Alex Koh - Project Portfolio

PROJECT: Duke Academy

Overview

Duke Academy is an all-in-one programming practice app targeted for students taking introductory and intermediate programming classes. While most of the user interaction happens through a CLI, the app also comes with a GUI built with JavaFX. Duke Academy was developed by a team of 4 software engineering students, and has a codebase of about 15k LOC, mostly written in Java.

Summary of contributions

- **Major enhancement**: Implemented the Personal Dashboard, which resides under the Home Tab, as well as two relevant user commands that interacts with it, namely `bookmark` `deletebookmark`.
    - What it does: The Personal Dashboard allows the user to keep track of his learning journey in Duke Academy, through three GUI components.

      1) Gamified progress indicator, with both percentage of questions completed and a corresponding skill tier

      2) The list of questions the user is still attempting

      3) The list of questions specially bookmarked by the user

      The `bookmark` command allows the user to add a question to the list of bookmarked questions, while the `deletebookmark` command allows the user to remove any question from that list.
    - Justification: The Personal Dashboard improves user experience significantly. When it comes to programming practice, there are many cases where a student wants to note down a particular question for future reference. For example, a student might find a few questions particularly challenging for him, and he wants to note down these questions for future revision. The `bookmark` command would allow him to conveniently do that, without the traditional need for pen and paper or an external notepad application. This is because the list of bookmarked questions on the Personal Dashboard would update immediately upon the bookmark command. The student is also able to remove any of the questions from the bookmarked list using the deletebookmark command. Also, the inclusion of a gamified progress indicator can actually provide many students with an extra source of motivation and interest towards the arduous and challenging activity of programming.
    - Highlights: The implementation of `bookmark` command and `deletebookmark` command required a deep understanding of the Question class, because Question objects were

required to be either bookmarked or not bookmarked. Also, the Personal Dashboard was developed to be able to support other commands implemented by other developers in the team, namely `attempt` and `submit`. Hence, it required strong understanding of the code written by other developers, and more importantly, effective communication with them to deliver a fully functional dashboard that can accurately support all 4 commands, `bookmark`, `deletebookmark`, `attempt` and `submit`.

- **Minor enhancement**: Added a Program Evaluation Panel that allows the user to view how well his program performed against the pre-defined test cases tied to each question.
- **Code contributed**: [Functional code] [Test code]
- **Other contributions**:
  - Project management:
    - Managed releases `v1.2` - `v1.4` (3 releases) on GitHub, in terms of creation, assignment and tracking of GitHub issues, and also deadline scheduling of the version release.
  - Enhancements to existing features:
    - Modified AB3's `help` command to provide user with a quick overview of all commands used in Duke Academy (Pull request #175)
  - Documentation:
    - Updated README Page to be aligned with the app's full release (#95, #125)
    - Wrote About Us Page (#14, #19, #96, #116)
    - Wrote Contact Us Page (#15, #20, #98)
    - Contributed to the User Guide for Home Tab, `bookmark`, `deletebookmark`, `help` (#174)
    - Contributed to the Developer Guide for Home Tab, `bookmark`, `deletebookmark`, `help` (#79)
  - Tools:
    - Provided assistance with GUI development through expertise with Gluon Scene Builder

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.

Access Dashboard: `dashboard`

Navigates to the **Dashboard** Tab.
The Dashboard allows the user to keep track of his learning journey and progress made in Duke Academy.

Format: `dashboard`

[home] | *home.png*

Adding bookmark: `bookmark`

Bookmarks a specific question.

Format: `bookmark [id]`

- The id of a question can be found next to its title.
- The bookmarked question will appear in the list of bookmarked questions located within your **Personal Dashboard**.

[bookmark] | *bookmark.png*

Removing bookmark: `deletebookmark`

Removes the bookmark from a specific question.

Format: `deletebookmark [id]`

- The id of a question can be found next to its title.
- The question with the bookmark removed will disappear from the list of bookmarked questions located within your **Personal Dashboard**.

Access Dashboard: `dashboard`

Navigates to the **Dashboard** Tab.
The Dashboard allows the user to keep track of his learning journey and progress made in Duke Academy.

Format: `dashboard`

[home] | *home.png*

Access Help Tab: `help`

Navigates to the **Help** Tab.
The **Help** tab contains a quick overview of commands used in Duke Academy, and also a URL to the official User Guide.

Format: `help`

[help] | *help.png*

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.

UI component

[UiClassUmlDiagram] | *UiClassUmlDiagram.png*

Figure 1. Structure of the UI Component

API : `Ui.java`

The UI consists of a `MainWindow` that is made up of 7 parts e.g.`CommandBox`, `ResultDisplay`, `Workspace`, `NotesPage`, `HelpPage`, `Dashboard` and `QuestionsPage`. All these parts, including the `MainWindow`, inherit from the abstract `UiPart` class. Some of the 7 UI parts are also made up of other UI parts. For example, listed on the UML diagram, `QuestionsPage` is made up of `QuestionsListPanel`.

The `UI` component uses the JavaFX UI framework. The layout of these UI parts are defined in matching `.fxml` files that are in the `src/main/resources/view` folder. For example, the layout of the `MainWindow` is specified in `MainWindow.fxml`

The `UI` component,

- Executes user commands using the `Logic` component.
- Listens for changes through the Observer pattern. The UI updates as the `ObservableList` changes.

Bookmark Command

[BookmarkCommandSequenceDiagram] | *BookmarkCommandSequenceDiagram.png*

Figure 2. Sequence Diagram for the execution of a BookmarkCommand instance

The sequence is as follows:

1. User calls execute() on a BookmarkCommand object.
2. The BookmarkCommand object calls getUserSelectedQuestion(), activating an instance of QuestionsLogic.
3. The QuestionsLogic object returns userSelectedQuestion, which is the question the user chose to bookmark.
4. If userSelectedQuestion is already bookmarked, the BookmarkCommand object calls notifyUserNoActionTaken() as a response to the user. Else, the BookmarkCommand object calls bookmarkUserSelectedQuestion(), and then calls notifyUserBookmarkSuccess() as a response to the user.