Shaun Ng - Project Portfolio for CS2103 - Revision Tool

By: `Team F10-3` Since: `Aug 2019` Licence: `MIT`

1. Overview

RevisionTool is a Desktop application used as a personal tool for revisions and self-quizzes. The user interacts with it using a CLI, and it has a GUI created with JavaFX. It is written in Java and has about 14 kLoC.

2. Summary of contributions

- **Major enhancement**: added **the Restore Command function**
    - What it does: allows the user to restore their current Question Bank to the default question bank.
    - Justification: This feature improves the product rather significantly as users tend to over-add questions and this would allow them to revert their Question Bank to the default bank as a form of "Restart", as most of our default questions are taken from the weekly lecture quiz questions, and hence deemed as important by the professor.
    - Highlights: This enhancement required changes to the existing models and command classes as a warning alert would pop up to seek confirmation from the users before performing the restore command. Hence the implementation was challenging as it required UI, and model changes.
- **Major enhancement**: added **the Auto Complete function**
    - What it does: shows the user a list of possible AutoComplete commands while they are typing, in which they can select, in order to simplify the user's journey.
    - Justification: This feature improves the product significantly because for a CLI, users have to do a lot of typing and this greatly eases the amount of

work they have to do, and hence allow them to reap the maximum benefits from the application.

    - Highlights: This enhancement required in-depth analysis of design alternatives as the implementation required changes to the commandBox which was used in almost every file which had a textBox in it. A significant amount of thought was placed into the implementation as well as I had to formulate what sort of AutoComplete would be most intuitive and value-add to users.
    - Credits: *Took references and suggestions from 2 different sources:* [JavaFX AutoComplete[1]] [Stackoverflow JavaFX Auto-Suggestions[2]]

- **Minor enhancement**:

    - Created the initial UI interface for the startCommand that users use to start the quiz. The different modes such as "start mode/normal", "start mode/arcade", "start mode/custom" were built upon this.

- **Code contributed**:

    - Restore Command[3]
    - Auto Complete[4]
    - Tp Dashboard[5]

[1] https://github.com/AY1920S1-CS2103-F10-3/main/blob/master/src/main/java/seedu/revision/logic/commands/main/RestoreCommand.java
[2] https://stackoverflow.com/questions/36861056/javafx-textfield-auto-suggestions
[3] https://github.com/AY1920S1-CS2103-F10-3/main/blob/master/src/main/java/seedu/revision/logic/commands/main/RestoreCommand.java/
[4] https://github.com/AY1920S1-CS2103-F10-3/main/blob/master/src/main/java/seedu/revision/ui/AutoComplete.java/
[5] {contributedCodeUtl}/

- **Other contributions**:
  - Enhancements to existing features:
    - Name refactoring from AB3 to Revision Tool (Pull requests #154[6])
    - Wrote additional tests for existing features. (Pull requests #154[7])
  - Documentation:
    - Did minor changes in diagrams and content for User Guide: #4[8], #165[9]
  - Community:
    - PRs reviewed (with non-trivial review comments): #99[10], #101[11],#149[12], #153[13],#155[14],#158[15], #163[16]
    - Provided suggestions and comments for other teams in the class as seen from this feedback[17].

3. Contributions to the User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

3.1. AutoComplete Function

Helps you complete your command when you type.

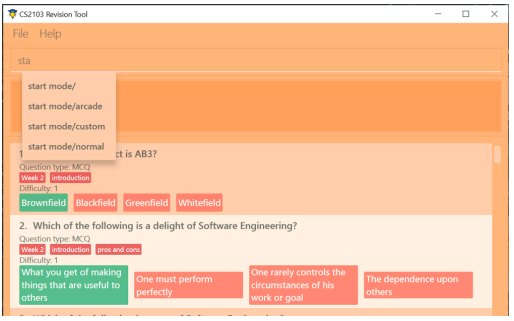
Users will be able to see a list of auto complete options while they are typing. Once the auto complete context menu is shown, users can use the arrow keys to

[6] https://github.com/AY1920S1-CS2103-F10-3/main/pull/154
[7] https://github.com/AY1920S1-CS2103-F10-3/main/pull/154
[8] https://github.com/AY1920S1-CS2103-F10-3/main/pull/4
[9] https://github.com/AY1920S1-CS2103-F10-3/main/pull/165
[10] https://github.com/AY1920S1-CS2103-F10-3/main/pull/99
[11] https://github.com/AY1920S1-CS2103-F10-3/main/pull/101
[12] https://github.com/AY1920S1-CS2103-F10-3/main/pull/149
[13] https://github.com/AY1920S1-CS2103-F10-3/main/pull/153
[14] https://github.com/AY1920S1-CS2103-F10-3/main/pull/155
[15] https://github.com/AY1920S1-CS2103-F10-3/main/pull/158
[16] https://github.com/AY1920S1-CS2103-F10-3/main/pull/163
[17] https://github.com/nus-cs2103-AY1920S1/forum/issues/52

choose the options they want and upon pressing the "ENTER" button, they would be able to select the option.

