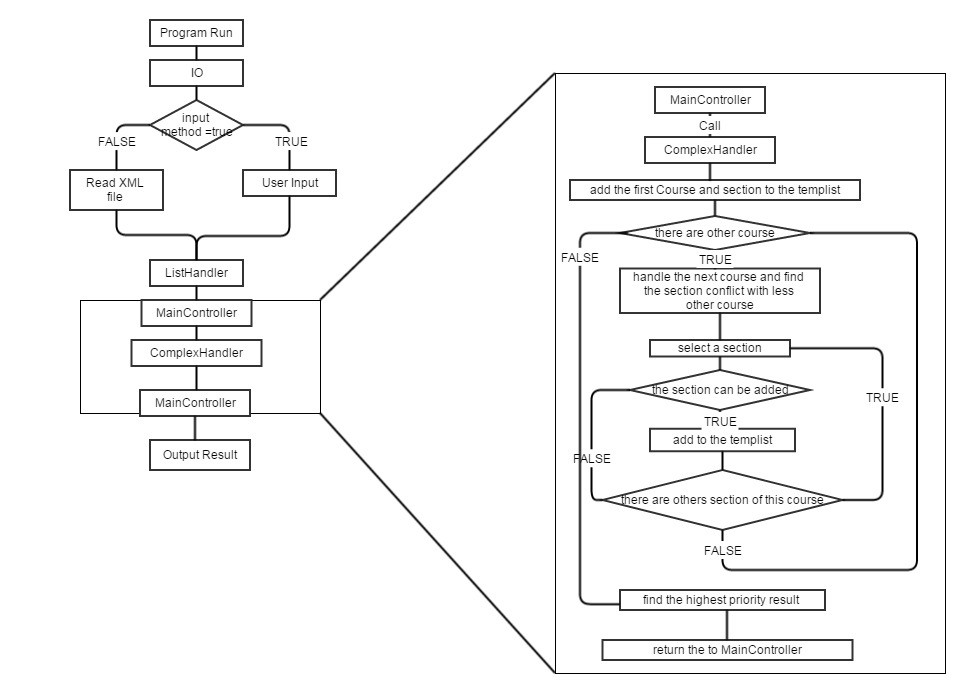
# **Introduction**

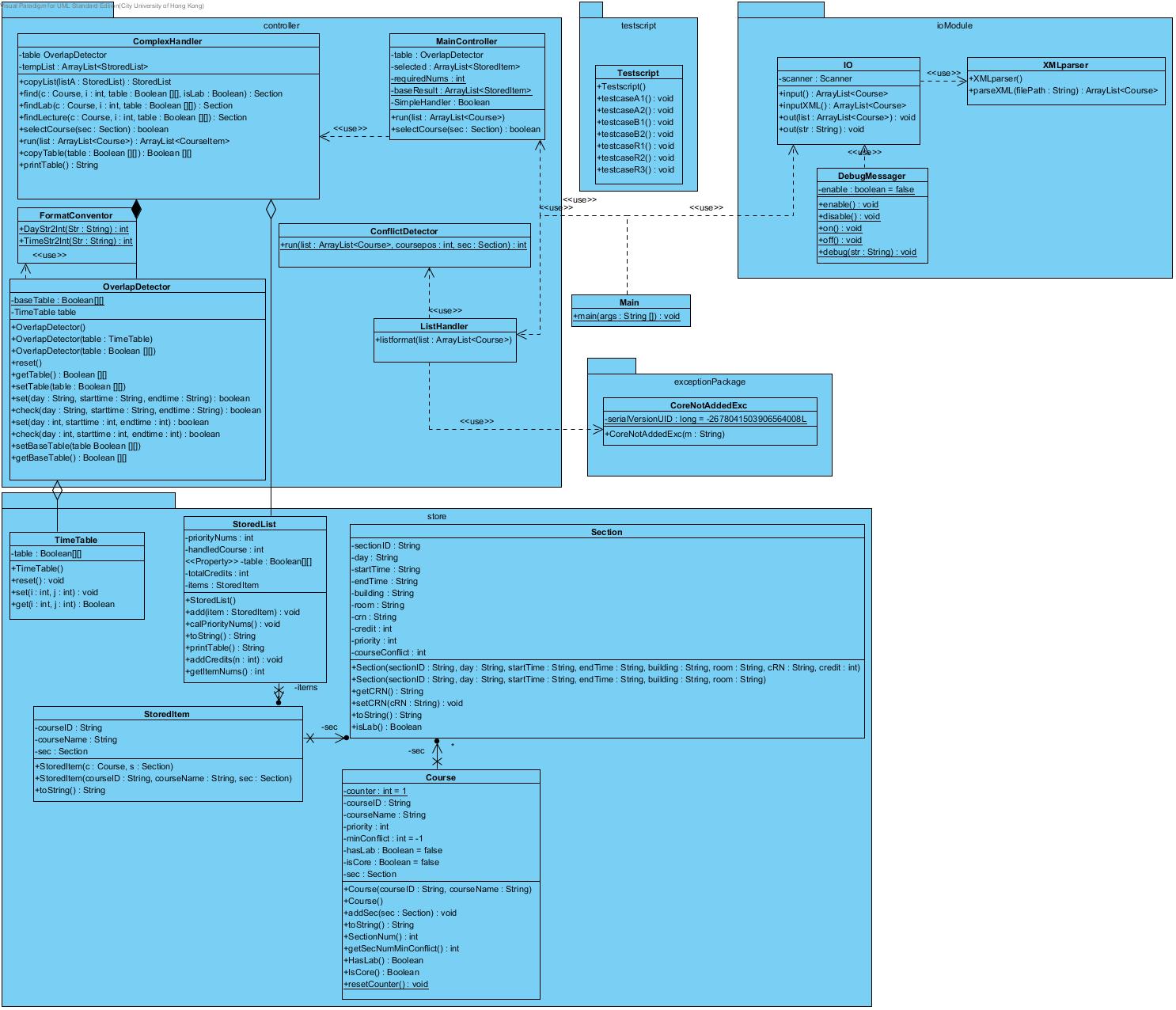
Design

Class Description

In the following, we will introduce our project design. The following list all the classes of our project that we have developed.

|  |  |
| --- | --- |
| Class | Brief introduction |
| **Package controller** | |
| ComplexHandler | The controller used to actually find the result of the input |
| ConflictDetector | A controller helps to find out all the conflict will happen |
| FormatConventor | A controller helps to convert some String(such as day, hour) to a integer value for other controller to use |
| ListHandler | A controller that helps to add more data that does not in the input to the course and section which can help the ComplexHandler find the result |
| MainController | The first controller will be run, it will call ComplexHandler and get the result for further action |
| OverlapDetector | A important controller helps to check can a section be selected for some situation |
| **Package exceptionPackage** | |
| CoreNotAddedExc | Throwed when a core course is not possible to be add. |
| **Package ioMudule** |  |
| DebugMessager | Used to output the debug message, can turn on or off |
| IO | Used to read the simple input and write the simple output |
| XMLparser | Used to read the XML file as a input |
| **Package store** | |
| Course | Provided all the accessor and mutator methods for the course object, and have method to find the smaller conflict number and find the number of section that conflict number equal to the minimum number of the course. |
| Section | Provided all the accessor and mutator methods for the section object |
| StoredItem | Provided method to get the courseID, courseName and courseSection for a particular course, and genetate these details in the output. |
| StoredList | Used arraylist to put all the course details |
| TimeTable | Used arraylist to allocate a course into a specific time slot |

Work Flow

Class Diagram

Design Pattern Used:

