Pham Tran Tuan Linh - Project Portfolio

PROJECT: FlashSpeed

Overview

FlashSpeed is a text-based flashcard application specifically designed for university students who are learning a foreign language. University students often have hectic schedules. With this in mind, FlashSpeed was created to allow students to be able to study and revise foreign vocabulary on the fly. With a single command, you can start a quick study session on FlashSpeed whenever!

Keeping, flipping, and tracking physical flashcards can be a pain. FlashSpeed enhances the studying process by having virtual flashcards and a smarter review system. Users will be tested more frequently on flashcards that they had trouble memorizing previously.

By using FlashSpeed, you will learn faster and remember for longer!

Summary of contributions

- **Major enhancement 1**: Develop everything related to User Interface (UI) and User Experience (UX): colour scheme, displaying contents, placement of contents.
    - What it does: Allows the user to view and interact with the application
    - Justification: This component is crucial to the application because it visualises the whole application in order for it to function.
    - Highlights: This component requires a lot of research, planning and in-depth analysis of the behaviours of all existing and future features. E.g: When and how the scenes should be switched?; JavaFX and FXML research; When the user updates a content, how to show the updated content immediately?
- **Major enhancement 2**: Modify `LogicManager` and `ModelManager` classes and integrate them with GUI.
    - What it does: `LogicManager` and `ModelManager` facilitate the logic behind every command made by the user and reflect the changes on GUI.
    - Justification: These components act as the brain cells of FlashSpeed to process every

operation.

    - Highlights: These components require in-depth understanding on the behaviours and the logic behind all features.

- **Minor enhancement**:
    - Contribute to the logic behind **Play view** commands and `select` command.
    - Design the application logo

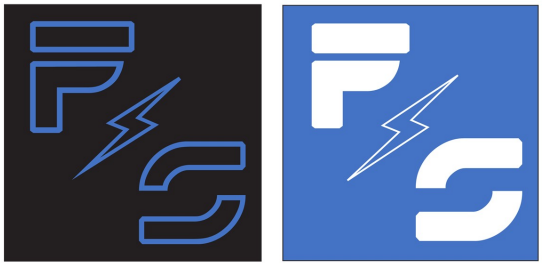

Figure 1. Draft version (left) and Final version (right) of FlashSpeed logo

- **Code contributed**: [Functional & Test code]
- **Other contributions**:
    - Project management:
        - Actively initiate discussions and generate ideas outside meetings.
    - Enhancements to existing features:
        - Display help without Internet connection needed by showing the User Guide directly when using `help` instead of just showing the link: #194 #195
        - Spot and fix bugs in other members' features such as `reset`, `stop`, `exit`, etc.: #271 #175 #193
    - Community:
        - PRs reviewed: #222 #335
        - Reported bugs and provided suggestions for other teams: #1 #2 #3 #4 #5 #6 #7 #8
    - Documentation:
        - Write multiple sections (shown below)
        - Provide all the product screenshots in the User Guide and Developer Guide and update them when there are changes to the current application

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.

Different Views

A view is simply the state or mode of FlashSpeed you are in now. You will be able to identify the view you are in from what you are currently seeing in FlashSpeed.

NOTE	Some commands only work in certain views. Don't worry, this guide will tell you all you need to know!

FlashSpeed can be in one of 3 different views, namely:

- **Library view:** when no deck is selected and no cards are shown

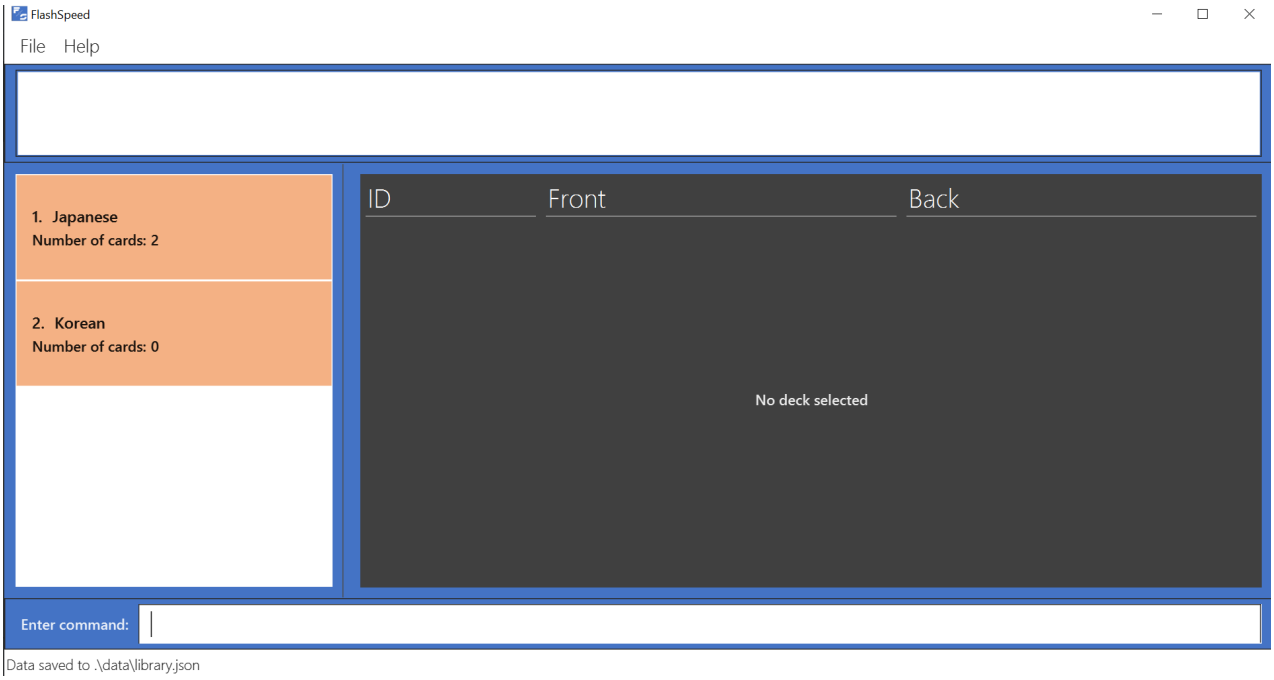

Figure 2. In Library view. No deck is selected.

- **Deck view:** when a deck is selected and its cards are shown

Figure 3. In Deck view. A deck has been selected.

