**Module Details**



**Computer Science and Creative Technologies**

**Module Code** UFCFB6-30-2

**Module Title** Object-oriented systems development

**Module Leader** Rakib Abdur

**Module Tutors** Kun Wei, Stewart Green, Nathan Duran, Barkha

Javed, Jun Hong

**Year** 2020-21

**Component/Element Number** B 1

**Total number of assessments for this module**

Two assessments

(1) Part-I: The first covers the analysis and design of a software system.

(2) Part-II: The second covers the coding and testing of the design created in the first

assignment.

**Weighting** This coursework is worth 100 marks representing

50% of your total course grade.

**Element Description** This assignment is to be completed individually.

**Dates**

**Date issued to students** November 16, 2020

**Making results and feedback to students**

(1) Part-I: February 8, 2021

(2) Part-II: April 26, 2021

**Submission Date** (1) Part-I: December 16 Wednesday@14:00, 2020

(2) Part-II: March 10 Wednesday@14:00, 2021

**Submission Place** Blackboard

**Submission Time** 14:00

**Submission Notes** (1) Part-I: Please submit a PDF document containing Use Case diagram, Class diagram, and Sequence diagram.

(2) Part-II: Please submit a portfolio as a ZIP file

with your code, Test cases used in your program in the form of a table, suitable program running screenshots capturing success/failure scenarios, as well as you need to make a short video to run and show all the

features of your software.

**Feedback**

**Feedback provision will be** Verbal feedback during the feedback session + as appropriate written feedback uploaded to Blackboard.

2

**Contents**

Module Details........................................................................................................... 1

Dates..................................................................................................................... 1-2

Feedback .................................................................................................................. 1

Contents ...................................................................................................................... 3

Section 1: Overview of Assessment ............................................................................ 4

Section 2: Task Specification...................................................................................... 5

Section 3: Deliverables.............................................................................................10

Section 4: Marking Criteria........................................................................................12

Section 5: Feedback mechanisms ..................................................................................15



**Section 1: Overview of Assessment**

This assignment assesses the following module learning outcomes:

• Apply object-oriented analysis and design techniques to a problem domain scoped at level 2 complexity

• Understand and use a Unified Modelling Language (UML) modelling tool anda Java- based Interactive Development Environment (IDE) to develop object-oriented software implementations appropriate to level 2 complexity

• Design and implement Graphical User Interfaces

• Apply good practice in code design/testing

The assignment is worth **50%** of the overall mark for the module.

Broadly speaking, the assignment requires you to design and implement a moderately realistic object-oriented system. You will produce detailed object models and designs from system requirements; use the modelling concepts provided by UML. You will then map the designs into code and perform testing using an automated testing tool.

The assignment is described in more detail in section 2. This is an **Individual** assignment.

Working on this assignment will help you to understand more clearly the concepts, problems, and techniques of object-oriented systems, and how these can be used to design

and implement a reasonably large size software system (to a professional standard).

If you have questions about this assignment, please contact/discuss with your lab tutors.

**Section 2: Task Specification**

**Note: You may develop the system using Java (and JavaFX) considering the following functionalities. You should be creative and come up with your own User Interface Design. The following design shows just an example.**

**2.1 Problem description**

The local table tennis association would like you to design and implement a computer system to support the association secretary, Brian, and the players.

Currently, after each match the players send the match scores to Brian. Brian keeps paper copies of the scores, and he subsequently produces various reports such as team scores and

rankings etc.

The association has about a dozen of teams. Each team has 2 or more players. During each year, each team will play two matches against every other team, once at the home venue and once at the away venue. Currently every match has two players from each team. A match consists of 4 single sets and a double set. Every set consists of 3 games. The player who wins the most games wins the set. (For simplicity, you can assume that every set always has three games even if one player has already won the first two games.) The following shows an example of the match score sheet. In this example, the two teams are “page” and “uwe”. The match is played at page’s venue. So “page” is the home team and “uwe” is the away team. “peter” from team “page” has won two sets, one against “jin” from

team “uwe”, the other against “julia” from team “uwe”. “phil” from team “page” has not won

any set. The home team has won the double set. In total, the home team has won 3 sets

and lost 2 set. So “page” has won the match.