Examples:

- `User wants to type the "start" command in the command box`
- `He will be able to see a list of dropdown options as shown:`
- `First "ENTER" button will select the option`
- `Second "ENTER" button will execute the option`



3.2. Restoring all entries : `restore`

Clears all current questions from the question bank and restores the default questions that were in the original app.
Users will be prompted if they really want to restore their current question bank as the command is non-reversible. Format: `restore`

AutoComplete: Suggested commands will be shown as you type. Navigate through the dropdown list using the up and down arrow keys and hit **Enter** to complete the command.

4. Contributions to the Developer Guide

Given below are sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.

4.1. Model component

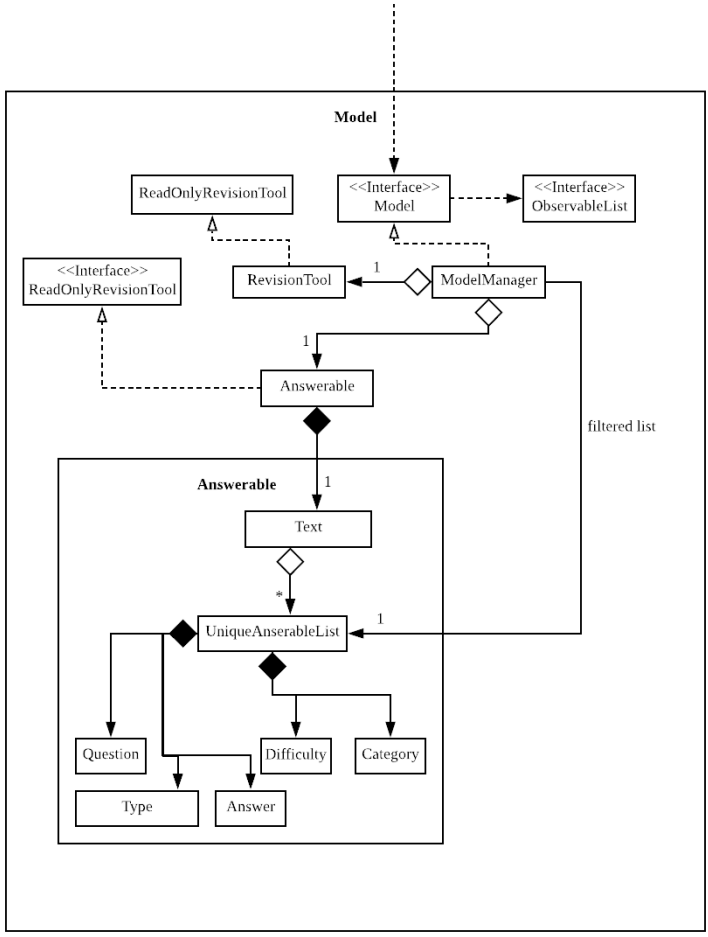


Figure 1. Structure of the Model Component

The `Model`,

- stores a `QuestionBank` object that represents the Question Bank.
- stores the Question Bank data.
- exposes an unmodifiable `ObservableList<Answerable>` that can be 'observed' e.g. the UI can be bound to this list so that the UI automatically updates when the data in the list change.

- does not depend on any of the other three components.

4.2. Storage component

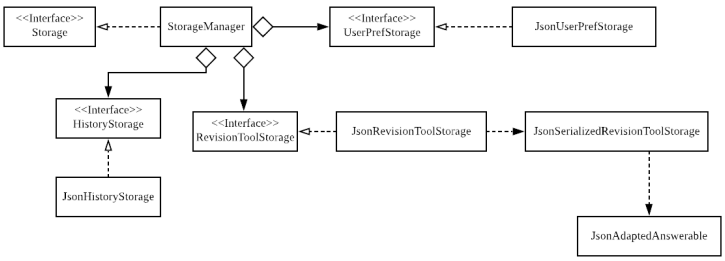


Figure 2. Structure of the Storage Component

The `Storage` component,

- can save `Question Bank` objects in json format and read it back.
- can save the Test Bank data in json format and read it back.

4.3. Restore feature

Implementation

The restore mechanism is facilitated by `RestoreCommand`. It extends `Command` that will read a user command and execute the command result. Additionally, it implements the following operations:

- `#handleRestore()` — Prompts the user with an alert box if he really wishes to execute the restore function.
- `#setRevisionTool()` — Clears the current question bank and reset it with our own default questions.

These operations are exposed in the `Model` interface as `Model#setRevisionTool()` and from `MainWindow` as `#handleRestore()` respectively.

Design Considerations

- When implementing the restore feature, we didn't want users to face a problem if they entered the command accidentally hence the alert popup was implemented, to prompt users if they really want to carry out the command before executing it.
- With this popup, users will now be more cautious when trying to restore and only do so when they really want to reset their revision tool.
- Furthermore, the questions that we included in the default revision tool question bank are questions taken from the lecture quiz and weekly quiz which are most probably deemed important by the professor himself.

