NATIONAL UNIVERSITY OF SINGAPORE

CS2103/T – SOFTWARE ENGINEERING

(Semester 1 AY2014/2015)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. Please write your Student Number only. Do not write your name.
2. This assessment paper contains **SIX** questions and comprises **NINE** printed pages.
3. Answer **ALL** questions within the space in this booklet.
4. This is an OPEN BOOK assessment.

STUDENT NO: ______________________________________

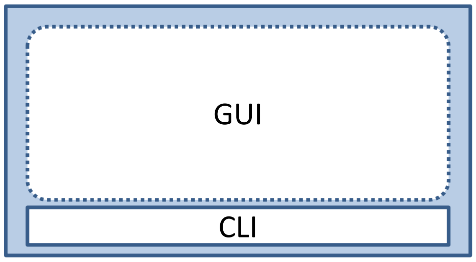
This portion is for examiner's use only

Question	Marks	Remarks
Q1	/5	
Q2	/5	
Q3	/10	
Q4	/10	
Q5	/5	
Q6	/5	
Total	/40	

Given below is the scenario used for all questions in this assessment paper:

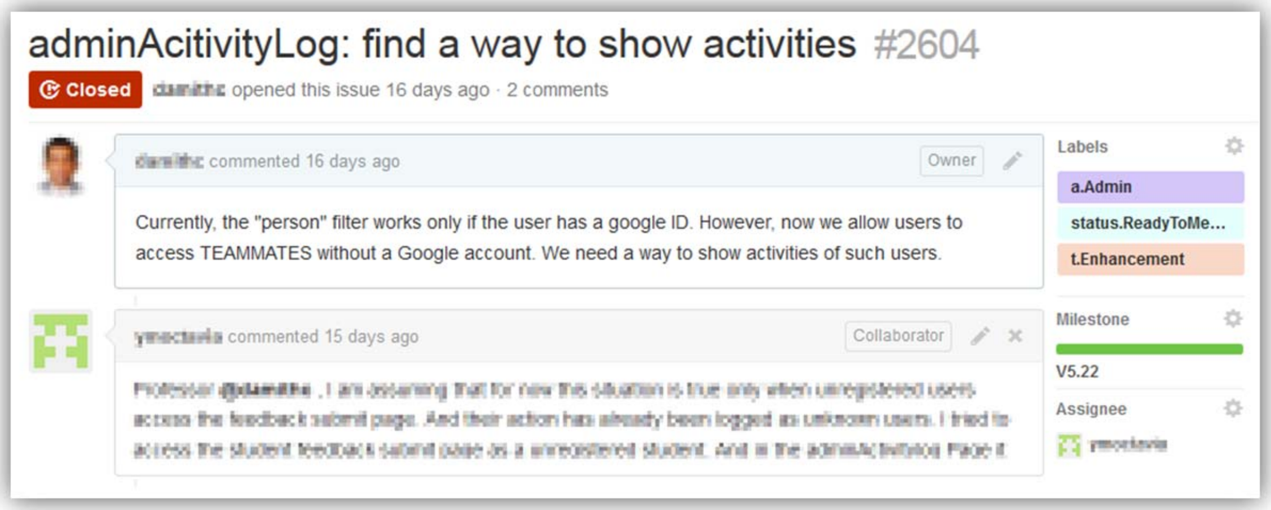
Your team has been asked to build a desktop client for GitHub issue tracker. The name of the application is **HubIssues**. It is expected to provide a more convenient way to access the GitHub issue tracker so that users can work with GitHub issues without visiting the GitHub Website. Here is the expected behavior of HubIssues.

1. HubIssues has a GUI and a Command Line Interface (CLI). Most actions can be performed using either the GUI or the CLI, whichever the user finds more convenient.
2. HubIssues does not store issues locally (i.e. in the user's Computer). Instead, it connects to the GitHub remote APIs to read/write issues in the GitHub servers. However, user preferences are saved locally in a text file.



3. HubIssues synchronizes with issues on GitHub after each write operation on HubIssues. E.g. after a user edits an issue using HubIssues. In addition, users can force HubIssues to synchronize either by typing a command in the CLI or clicking a button on the GUI. There is no need to use multi-threading to auto-synchronize periodically.
4. HubIssues will show issues of only one project at a time.
5. HubIssues supports the concept of *issue hierarchies*. i.e users can designate an issue as the parent of another issue. For example, an issue representing an epic can be designated as the parent of issues representing the user stories related to that epic. You can assume that the GitHub API supports issue hierarchies although the current GitHub Web interface does not.
6. HubIssues ignores issues that represent *pull requests*. If you do not know what pull requests are, you can safely ignore this point.
7. HubIssues does not interact with either the local Git repo or the remote Git repo on GitHub.

Given below is a screenshot of an issue in the GitHub issue tracker (Web UI), to refresh your memory.



adminAcitivityLog: find a way to show activities #2604

Closed [redacted] opened this issue 16 days ago · 2 comments

[redacted] commented 16 days ago Owner

Currently, the "person" filter works only if the user has a google ID. However, now we allow users to access TEAMMATES without a Google account. We need a way to show activities of such users.

[redacted] commented 15 days ago Collaborator

[redacted]

Labels: a.Admin, status.ReadyToMe..., t.Enhancement

Milestone: V5.22

Assignee: [redacted]

Note: All questions in this assessment paper are interlinked. Please answer sequentially. Informally-drawn UML sketches with just enough information are acceptable. Answers may be written using a pencil.