The association is expecting to have a lot more interests in the near future. Hence, they are likely to have more teams and more players. Any system designed for now needs to take this into account so that future modifications can be done easily.



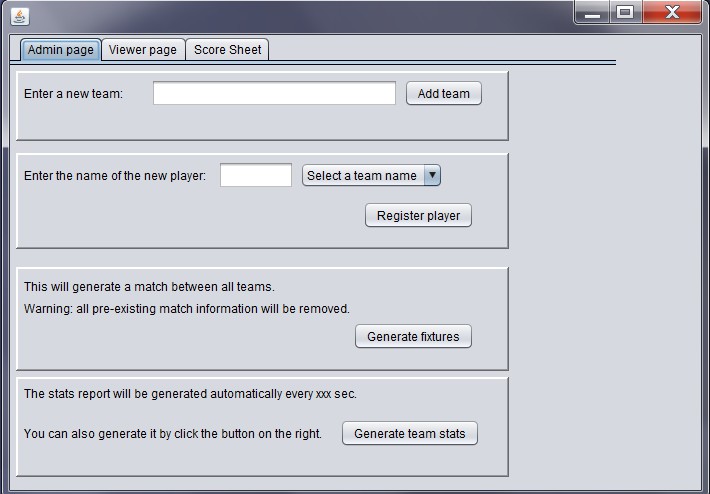
**2.2 Functionalities for the computer system**

The association has done some design for the front end of the system. The following illustrates what they would like to have. There should be three pages/tabs:

• Admin page

• Viewer page

• Score sheet page



For the Admin page, the association secretary should be able to

• Add a new team: enter the team name and click on the add team button.

• Add a player and attach the player to a team: enter a player’s name and select a

team from the select team dropdown box and then click the register player button

• Generate match fixtures: clicking the generate fixtures button will create the fixtures which includes a match being set for every pair of teams registered.

• Generate team statistics: This button enables the association secretary to force the

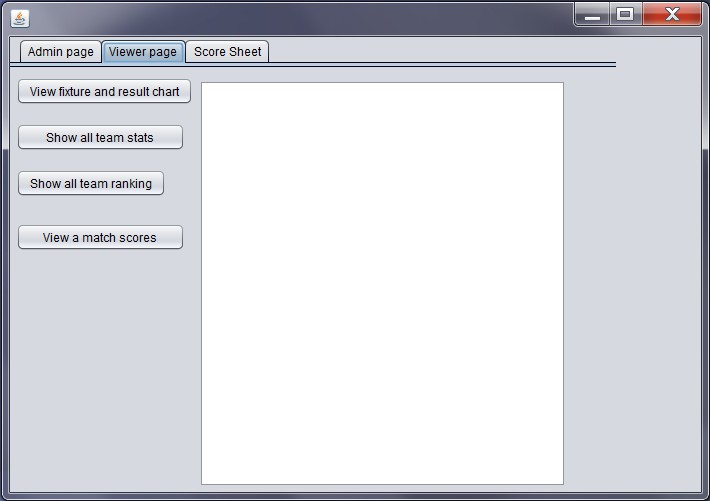
system to generate the team stats, i.e. how many matches have played, how many have been won and how many sets have been won. The system should also generate the team stats automatically every 100 seconds. Here is an example of a list of team stats:

Uwe: matchesPlayed=2, matchesWon=1, setsWon=6

Page: matchesPlayed=1, matchesWon=0, setsWon=1

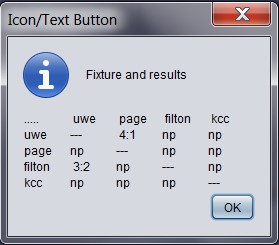
Filton: matchesPlayed=1, matchesWon=1, setsWon=3

Kcc: matchesPlayed=0, matchesWon=0, setsWon=0



For the Viewer page, the viewers/players and the association secretary should be able to

• View match fixtures and the result chart: this will display all the fixtures and the current status of the matches. The following is an example. In this example, there are 5 teams. The first column indicates the names of the home team. The top row indicates the away team names. “np” indicates that the match has not been played yet. In the example below, uwe (as home team) has played page (as away team). The final score is 4:1, i.e. uwe has won 4 sets and page has won 1 set.