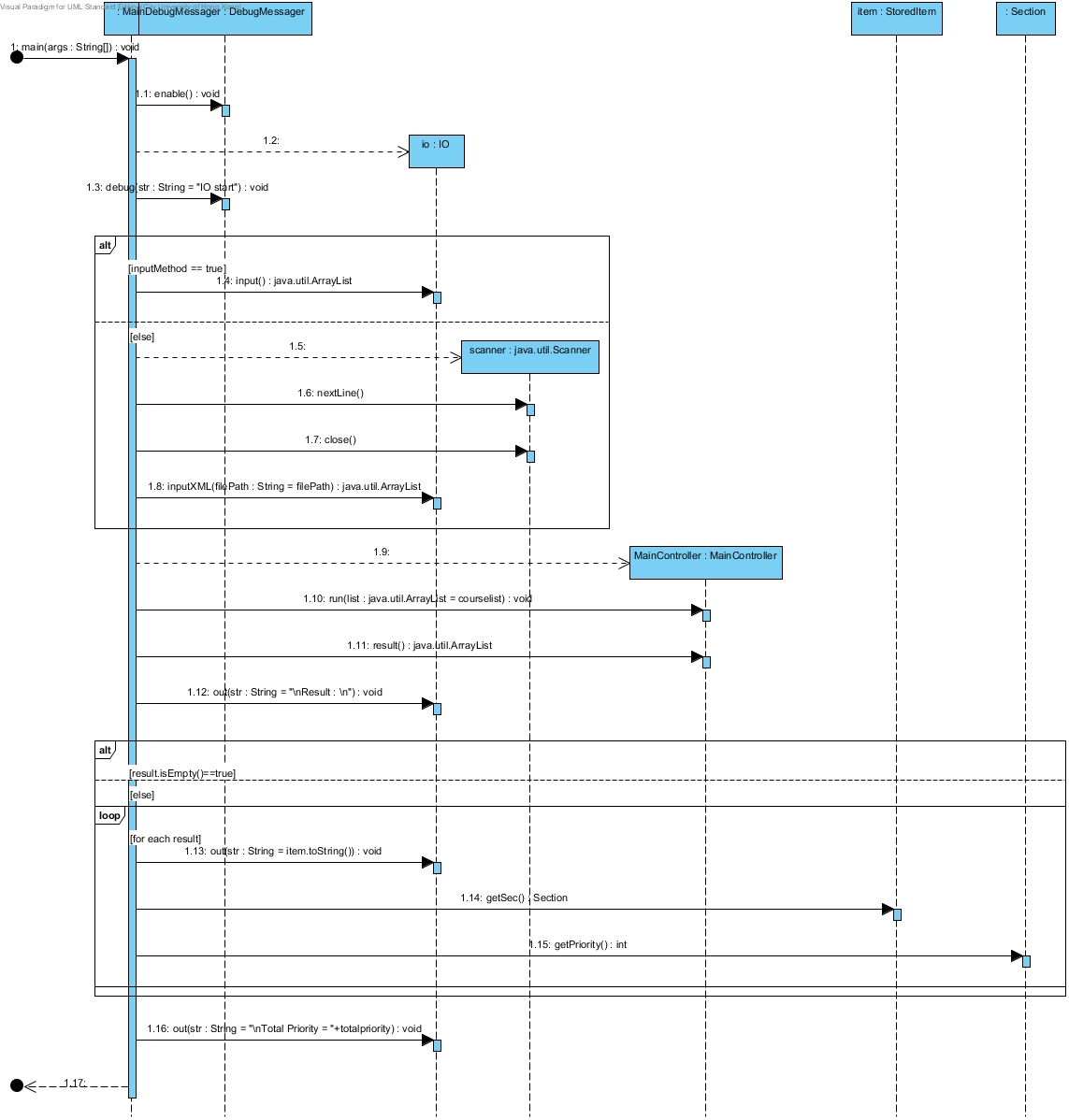
1. Façade Pattern:

We used different types of controller as interface to handle all of the courses. The controller object will handle those complicate operations in these interfaces.

2. Strategy Pattern

The Main Controller will consider different situation to decide use which controller to handle the course, such as OverlapDetector and ConflictDetector in our program.

Sequence Diagram



Terms

Conflict Number

Conflict Number is a number used to representing the effect of selecting a section of a course. This value will be calculated when the ListHandler run.

This value is the number of course that could not be selected LATER. if there are at least one session conflict with the current session, this value will +1, no matter how many session are conflicted, this value will only +1, as this can mark the calculation more easier.

**Example**

CS0001 TEST1 5  
C01 Fri 1200 1400 AC1 LT6 10001 3  
C02 Mon 1100 1300 AC1 LT8 10002 3   
C03 Wed 2000 2200 AC1 LT7 10003 3   
C04 Mon 0900 1100 AC1 LT3 10004 3   
C05 Fri 1500 1700 AC1 LT2 10005 3  
  
CS0002 TEST2 5  
C01 Fri 1200 1300 AC1 LT6 20001 3  
C02 Mon 1200 1300 AC1 LT6 20002 3  
C03 Tue 1200 1300 AC1 LT6 20003 3  
C04 Wed 1200 1300 AC1 LT6 20004 3  
C05 Thu 1200 1300 AC1 LT6 20005 3  
  
CS0003 TEST3 5  
CA1 Fri 1200 1400 AC1 LT6 30001 3  
CA2 Mon 1100 1300 AC1 LT8 30002 3   
CA3 Wed 1800 2000 AC1 LT7 30003 3   
CA4 Mon 0900 1100 AC1 LT3 30004 3   
CA5 Fri 1500 1700 AC1 LT2 30005 3  
  
