Glen Wong Shu Ze — Project Portfolio for JavaCake

Introduction

The purpose of this Project Portfolio is to highlight responsibilities I had and the contributions I made in the project development of the application – JavaCake, in the CS2113T module.

I will be elaborating more on the details of the project in the “About the project” section below.

About the project

The CS2113T has a main group project which started off as being a command line interface (CLI) task manager. However, my team of 3 and I decide to morph this initial idea into a learning application instead, since we felt that there is more potential for interesting and meaningful features.

Our aim is to help incoming CS2113T students transition into this software engineering module by providing them with summarised but essential content and quizzes that can test their understanding.

My role was to design and write the code for the *Logic*, `list`, `overview`, `goto`, `back`, `createnote`, `editnote`, `viewnote`, `deletenote` and `listnote` features.

Note the following key symbols that are used in this Project Portfolio:

ℹ	This symbol indicates important information.
`list`	A grey highlight (called a mark-up) indicates that this is a command that can be inputted into the command line and executed by the application.
`Logic`	Blue text with grey highlight indicates a component, class or object in the architecture of the application.
currentFilePath	Italicised `Consolas` font will be used to denote variable/attribute names used inside a Java Class.

The following sections will elaborate and illustrate some of these features and the relevant sections that I have written in user and developer guides.

Summary of contributions

This section contains a summary of features implemented by me, link to my sample code and other noteworthy contributions to the project. The level of importance of each feature is highlighted by [Major] and [Minor].

- [Major] feature is of high importance such that the application cannot work them.
- [Minor] feature is of low importance such that the feature adds value to the application.

Table 1: Enhancement added in *JavaCake*

Feature	Purpose	User's benefits
`Logic` [Major]	Backend logic for program and ensures scalability of content files.	• Ensures that content can be updated easily by writing new files without writing new code. • Serves as a backend logic for other navigation features such as `list` and `goto`.
`List` [Major]	Displays the main list of content.	• Provides a concise directory of main topics for new user to navigate.
`overview` [Minor]	Displays the expanded list for all the content.	• Allows users to view all the content *JavaCake* has to offer in a single screen.
`goto` [Major]	Transition from parent list to sub list of content.	• Allows easy navigation for users to transition into their desired topics.
`back` `[Major]`	Transition from a sub list of content to its parent list.	• Allows easy navigation for users to return back to their previous view page.
`createnote` `[Minor]`	Creates a note text file for user to store personal note.	• Serves as a convenient tool for user to create, view, edit and delete their personalised notes at any time. • Allows user to consolidate their learning by writing their own summarised notes.
`editnote` `[Minor]`	Edit an existing note file that user created.	
`viewnote` `[Minor]`	View an existing note file that user created.	
`deletenote` `[Minor]`	Delete an existing note file that user created.	
`listnote` `[Minor]`	List all the note files that user created.	• Allows user to see all the notes created.

Code contributed: Please click on this link to see a sample of my code: link

Table 2: Other contributions to *JavaCake*

Name of contribution	**Brief Description**
Project Management	There were 3 releases, from version 1.1 to 1.3. I managed the release of version 1.3 on GitHub. Link
Contributions to project management	1. Setting up of the initial skeleton code for implementing JavaFX and fxml.
Documentation:	1. Designed sequence diagram for `list` feature in Developer Guide 2. Designed structure diagram for *`Logic`* module in Developer Guide
Community	1. Non-trivial review for Pull Request: #115, #150, #153, #158, #165 2. Addressed issues highlighted by peer testers: #130, #135, #139, #145 3. Conducted and led ideas generation for features for milestone 1.2.

The next section will cover my contributions towards the writing of **JavaCake User Guide.**

Contributions to the User Guide

We have updated the **JavaCake User Guide** with instructions for the enhancements that we had added. The section contains an excerpt from our **JavaCake User Guide**, showing additions that I have made for `createnote` and `editnote`.

Creating a personalised note: `createnote`
This command allows you to create your own notes in JavaCake that can be edited and deleted.

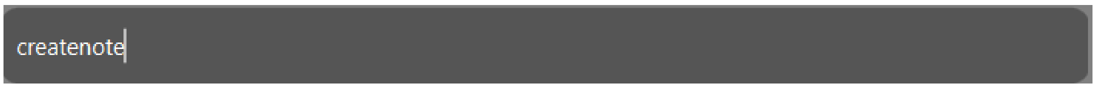
Example of how you can benefit:

- As you are going through the content in *JavaCake,* you wish to summarise your learning by writing down your own notes.
- Instead of using an external software or writing using pen and paper, you can use the built-in feature to create and edit your own notes as you learn which saves time and effort.