• View the scores of a previously played match. This will request the user to enter the home team name and the away team name. If the match has been played, it will display all the information about the match. If it has not been played, it will display a message saying so. The following is an examle:

Match: uwe vs page

SingleSets

Set{jin vs peter = 11:2,3:11,11:5} Set{jin vs phil = 11:1,5:11,11:6} Set{julia vs peter = 11:9,11:1,11:1} Set{julia vs phil = 11:2,3:11,11:5}

Double set: Set{null vs null = 0:11,1:11,2:11}

final scores: 4:1

Note that there is no need to store the player’s names for the double set.

• View the team stats of all teams. For each team, this includes the number of

matches played, the number of matches won and the total number of sets won by its players. For example, here is an output:

Uwe: matchesPlayed=2, matchesWon=1, setsWon=6

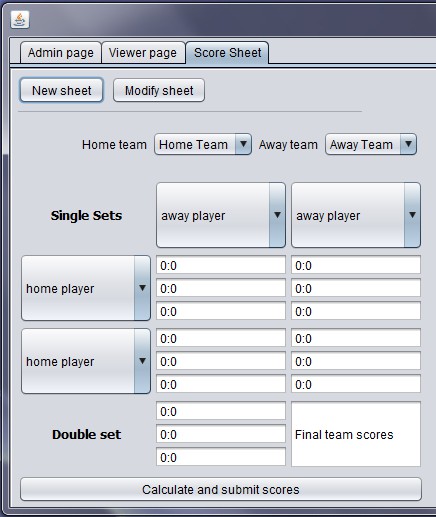
Page: matchesPlayed=1, matchesWon=0, setsWon=1

Filton: matchesPlayed=1, matchesWon=1, setsWon=3

Kcc: matchesPlayed=0, matchesWon=0, setsWon=0

• View the team ranking. The information displayed will be the same as above, but it is in the order of sets won by each team. Here is an example output:

uwe: matchesPlayed=2, matchesWon=1, setsWon=6 filton: matchesPlayed=1, matchesWon=1, setsWon=3 page: matchesPlayed=1, matchesWon=0, setsWon=1 kcc: matchesPlayed=0, matchesWon=0, setsWon=0



For the Score sheet page, the user should be able to

• Create a new score sheet, or

• Modify an existing score sheet

For creating a new score sheet, the user should

• Select the home team and the away team from the drop down menues.

• Once the teams have been selected, the system should set the correct players for the relevant players dropdown boxes, i.e. if the user select “uwe” for the home team, the dropdown menus for the home players should be set to have all the

players in the uwe team. The user then select the players for the home team and for

the away team.

• The user then enter the scores for all the sets.

• Once all the scores are entered, the user can click the “calculate and submit scores”

button. The information about this match should all be stored in the system.

For modifying an existing score sheet, the user should

• Enter the home team and the away team names. The system should find the information about the match and display relevant score sheet.

• The user can change any information about the players and the scores, and then

click on the “calculate and submit scores” button. The information about this match

should all be updated in the system.

**Section 3: Deliverables**

You are asked to produce the following deliverables for the scenario given above.

**Part-I**

1. Produce a use case diagram to capture the functionality for the system to be built.

2. Produce a class diagram to meet all the requirements captured in the use case

diagram.

3. Produce at least three sequence diagrams for some use cases that adds information.

**Part-II**

4. Develop the system using Java (and JavaFX) considering all the functionalities described above.

5. Testing is typically a part of the program development – you should use a test strategy to test your system thoroughly.

You need to have the following test data in your system. The markers will use some of these data to test your programme. If you do not have these data in your system, your

programme may not pass some of the test cases.

Teams: filton, uwe, kcc, page

Players:

“filton” has the following players: “alex”, “brian”.

“uwe” has the following players: “jin”, “julia”, “stewart”. “kcc” has the following players: “chris”, “ryan”.

“page” has the following players: “peter”, “phil”.