- **Play view:** when in a study session of a deck

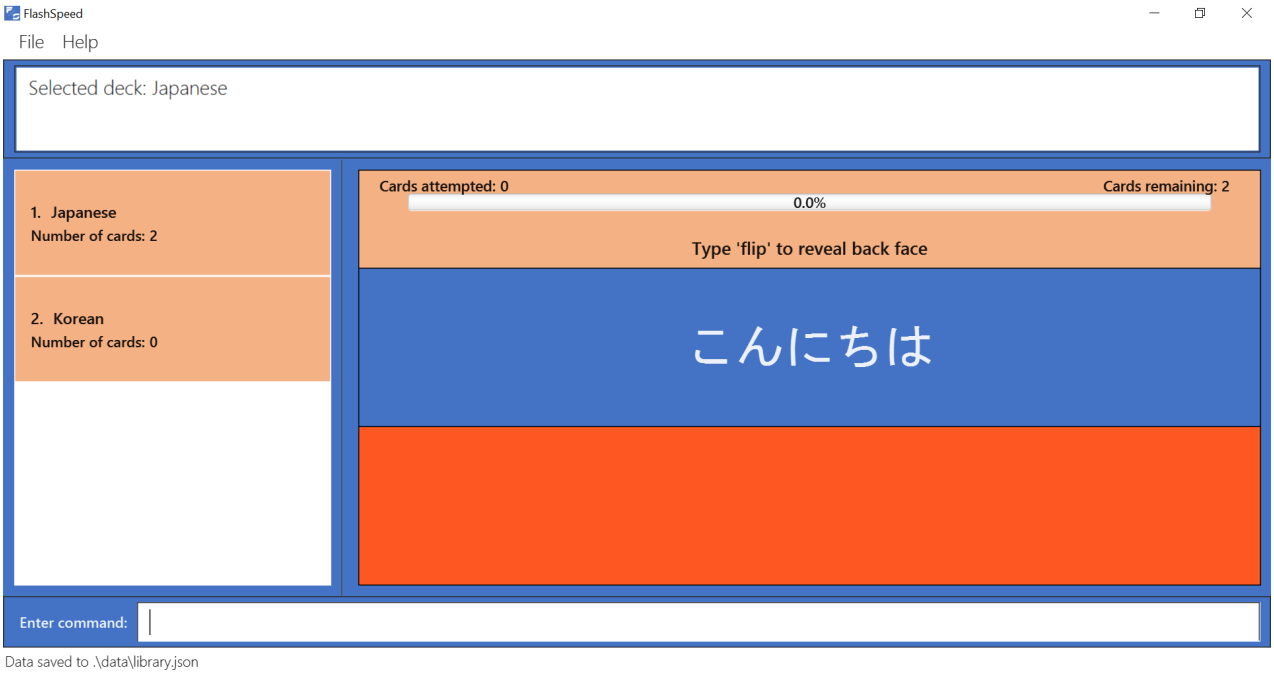

Figure 4. In Play view. A deck is being studied.

FAQ

Q: Does this application require an Internet connection?
A: No, FlashSpeed does not require an Internet connection to use.

Q: What is the maximum length of text I can enter into a flashcard?
A: The flashcard method of studying benefits from succinct and concise flashcards. Even though FlashSpeed does not limit the maximum length of text that can be entered and stored, it will only show the text up to the size of the available display space. Therefore, we recommend keeping any text **under 60 characters**.

Q: How do I save my data?
A: FlashSpeed automatically saves your data whenever you make a change. There is no need to save manually.

Q: Will my data be sent anywhere else or shared with third parties?
A: Your data is stored locally on your own computer. FlashSpeed does not use any Internet connection so no data can be sent to any online servers.

Q: How do I transfer my data to another computer?
A: Simply copy the `data` folder in FlashSpeed's home folder over to the home folder in the other computer.

Q: How do I update FlashSpeed to the latest version when there is an update?
A: You can check for any updates to FlashSpeed here. You can then follow the same instructions as found in [Quick Start].

Q: I am not able to run this application. What can I do?
A: Refer to [Quick Start] for the installation guide. Ensure that your computer has Java 11 installed. FlashSpeed may not be able to run on other versions of Java. Alternatively, you can contact us here for any further help.