Q1 [1+2+2 = 5 marks]:
Assume you have categorized HubIssues user stories into categories *must-have*, *nice-to-have*, and *unlikely-to-have*.
When answering this question, you can consider the requirements specified in the previous page as well as requirements you would like to add yourself.

a) Give a *must-have* user story for HubIssues.

b) Give a *nice-to-have* user story for HubIssues. Choose a user story that makes HubIssues more useful than the GitHub Web interface.

c) Give a non-functional requirement for HubIssues. Choose a requirement that is specific to HubIssues (as opposed to a requirement that is applicable to many other software).

Q2 [5 marks]:

Propose an architecture for HubIssues that includes the following high-level components. You may not leave out any of the given components, but you may add more components. You are expected to minimize coupling and maximize cohesion in your architecture.

- `Parser` : Parses the CLI commands typed by the user
- `Data` : The in-memory container for objects representing HubIssues data. E.g. Issues, Users, Labels, Milestones, Comments, etc.
- `Connector`: Handles the connection with the GitHub remote API.

Q3 [10 marks]:

Given below are two CLI commands available in HubIssues.

`select` `keyword1 [keyword2]…` Selects the issue that is the best match for the keywords specified. e.g. **`select`** `customer crash minimize` The above example will search issue titles for words `customer`, `crash`, and `minimize` and choose the issue with the 'best matching' title as the 'selected' issue. This chosen issue will remain as the 'selected' issue until another issue is selected.	
`label` `label1 [label2] …` Adds one or more labels to the currently 'selected' issue. e.g. **`label`** `urgent bug ui` The above example will add the labels `urgent`, `bug`, and `ui` to the currently selected issue.	

Use suitable UML diagram(s) to explain the interactions <u>between architectural components</u> required to execute the command **`select`** `window view` followed by the command **`label`** `bug medium`
The diagram(s) should show the API methods used in the interactions.

Q4 [10 marks]:
Propose an object-oriented design for the `Data` component.
Include navigabilities, multiplicities, association roles/labels etc. when they add value to the diagram.

Q5 [3+2=5 marks]:
(a) Propose *system test* cases for the **`label`** command described in Q3. One example is provided. Design test cases in a way that maximize the efficiency and effectiveness of testing.
Note: Please read both part (a) and (b) before answering part (a) of this question.

Input	Rationale
1. `label bug ui`	Typical, correct use of the command.

(b) Explain one example of how you used equivalence partitions and one example of how you used boundary values when designing test cases listed above.

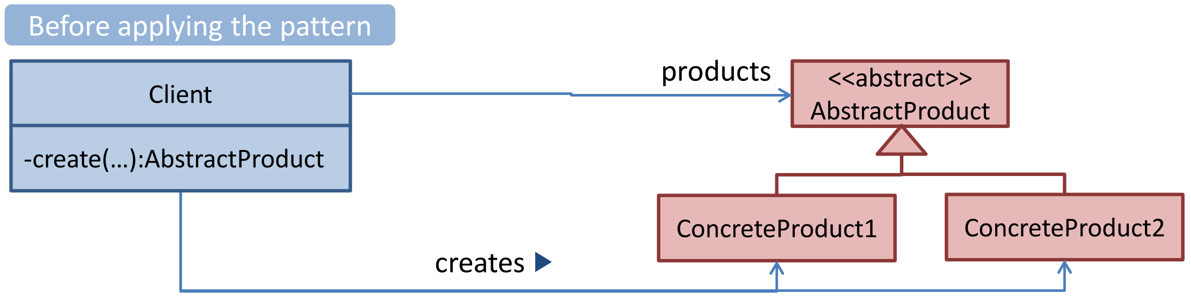
Q6 [5 marks]:

Assume you are already using the *Command pattern* in your HubIssues design. Would the *Simplified Factory pattern* (given below) apply in your design? Justify your answer.

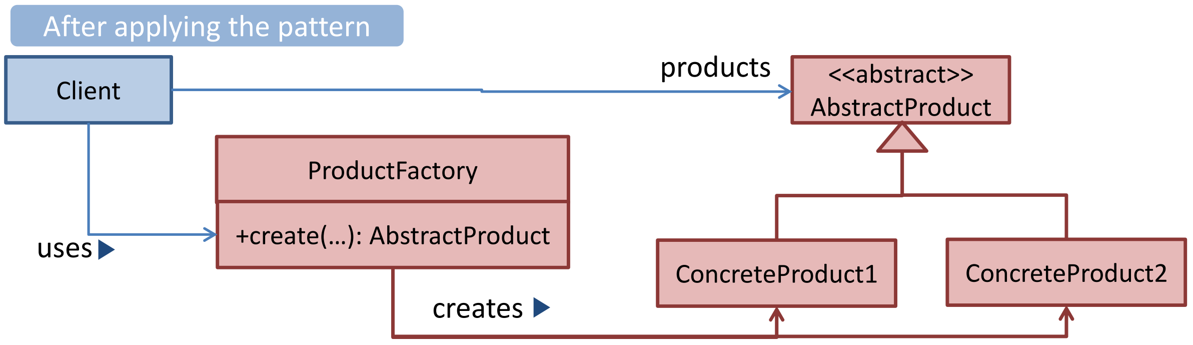
Pattern: Simplified Factory

Context: A class has to create objects of different subclasses. E.g. the `Client` class below needs to create `ConcreteProduct1` objects or `ConcreteProduct2` objects.

Problem: The logic for selecting which subclass to create complicates the class unnecessarily and makes the class unnecessarily dependent on the subclasses in concern.



Solution: Move the 'selection logic' to another class, which then provides a method to create objects.



---Use this page if you need extra space for any of the questions---

--- End of Paper---