Match Scores: Match: filton vs uwe SingleSets

Set{alex vs jin = 11:2,3:11,11:5} Set{alex vs julia = 1:11,5:11,11:6} Set{brian vs jin = 11:9,11:1,11:1} Set{brian vs lulia = 11:2,3:11,11:5}

Double set: Set{null vs null = 0:11,1:11,2:11}

Match: uwe vs page

SingleSets

Set{jin vs peter = 11:2,3:11,11:5} Set{jin vs phil = 11:1,5:11,11:6} Set{julia vs peter = 11:9,11:1,11:1} Set{julia vs phil = 11:2,3:11,11:5}

Double set: Set{null vs null = 0:11,1:11,2:11}

**Section 4: Marking Criteria**

**Part-I**

0.0-2.9 3.0-3.9 4.0-4.9 5.0-5.9 6.0-6.9 7.0-8.4 8.5-10.0 Mark & Advice for Improvem ent

**Use case diagram (10%)**

**Class diagram (10%)**

Little or no diagram provided (overall less than

30%

complete

).

Little or no diagram provided (overall less than

30%

Partial diagram provided but actors and/or use- cases/relation ships are only partially correct (overall less than 40% complete).

Partial diagram provided but Class naming convention, Class relationships,

Partial diagram provided, actors and/or use- cases/relation ships are correct (overall less than 50% complete).

Partial diagram provided but Class naming convention, Class relationships,

Almost complete diagram provided, actors and/or use- cases/relation ships are only partially correct (overall less than 60% complete). Almost complete diagram provided, Class naming convention, Class

Almost complete diagram provided, actors and/or use- cases/relation ships are correct. (overall, less than 70% complete)

Almost complete diagram provided, Class naming convention, Class

Complete diagram provided, actors and/or use- cases/relation ships are partially correct (overall less than 85% complete)

Complete diagram provided, Class naming convention, Class relationships,

Complete diagram provided, actors and/or use- cases/relation ships are correct

(100%

complete)

Complete diagram provided, Class naming convention, Class relationships,

**Sequence diagram (at least three diagrams) (10%)**

complete

).

Little or no diagram provided (overall less than

30%

complete

).

Attributes/met hods of the classes and their visibility and type are only partially correct (overall, less than 40% complete).

Partial diagram provided but Lifeline Notation, Activation Bars, and Message Arrows are only partially correct (overall less than 40% complete).

Attributes/met hods of the classes and their visibility and type are correct (overall, less than 50% complete).

Partial diagram provided, Lifeline Notation, Activation Bars, and Message Arrows are correct (overall less than 50% complete).

relationships, Attributes/met hods of the classes and their visibility and type are only partially correct

(overall less than 60% complete). Almost complete diagram provided but Lifeline Notation, Activation Bars, and Message Arrows are only partially correct (overall less than 60% complete).

relationships, Attributes/met hods of the classes and their visibility and type are correct

(overall less than 70%

complete).

Almost complete diagram provided, Lifeline Notation, Activation Bars, and Message Arrows are correct (overall less than 70% complete).

Attributes/met hods of the classes and their visibility and type are partially

correct (overall less than 85% complete)

Complete diagram provided but Lifeline Notation, Activation Bars, and Message Arrows are only partially correct (overall less than 85% complete).

Attributes/met hods of the classes and their visibility and type are correct

(100%

complete)

Complete diagram provided, Lifeline Notation, Activation Bars, and Message Arrows are correct (100% complete).

**Part-II**

0.0-8.9 9.0-11.9 12.0-14.9 15.0-17.9 18.0-20.9 21.0-23.9 24.0-30.0 Mark & Advice for Improvem ent

**Implementa tion**

**(30%)**

Little or no developm ent (overall less than

30%

complete

).

Partial development, compiles and runs, but GUI displays input and output messages only partially correctly (overall less than 40% complete).

Partial development, compiles and runs, GUI displays input and output messages correctly (overall less than 50% complete).

Partial development, compiles and runs, but GUI displays input and output messages only partially correctly (overall less than 60% complete).

Almost complete development, compiles and runs, GUI displays input and output messages correctly (overall less than 70% complete).