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

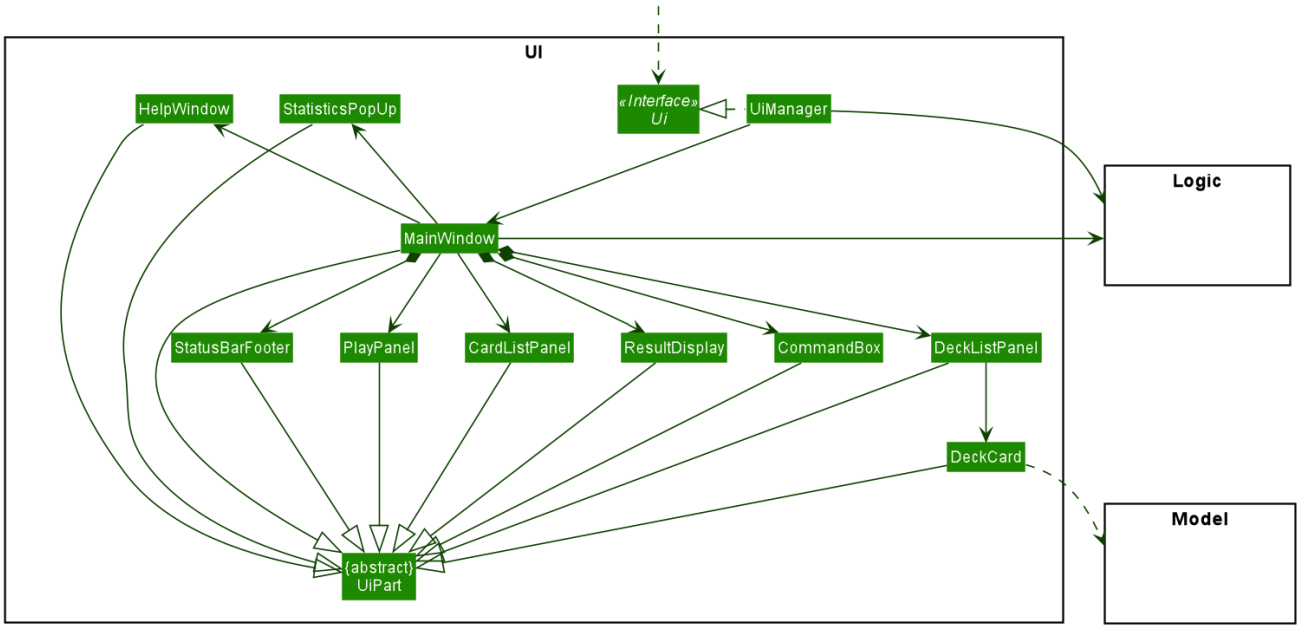

Figure 5. Structure of the UI Component

API : `Ui.java`

The UI consists of a `MainWindow` that is made up of parts e.g.`CommandBox`, `ResultDisplay`, `DeckListPanel`, `CardListPanel`, `StatusBarFooter` etc. All these, including the `MainWindow`, inherit from the abstract `UiPart` class.

The `UI` component uses 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 to `Model` data so that the UI can be updated with the modified data.
- `HelpWindow` will only be shown when executing `help` command.
- `StatisticsPopUp` will only be shown after finishing or stopping a **Play session**.
- Either `CardListPanel` or `PlayPanel` is displayed depending on the current view.

Selecting a Deck

Current Implementation

The `select` command allows user to view the Card content of a Deck.

Accepted syntax: `select INDEX`

This functionality is implemented by getting the Deck based on the index provided. Subsequently,

the Card(s) that belongs to the selected Deck will be displayed on the right panel via a `TableView`.

The validation of the arguments in the `select` command is performed in `SelectDeckCommandParser#parse()`. It ensures that the user has entered a valid index (valid data type and range). This is also used for separation of parsing logic and model management logic.

In `SelectDeckCommandParser#parse()`, the `INDEX` of the selected Deck is extracted from the arguments in the `select` command. The `INDEX` is converted to an Index object. An `SelectCardCommand` object is then constructed with the Index as its parameter.

When `SelectDeckCommand#execute()` is executed a list of currently available Deck is requested from the `ModelManager#getFilteredDeckList()` method. The `ModelManager#selectDeck()` command will be invoked to update the variable that keeps track of the current Deck. After that, `ModelManager#setSelectedDeck()` method will be called to update the UI and display the Deck content on the right panel. Lastly, the name of the selected Deck will be displayed together with the `MESSAGE_SUCCESS` on the `ResultDisplay` panel.

Design Considerations

The UI will have to be constantly updated when we select to view a deck, and other decks might be selected afterward. As a result, an `ObservableValue<Deck>` variable will have to be updated constantly via the `ModelManager#setSelectedDeck()` method. Various event listeners are implemented in the UI classes (e.g `CardListPanel`, `DeckListPanel`) in order to instantly react if there is any changes to the selected deck.

