NATIONAL UNIVERSITY OF SINGAPORE

CS2103/T – SOFTWARE ENGINEERING

(Semester 2 AY2014/2015)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. Answer **ALL** questions within the space in this booklet.
2. This is an OPEN BOOK assessment.

STUDENT NO: ______________________________________

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/7	
Q2	/10	
Q3	/11	
Q4	/8	
Q5	/4	
Total	/40	

[Examiner comments]

- Do not take example answers given here as 'the only right answer'. In most cases it is possible to get full marks even if your answer differs from the example given.
- The project used in this paper is slightly harder and slightly different than the project done during the semester.
- This assessment paper does not cover project management and implementation. Those aspects were covered using CA.

The **problem description** used for all questions in this assessment paper is given in the appendices (last two pages of this assessment paper).

Q1 [5+2=7 marks]:
Please read *Appendix A* of the problem description. When answering this question, you may consider the requirements specified in the problem description as well as requirements that you would like to add. Assume that you have categorized PointTracker user stories into categories *must-have*, *nice-to-have*, and *unlikely-to-have*.

a) Give one *nice-to-have* user story for each of the four most important user types. Three user types to include are Lecturer, Tutor, and Student. You may decide the fourth user type.

[Examiner comments]
Areas tested by this question: Functional requirements, product design

Answers should be nice-to-have user stories, not must-have ones. For example, 'tutor can edit points' is a must-have user story because without it the system becomes impractical to use. User stories should be written in correct format and at least some should mention the benefit. The 4th user type should be one of the important ones, for example, `Admin`.

Some examples:

1. As the lecturer, I can see how many times a tutor did not submit points on time so that I can identify tutors who are habitually late in submitting points.
2. As a tutor, I can get notifications of upcoming point submission deadlines so that it reduces the risk of me missing a deadline.
3. As a student, I receive updates to my points in email so that I don't have to remember to check my points.
4. As an admin, I can reuse past data when creating accounts so that it is easier for me to create accounts for returning tutors/instructors.

b) Give a non-functional requirement for PointTracker. Choose a requirement that is more specific to PointTracker, as opposed to a requirement that is applicable to most other software.

[Examiner comments]
Areas tested by this question: Non-functional requirements.
An example:

- The user should have logged into the computer before the desktop application is activated.

Common mistakes:

- Giving a functional requirement when the question requires a non-functional requirement.
- The given answer is not specific enough to the PointTracker system. E.g. good usability is a non-functional requirement but it is not specific to the PointTracker.

Q2 [2+8=10 marks]:

(a) Consider the two architecture choices given in *Appendix B*. Which one is better? Justify your answer.

[Examiner comments]

Areas tested by this question: Architecture, design quality

The answer should reference concepts related to design quality such as coupling and cohesion.

Example answer:

Choice B is better because in that design the `Parser` does not deal with executing the command and therefore more cohesive (it does parsing only) and not coupled to the `Logic` component.

Common mistakes:

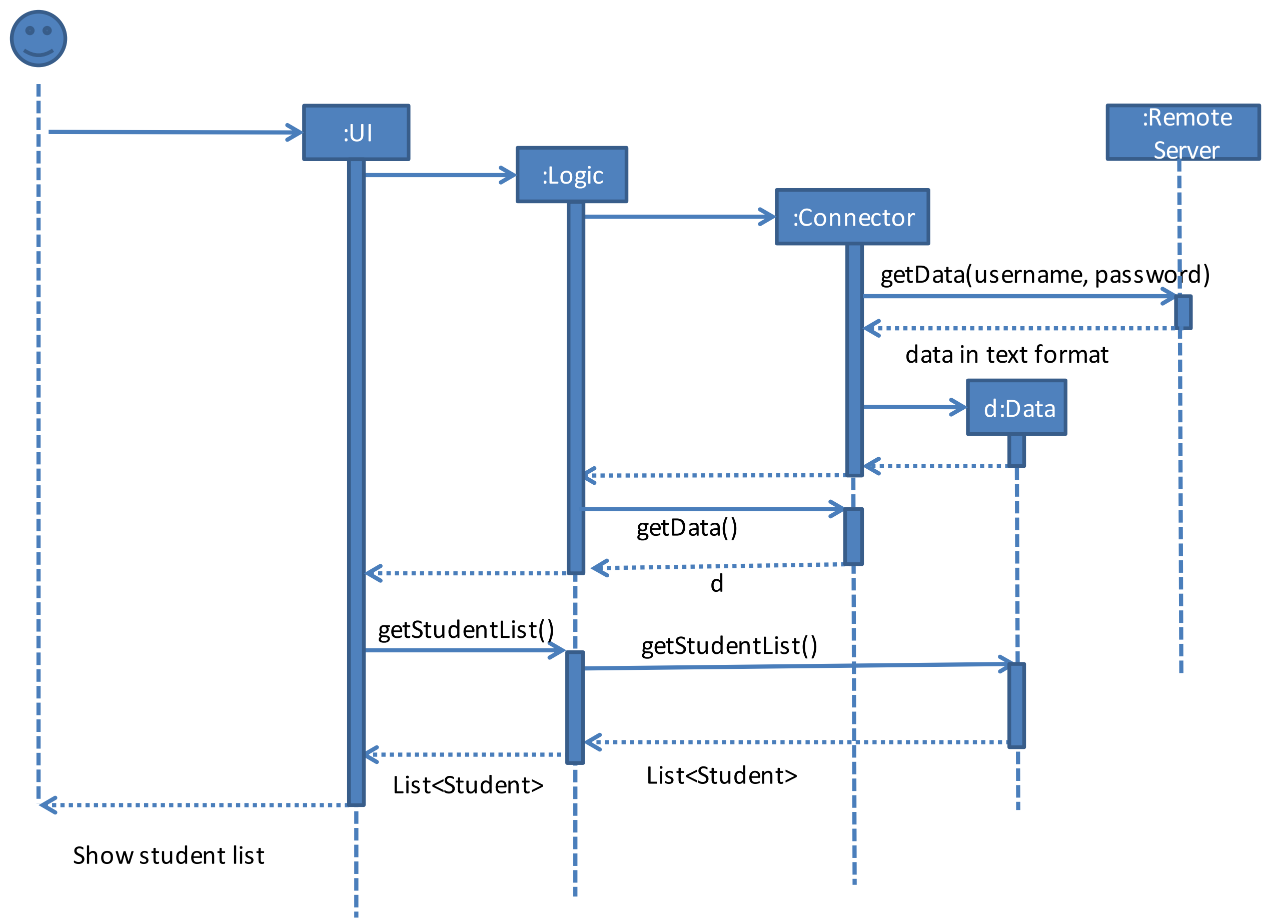
- No justification given or the justification does not reference relevant concepts.

(b) Assume that when a tutor runs the PointTracker executable, it initializes the components, downloads the relevant data from the remote server, and shows the student list of the class. Use suitable UML diagram(s) to explain the interactions <u>between architectural components</u> required during that scenario. Remember to include the API functions taking part in the interactions.

[Examiner comments]

Areas tested by this question: API design, Sequence diagrams

An example:



Common mistakes:

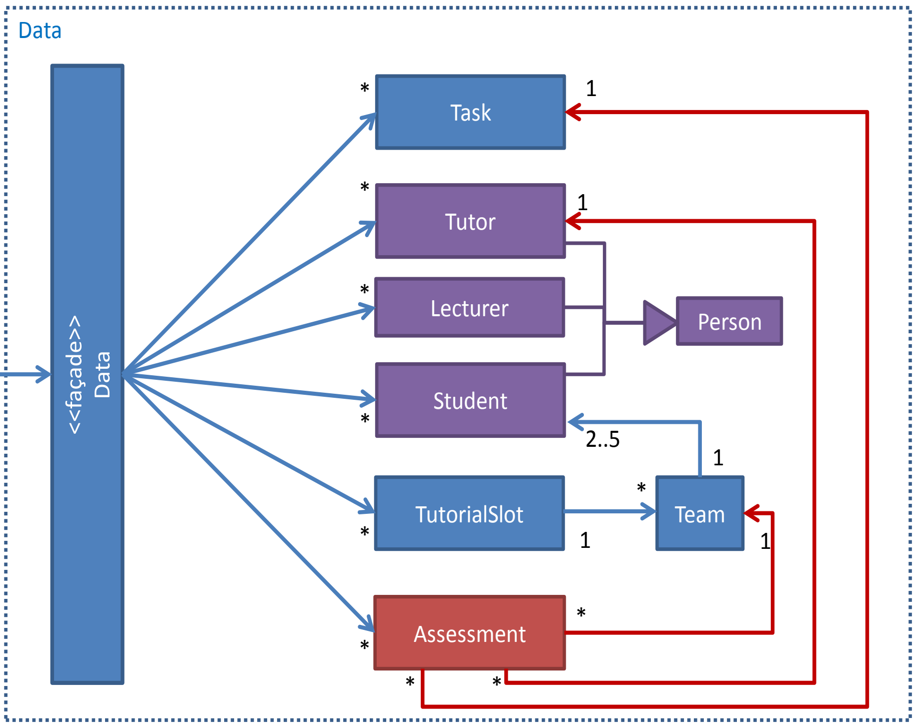
- Not using a sequence diagram (Out of the UML diagrams we have covered, Sequence diagrams is the most suitable to illustrate interactions between components for a specific scenario).

- Incorrect sequence diagram notations
- Sequence diagram not drawn at architecture level or does not match one of the two architectures given
- Not showing object construction in the sequence diagram
- Not showing the `getData(username, password)` function in the diagram

Q3 [8+3=11marks] (a) Use suitable UML diagrams to propose an object-oriented design for the `Data` component. Include navigabilities, multiplicities, association roles/labels etc. when they add value to the diagram. Methods and attribute details need not be shown. Your design should include the following class.

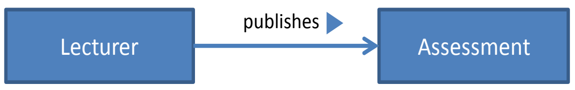
Assessment: An object of this class represents the assessment by a tutor for a specific tutorial task for a specific team.

[Examiner comments]
Areas tested by this question: OOP, class diagrams
An example:



Common mistakes:

- Not using a class diagram. Some drew a domain model instead.
- Showing unnecessary associations. For example, the 'Lecturer publishes Assessment' association (shown in the diagram below) should not be included unless PointTracker is intended to track which Lecturer published which Assessment.