Complete development, compiles and runs, but GUI displays input and output messages only partially correctly (overall less than 85% complete).

Complete development, compiles and runs, GUI displays input and output messages correctly (100% complete).

0.0-5.9 6.0-7.9 8.0-9.9 10.0-11.9 12.0-13.9 14.0-16.9 17.0-20.0 Mark & Advice for Improvem ent

**Testing**

**(20%)**

**See at Page**

**16**

**Appendix A for the test plan**

Little or no test cases (overall less than

30%

tested).

Partially tested, (overall less than 40% tested).

Partially tested, (overall less than 50% tested).

Partially tested, (overall less than 60% tested).

Almost tested, (overall less than 70% tested).

Almost tested, (overall less than 85% tested).

Completely tested, (overall less than 100% tested).

**Video Recorded Presentatio n**

**(20%)**

Absent 0.0

Inadequate (barely able to explain the work done) 5.0

Good (good explanation of the work done) 10.0

Very Good (very good explanation of the work done) 15.0

Excellent (excellent explanation of the work done) 20.0

**Section 5: Feedback mechanisms**

Verbal feedback during the feedback session + as appropriate written feedback uploaded to Blackboard. Formative feedback provided in Part-I will be useful to complete Part-II of the assignment.

**Appendix A Test Plan**

Test case State of the

System

Input Output Mar ks

1.

Add a new team,

2.

Add a new player to a

team

The required test data are in the system

Same as above

Enter a new team name, (e.g. team “abc”) and click the add team button Enter a player name,

(may need to enter the

return key)

Select an existing team,

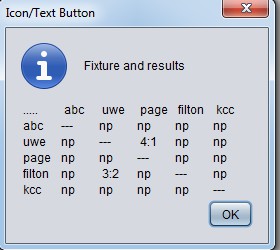
Click the register player button

No error and a confirmation /2 message

The select team drop down box /2 should have all the pre-entered

team names + plus the team added test case 1.

No error



3. Generate fixtures

4.

Show all team stats

The required test data (teams) are

in the system, a

new team

was added (e.g. team “abc”)

The required test data are in the

system, a new team

was added

(e.g. team

“abc”), Fixtures is generated

Click the generate fixtures button, And then go to the viewer’s page and click on “view fixtures and result chart”

Click the show all team stats button

The fixture should be generated as /2 shown

Note: “Generate fixtures” clears the score information for all matches. The above screen shot is the result of doing “Generate fixtures” and followed by populating the test data as specified in the assignment specification.

You need to populate these test data as the markers will be using these to mark your program.

The following are displayed in the /2 text area in the viewer page:

abc matchesPlayed=0, matchesWon=0, setsWon=0

uwe matchesPlayed=2, matchesWon=1, setsWon=6

page matchesPlayed=1, matchesWon=0, setsWon=1

filton matchesPlayed=1, matchesWon=1, setsWon=3

kcc matchesPlayed=0, matchesWon=0, setsWon=0

5.

View a match

6.

Enter new scores

Enter home team and away team, e.g. uwe and page

Click the “New sheet” button in the “Score sheet” tab,

Select a home team and an away team

from the drop down

boxes

Select suitable players from the drop down boxes

Enter the points (Test data may be

automatically generated. Code for

this will be given.)

Note that this may not show the latest stats. It will only show the latest team stats after the “generate team stats” function is triggered either manually or by the time trigger.

The following is an example /2 showing what information are

required to be displayed in the text area in the viewer page:

Match: uwe vs page

SingleSets

Set{jin vs peter = 11:2,3:11,11:5} Set{jin vs phil = 11:1,5:11,11:6} Set{julia vs peter = 11:9,11:1,11:1} Set{julia vs phil = 11:2,3:11,11:5} Double set: Set{null vs null =

0:11,1:11,2:11}

final scores: 4:1

If a match is not played yet, display the message “match not played yet”.

/2

All registered teams should be in the team dropdown boxes.

Players from the right team should be listed in each of player drop down box.

7.

Calculate and submit scores

Test case 6 is successfully completed

Click the “Calculate and submit scores” button

The final score should be /2 displayed in the text field at the

bottom right corner, the following

is an example

8.