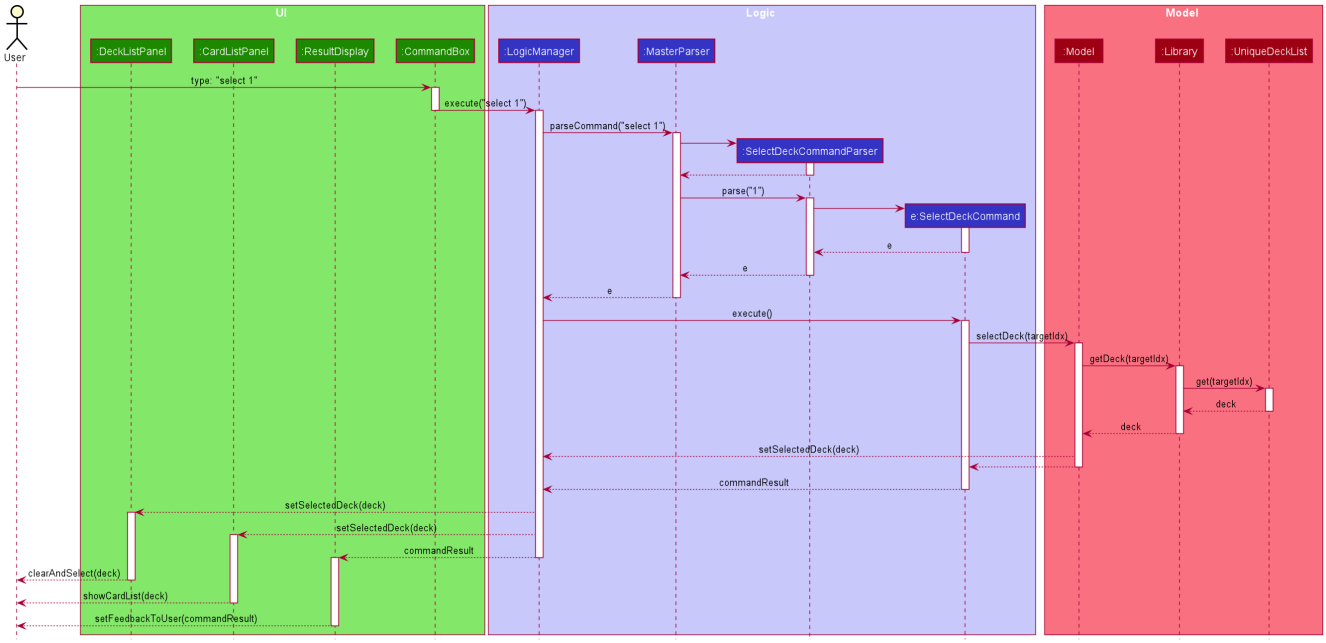

Figure 6. Interactions inside Model, Logic and UI components to reflect UI changes when selecting a deck

View

The state or mode which FlashSpeed is in. There are three different views in FlashSpeed.

- Library View

- Deck View
- Play View

Library View

When no deck is selected and no cards are shown.