CS0004 TEST4 1  
C01 Tue 0900 1200 AC2 2550 40001 3  
  
CS0005 TEST5 3  
C01 Wed 0900 1300 AC2 2550 50001 3  
C01 Mon 1700 1900 AC2 2550 50001 3  
C01 Mon 1500 1700 AC2 2550 50001 3  
  
CS0006 TEST6 1  
C01 Fri 1200 1400 AC2 2550 60001 3

* For CS0001 - C01, it conflict Number = 3 (conflicted with CS0002 - C01, CS0003 - C01 & CS0006-C01) because **if current section conflict with any session of a single course, it Confilct number will +1**
* For CS0002 - C01, it conflict Number = 2 (conflicted with CS0003 - C01 & CS0006-C01) because **the conflict number ONLY consider the course that having lower priority than the current course.**

Priority

Priority is used to determine the satisfy result. For each Sessions, their priority will be calculate as follow:

1. Get the ordered course list and session list from the user input
2. For the Courses, starting at the bottom number it from 1 to n.
3. Find the largest number of the sessions of all the Courses, calculate the digit number needed to represent the priority sessions.
4. For the Sections, starting at the top and number it from 1 to n.
5. Combine the number of course and Session.

**Example**

1. Get the course list and Numbering the Course list, Find largest Session Number of all the Course

CS0001 TEST1 120 (Priority Number = 2)  
C01 Fri 1200 1400 AC1 LT6 10001 3  
C02 Mon 1100 1300 AC1 LT8 10002 3   
...  
C120 Sat 1100 1300 AC1 LT8 10102 3   
  
CS0002 TEST1 2 (Priority Number = 1)  
C01 Fri 1500 1700 AC1 LT6 20001 3  
C02 Mon 1600 1800 AC1 LT8 20002 3   
  
Max(Session Number) = 120  
Digit needed = 3

1. Numbering the Session

CS0001 TEST1 120 (Priority Number = 2)  
C01 Fri 1200 1400 AC1 LT6 10001 3 (Priority Number = 001)  
C02 Mon 1100 1300 AC1 LT8 10002 3 (Priority Number = 002)  
...  
C120 Sat 1100 1300 AC1 LT8 10102 3 (Priority Number = 120)  
  
CS0002 TEST1 2 (Priority Number = 1)  
C01 Fri 1500 1700 AC1 LT6 20001 3 (Priority Number = 001)  
C02 Mon 1600 1800 AC1 LT8 20002 3 (Priority Number = 002)  
  
Max(Session Number) = 120  
Digit needed = 3

1. Combine the number

CS0001 TEST1 120 (Priority Number = 2)  
C01 Fri 1200 1400 AC1 LT6 10001 3 (Priority Number = 001, Final Priority Number = 2001)  
C02 Mon 1100 1300 AC1 LT8 10002 3 (Priority Number = 002, Final Priority Number = 2002)  
...  
C120 Sat 1100 1300 AC1 LT8 10102 3 (Priority Number = 120, Final Priority Number = 2120)  
  
CS0002 TEST1 2 (Priority Number = 1)  
C01 Fri 1500 1700 AC1 LT6 20001 3 (Priority Number = 001, Final Priority Number = 1001)  
C02 Mon 1600 1800 AC1 LT8 20002 3 (Priority Number = 002, Final Priority Number = 1002)  
  
Max(Session Number) = 120  
Digit needed = 3

Our algorithm

All of the main logic are implement in the ComplexHandler.

The Step to find out the result is as follow.

* 1. For the first course, copy all the section into the tempList (storing the possible results). At this stage we don’t consider the conflict number as we want to try to achieve a result will highest priority.
  2. For all of the possible result in the tempList, we try to add more course into the result. We will choose the next course and consider the section with the lowest conflict number only, this can help the program to find out the result more quickly and effectively.
  3. For the Section satisfy the condition for step 2 mentioned, if the OverlapDetector report that the timeslot for this section are available, this section will be selected and add into the tempList.
  4. Repeat Step 2 and 3, until there are no more Course and all of the result in the tempList had been handled. Now we got a list of all the possible results.
  5. Access all of the results, and find the FIRST course with the higher priority. This can avoid the case that many result have the same priority and make the program hard to determine with result is the most satisfy as the highest priority course will always in the top of List.