Modify an existing score

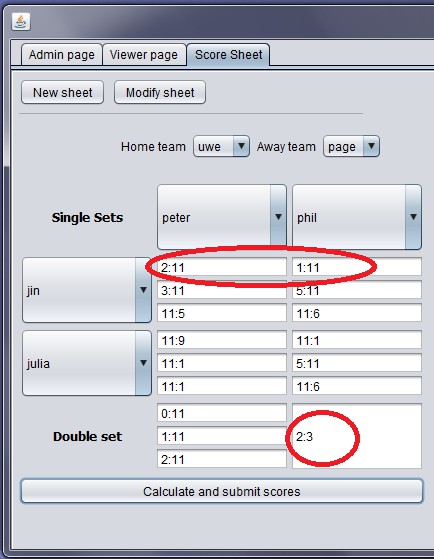
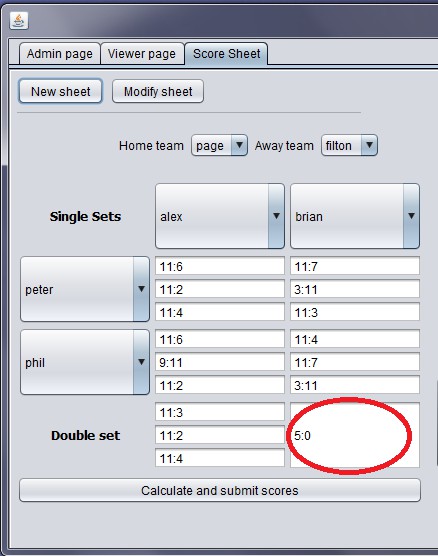
sheet

Click the “Modify sheet” button in the “score sheet” tab

System prompts for home team name, user enter home team

System prompts for away team name, user enter away team

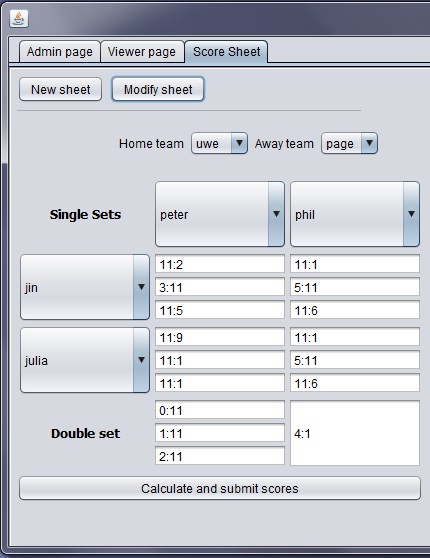
System brings up the score sheet (example shown on the right)



Change some scores and click the “Calculate and submit scores” button

It should bring up the score sheet /2 as follows:

It should revise the scores, for example:



9.

View fixtures and result

chart

Test cases 7 and 8 are completed

Click on “View fixtures and result chart” button

The result chart should show the /2 changes, for example if we use the

data entered in test cases 7 and 8, we get the following:

10.

Show team ranking

Team stats need to be generated since the changes made in test cases 7, 8 and 9. This can be done in two ways: click the “Generate team stats” button in the Admin tab

or wait for the reporting timer thread to generate it automaticall y. The

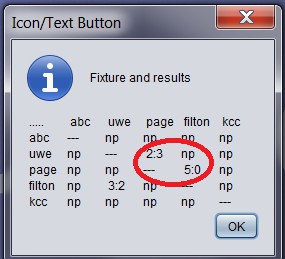
marker may check for

either.

It should display the teams in the /2 order of the number of sets won.

The following is an example based on the data used in the above test

cases.



[page: matchesPlayed=2, matchesWon=2, setsWon=8,

uwe: matchesPlayed=2, matchesWon=0, setsWon=4,

filton: matchesPlayed=2, matchesWon=1, setsWon=3,

abc: matchesPlayed=0, matchesWon=0, setsWon=0,

kcc: matchesPlayed=0, matchesWon=0, setsWon=0

]

Note that this may not show the latest stats. It will only show the latest team stats after the “generate team stats” function is triggered either manually or by the time trigger.