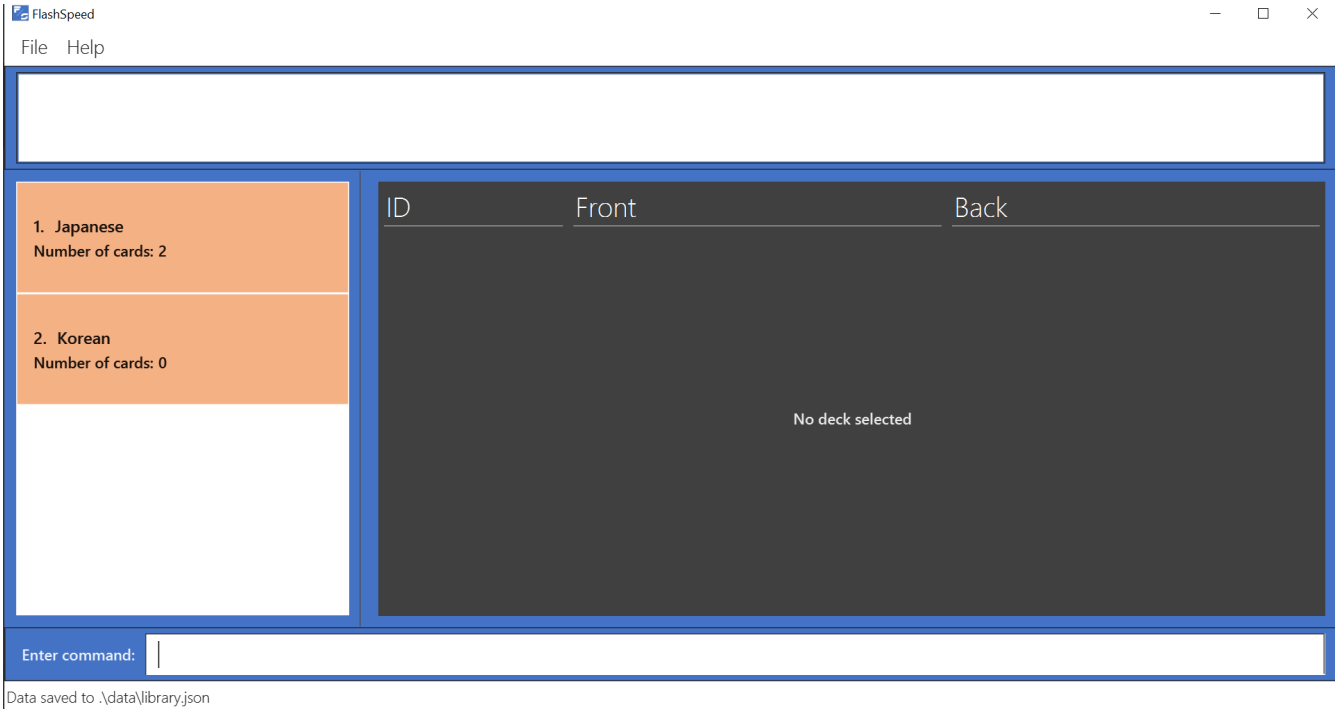

Figure 7. In Library view. No deck is selected.

Deck View

When a deck is selected and its cards are shown.