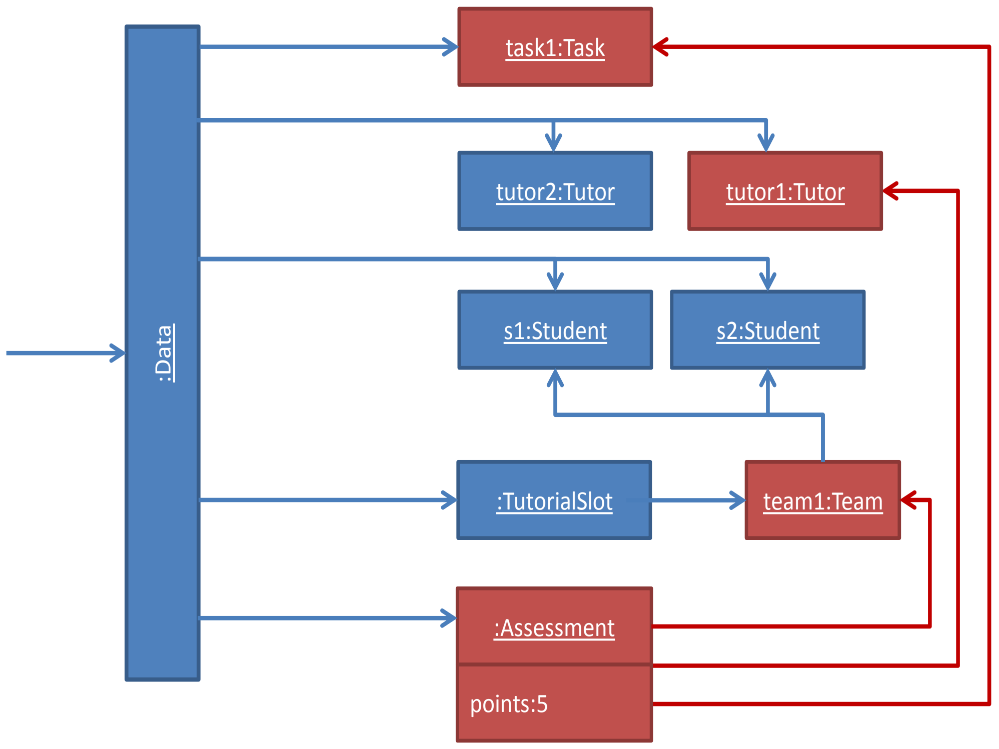


Keep in mind that class diagrams are structure diagrams. They show structural relationships between objects, not how objects behave. Of course most structural relationships are results of certain behaviors. For example, 'a tutor grades a team for a specific task' is a behavior but it results in structural links between `Tutor`, `Team`, `Task`, and `Assessment` objects because we need to keep track of which tutor gave how many points to which team for which task.

- Not following the correct class diagram notation.
- Not including the `Assessment` class or not showing how Assessment is associated with `Team`, `Tutor`, and `Task` classes. These associations are required because the question mentions that `Assessment` class represents the assessment by a tutor for a specific tutorial task for a specific team.
- Not showing the entire component (Some drew only the classes mentioned in the question).
- Not showing important multiplicities and navigabilities.

(b) Consider the sample data given in *Appendix C* and your proposed design above. Use a suitable UML diagram to show the object structure in the Data component for the given sample data.

[Examiner comments]
Areas tested by this question: object diagrams
An example:



Common mistakes:

- Not using an object diagram
- Not using the correct object diagram notation e.g. underline missing from the class/object names.
- The diagram does not match with the data given
- The diagram does not match with the class diagram in part (a)
- The diagram does not capture the data item '5 points'
- Showing the parent class `Person` in the object diagram as a separate object.

Q4 [3+5=8 marks]:

(a) Consider testing of the assignPoints function given in *Appendix D*. Give equivalence partitions for each parameter.

[Examiner comments]

Areas tested by this question: test case design heuristics

An example:

Parameter	Equivalence partitions
teamID	{belongs to a team in the class} {not a team in the class/ everything else}
tutorUsername	{belongs to a tutor in the class, assessing the team} {belongs to a tutor in the class, but not assessing the team} {everything else}

points	{less than 0}
	{0..10}
	{more than 10}

Other possible equivalence partitions: empty string, null, a user id of a student/lecturer

Common mistakes:

- Not including the valid data partition. For example, some forgot to include the {0..10} partition.
- Giving boundary values instead of describing the partition.

(b) Assume you have been asked to unit test the assignPoints function and the software has been loaded with the sample data given in *Appendix C*. Add 7 more unit tests to the example given. Try to maximize the efficiency and effectiveness of testing when you design the test cases.