Aspect: How Restore executes

- User enters the command "restore".
- Command is taken in and a popup is shown to reconfirm if the user would like to carry out the restore command.
- Upon clicking yes, restore command will be handled.
- Current questions will be deleted and default questions will reset to the revision tool.

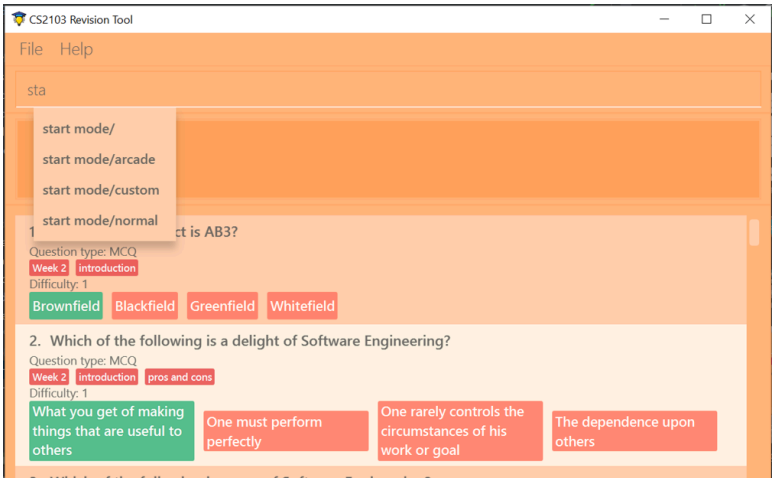
4.4. AutoComplete feature

Implementation

- A set of commands and auto completed text are saved in a set.
- When users type a command on the text box, method `#populatePopup` will be called where the user's command will be matched against our SortedSet.
- If there is a match, a contextMenu showing all possible auto complete text will show up.
- This method is implemented such that the results in the contextMenu will change and show as the user is typing and this would make it more intuitive for users.

Design Considerations

- The main design consideration here would be to have value added auto complete list to pop up.
- How we managed that is to show:
    - The basic command
    - Basic command + possible parse commands where they can easily fill in.



Aspect: How AutoComplete works

- Users wishes to enter an "Add" command `add type/mcq q/What is 1 + 1 y/2 x/1 x/3 x/4 cat/easy diff/1`
- Upon typing either "a", "ad" or even "add", the auto complete context menu will pop up showing possible auto complete list, mainly:
    - add
    - add type/ q/ y/ x/ cat/ diff/
- Upon seeing that, users will be able to select those options or use those as a guideline to complete his commands more intuitively.

A. Non Functional Requirements

1. RevisionTool should work on any mainstream OS as long as it has Java 11 or above installed.
2. RevisionTool be able to hold up to 1000 questions without any significant reduction in performance for typical usage.
3. A user with above slow typing speed for regular English text (i.e. not code, not system admin commands) should be able to accomplish most of the tasks faster using commands than using the mouse.
4. RevisionTool should be able to run without any internet connectivity.
5. RevisionTool does not require any further installation upon downloading the jar file.

B. Glossary

Answerables
A set of question answers, which includes :

- Type: MCQ, True False, Short Answered Question
- Question
- Correct Answers (Can contain multiple answers)
- Wrong Answers (Can contain multiple answers)
- Category
- Difficulty

C. Instructions for Manual Testing

Given below are instructions to test the app manually.

These instructions only provide a starting point for testers to work on; testers are expected to do more *exploratory* testing.

C.1. Adding an answerable

1. Adding a MCQ to the current list

a. Test case: `add type/mcq q/What is 1 + 1 y/2 x/1 x/3 x/4 cat/easy diff/1`
   Expected: new MCQ answerable will be created and appended at the bottom of the list. Details of the the added answerable will be shown at the bottom of the list, and the correct answer will be highlighted in green.

2. Adding a True False to the current list

   a. Test case: `add type/tf q/What is 1 + 1 = 2 y/true cat/easy diff/1`
      Expected: new True False answerable will be created and appended at the bottom of the list. Details of the added answerable will be shown at the bottom of the list, and only the correct answer will be shown and highlighted in green.

3. Adding a Short Answer Question (SAQ) to the current list

   a. Test case: `add type/saq q/What is smaller than 10 but bigger than 7? y/8 y/9 cat/easy diff/1`
      Expected: new SAQ answerable will be created and appended at the bottom of the list. Details of the added answerable will be shown at the bottom of the list and all the correct answers state will be highlighted in green.

4. Adding an Answerable that already exist in the Revision Tool

   a. Test case: `add type/mcq q/What is 1 + 1 y/2 x/1 x/3 x/4 cat/easy diff/1`
      Expected: No new answerable will be added as the question already exist in the Revision Tool. An error message will be thrown, informing users that the answerable already exist in the Revision Tool.