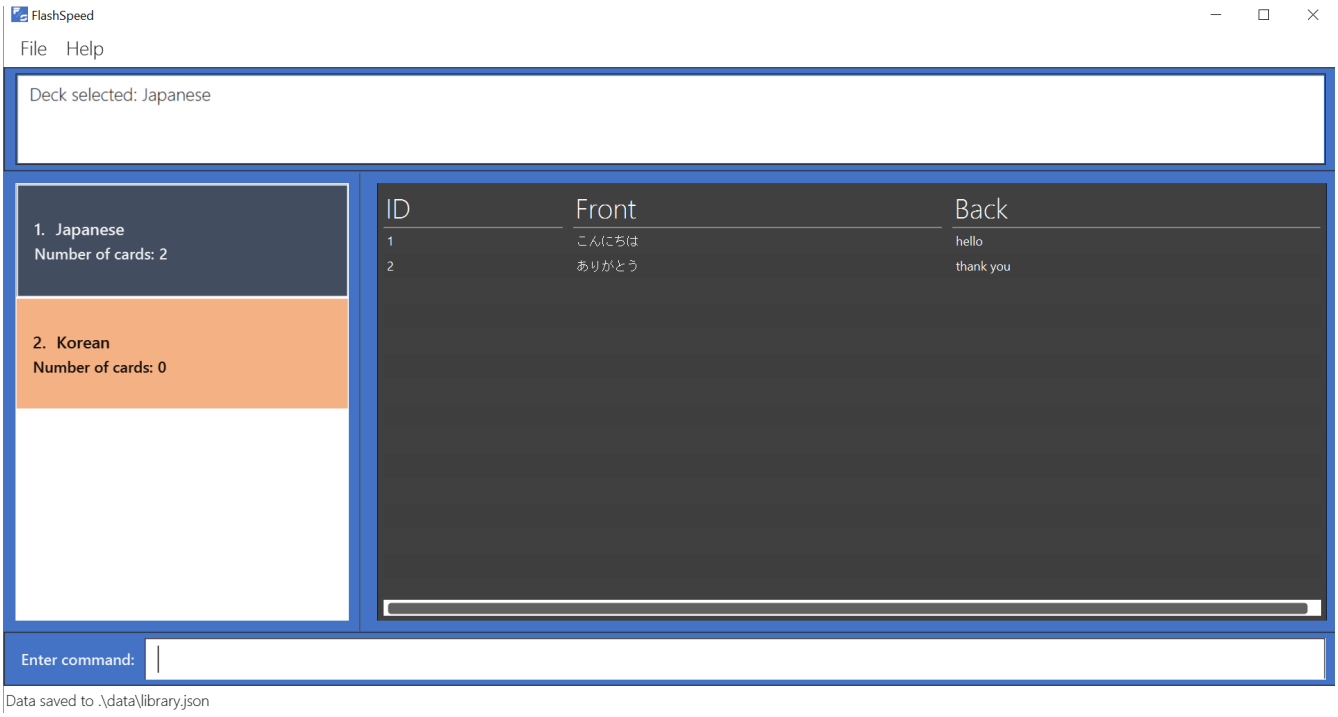

Figure 8. In Deck view. A deck has been selected.

Play View

When in a study session of a deck.

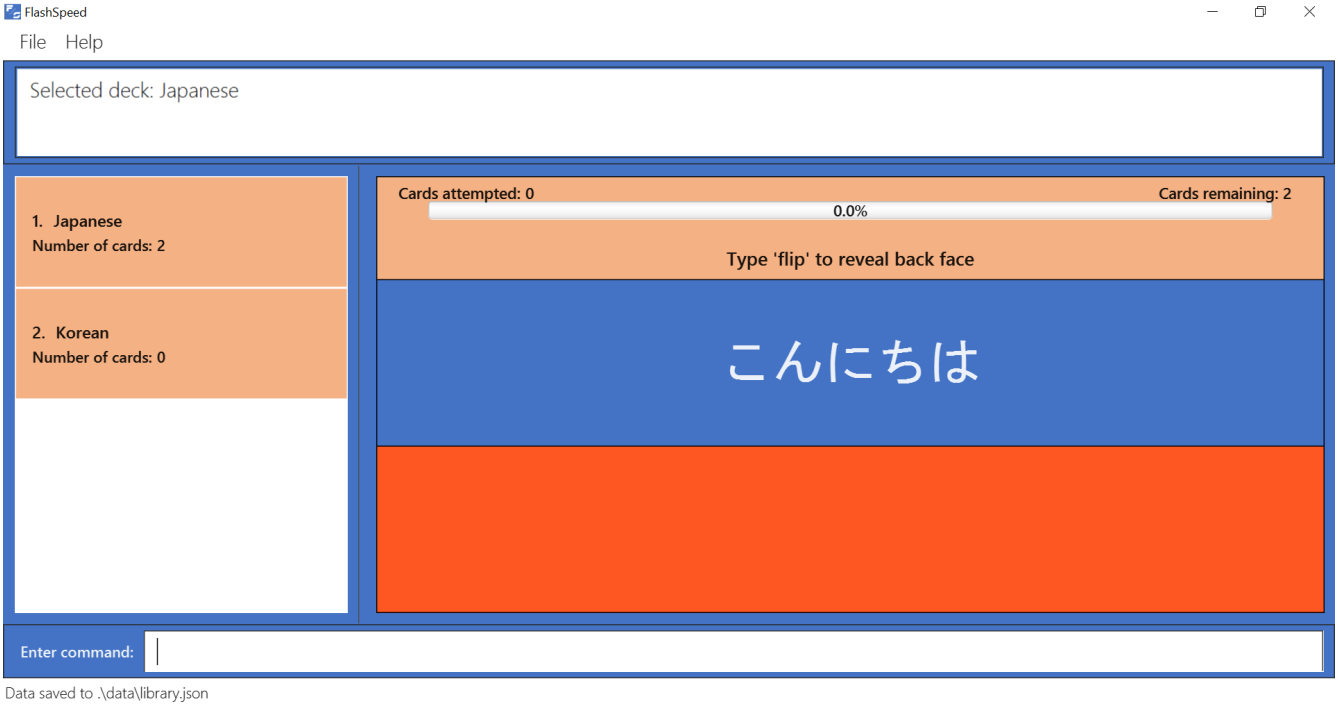

Figure 9. In Play view. A deck is being studied.