[Examiner comments]

Areas tested by this question: test case design heuristics

An example:

#	teamID	tutorUsername	points	expected
1	team1	tutor1	5	true
2	Any non-existing	Any existing	Any valid value	false
3	Any existing	Any non-existing	Any valid value	false
4	team1	tutor2	Any valid value	false
5	team1	tutor1	-1	false
6	team1	tutor1	11	false
7	team1	tutor1	0	true
8	team1	tutor1	10	true

Common mistakes:

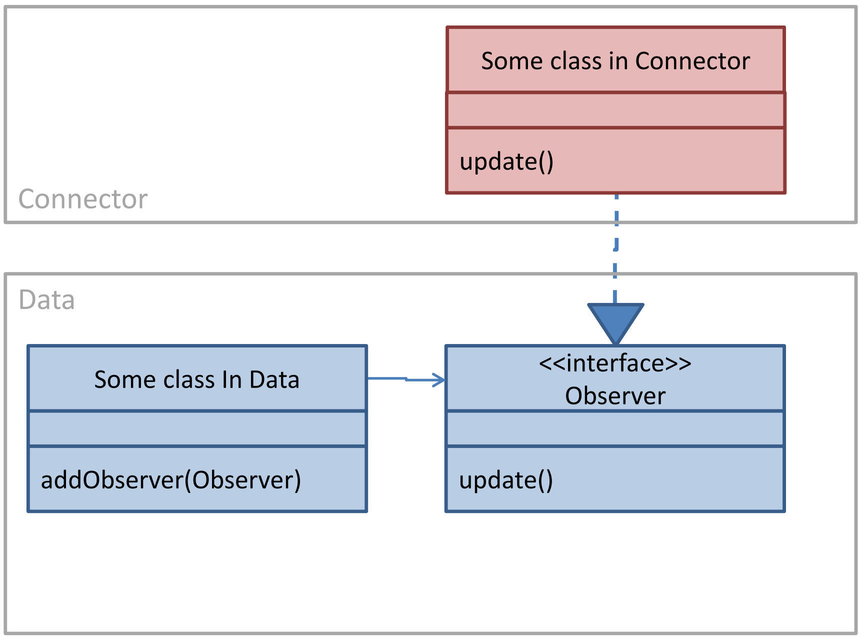
- Missing important boundary values -1, 0, 10, 11

- Combining multiple invalid values in a test case. For example, `{team1, tutor2, 11}` contains two invalid values because `tutor2` is not assigned to `team1`.
- Giving `infinity` as a test input. Integer data type cannot have `infinity` as an input.

Q5 [4 marks]:
Now, assume there is a need for the Data component to notify the Connector component when there is a change to the data, without adding a direct dependency from Data component to the Connector. Illustrate a way to achieve that using appropriate diagrams.
Hint: Use a suitable design pattern.

[Examiner comments]
Areas tested by this question: design patterns
The expected answer is 'Observer pattern'. To get full marks, the answer should have contained an illustration of how the pattern is to be applied.

An example (outline only):



Common mistakes:

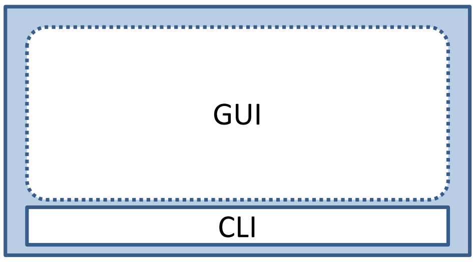
- Giving Façade pattern as the answer. Façade pattern is useful for hiding the internal structure of a component from the clients of that component. Clients still depend on the component that uses the Façade pattern. With the observer pattern, clients can communicate with the component without having a direct dependency on it.

--- End of Paper---

Given below is the **problem description** used for all questions in this assessment paper. You may detach this page from the assessment paper. There is no need to submit this page.

Appendix A: PointTracker product description

Your team has been asked to build a software application called **PointTracker** to keep track of CS2103 participation points. It has a desktop application and a remote server. Here is the expected behavior of PointTracker.



1. Users access PointTracker using the desktop application. The remote server does not have a user interface.

2. PointTracker desktop application has a GUI and a Command Line Interface (CLI).

3. Tutors use the PointTracker desktop application to enter participation points students earned for each tutorial task. After the lecturer publishes the points for a particular tutorial task, students can use the PointTracker desktop application to view their points.

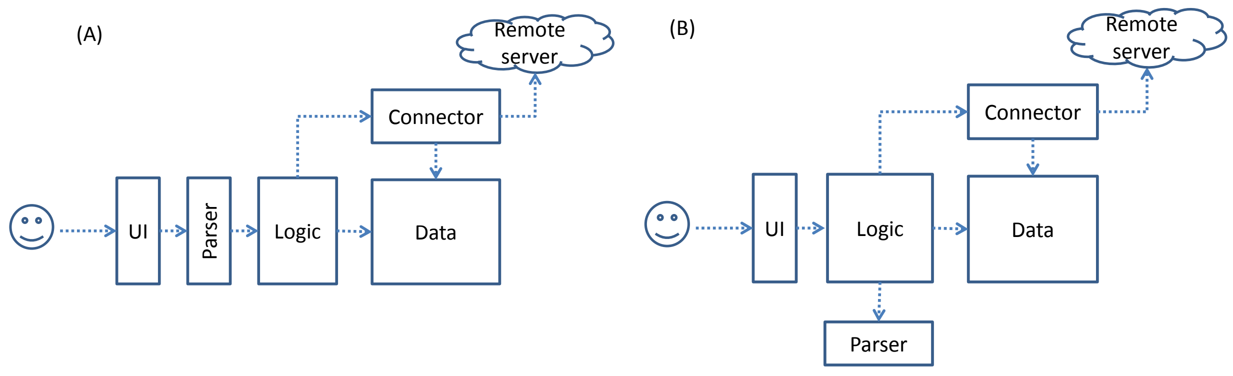
4. PointTracker desktop application does not store data locally (i.e. in the user's Computer). Instead, it connects to the PointTracker remote server to read/write data.

5. Authentication to the remote server is done using the login credentials the user used to log in to the computer. PointTracker picks up the username and password from the Operating System. There is no need for the user to type in username and password when using PointTracker.

6. The proposed PointTracker system has the following simplifications:
    a. It deals with data of the current semester only. Data of the past semesters are deleted at the end of the semester.
    b. It deals with team-based tutorial tasks only. It will not deal with individual tutorial tasks.
    c. The tutor allocation can change over time. A tutor may assess a different set of teams each week. However, each team (comprising 2 to 5 members) is allocated a tutorial slot (e.g. Wed 9am) which does not change for the entire semester.
    d. PointTracker keeps track of which tutor gave how many points to which tutorial task of which team.

Appendix B: Architecture choices

Given below are two architecture choices for PointTracker. The difference is in how the Parser component is connected.



An overview of the components is given below. It applies to both diagrams.

- `Parser` : Parses the CLI commands typed by the user
- `Data` : The in-memory container for objects representing PointTracker data. E.g. Tutors, Students, Tutorial Tasks, Assessments, etc.
- Remote Server: Represents an online server that stores PointTracker data. It is accessed by PointTracker using a remote API, which includes the following function.
  getData(username, password): String
  Returns all relevant data of the user in a String format (e.g. XML or JSON) if the username and the password are correct.
- `Connector`: Handles the connection with the remote server API.
- UI, Logic: As the names suggest.

Appendix C: Sample data

Given below are a set of sample data that exists in PointTracker at a particular point of time.

- Usernames of the tutors in the class: tutor1, tutor2
- Usernames of the students in team1: s1, s2
- tutor1 gave 5 points for task1 for team1

Appendix D: Testing the assignPoints function

assignPoints(string teamID, string tutorUsername, int points):boolean
Returns true if all the following conditions are true.

- teamID belongs to a team in the class
- tutorUsername belongs to a tutor of the class
- points are in the range 0..10 (both inclusive)
- tutor indicated by the tutorUsername is assigned to assess the team indicated by teamID

-- End of problem description--