If you wish to write your own notes to consolidate you learning, simply use `createnote`. You can also name your notes by `createnote ‘name of note’`.

Steps to `createnote`:

1. Type `createnote` into the command box and press Enter.

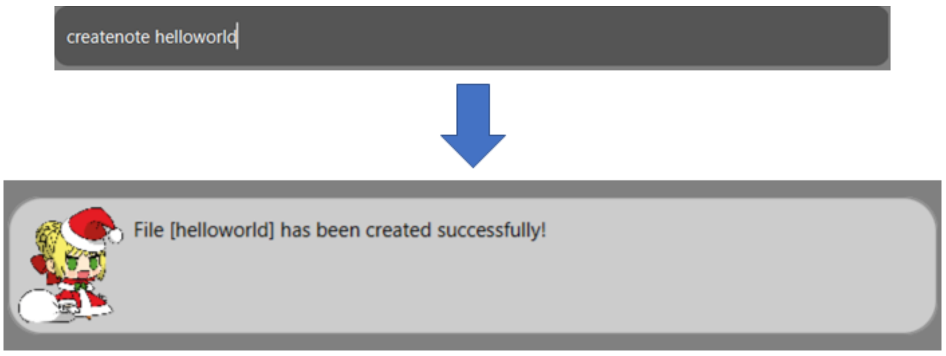


```
createnote
```

Figure 1: Input createnote in the command box

Steps to `createnote ‘name of note’`:

1. Type `createnote ‘name of note’` into the command box and press Enter.
2. For example, if you wish to name your file ‘helloworld’, simply type `createnote ‘helloworld’` into the command box and press Enter as shown in Figure 2.



```
createnote helloworld
```

```
File [helloworld] has been created successfully!
```

Figure 2: Creating a note file ‘helloworld’

Creating notes without specifying file name will be given a default file name of ‘Notes’. Subsequently, a number will be appended to ensure a unique file name.

`createnote` will produce a file - ‘Notes’. Using `createnote` again will produce ‘Notes1’.

JavaCake does not allow and display file names with special characters and limits the length of name to be 20 characters.

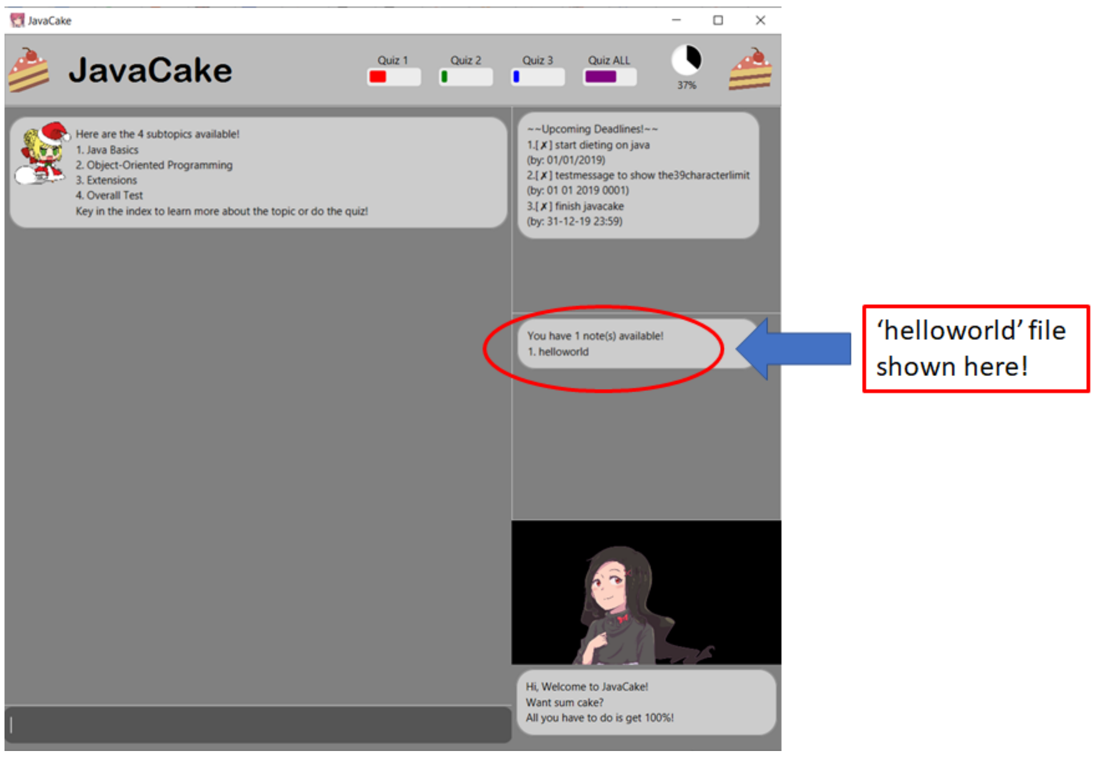
Special characters that not allowed:

/	\n	\r	\t	\0	\f	`	?	*
\\	<	>	|	\	:	.	,	

When creating notes with file names that already exist, JavaCake will notify you to edit the existing note instead.

E.g. trying to createnote ‘helloworld’ note again will prompt user to edit current ‘helloworld’ note.

3. Upon creation, the list of notes can be seen in the side window as seen below.



Here are the 4 subtopics available!
1. Java Basics
2. Object-Oriented Programming
3. Extensions
4. Overall Test
Key in the index to learn more about the topic or do the quiz!

~~Upcoming Deadlines!~~
1.[✗] start dieting on java
(by: 01/01/2019)
2.[✗] testmessage to show the39characterlimit
(by: 01 01 2019 0001)
3.[✗] finish javacake
(by: 31-12-19 23:59)

You have 1 note(s) available!
1. helloworld

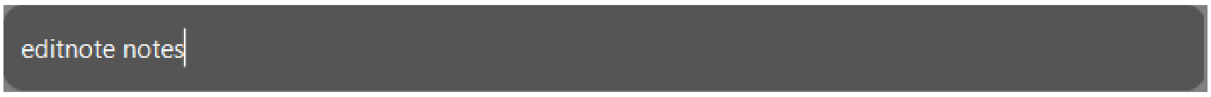
Hi, Welcome to JavaCake!
Want sum cake?
All you have to do is get 100%!

Figure 3: ‘helloworld’ file being stored and displayed at side window

If you wish to write new content in the notes or make changes to a note, simply use `editnote 'name of note'`. You have to make sure that the note you wish to edit exist and you can view all the notes that exist in the side window of *JavaCake* as seen in *Figure 5*.

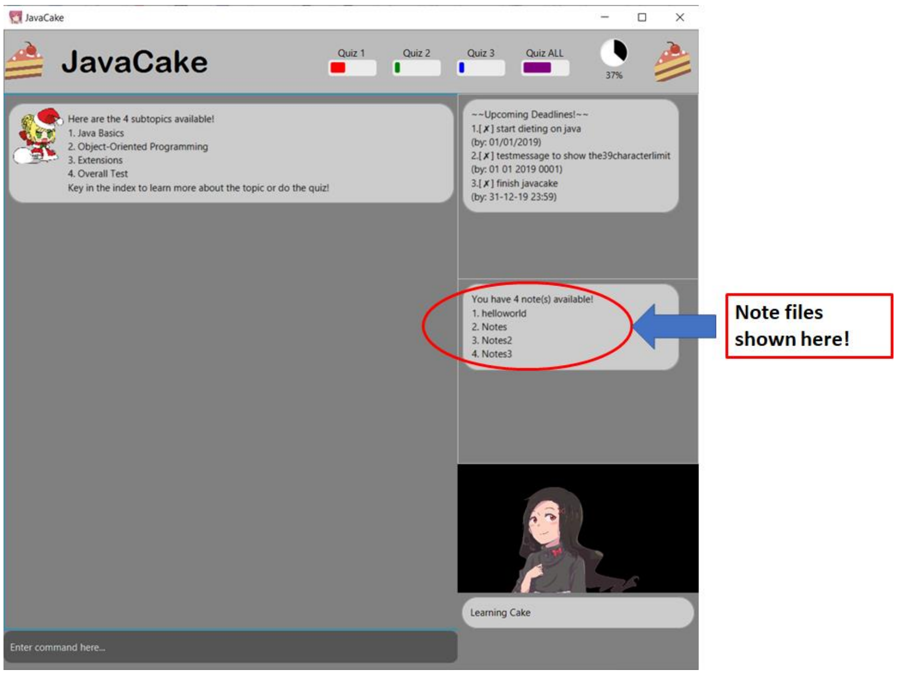
Steps to `editnote`:

1. Type `editnote` followed by the name of the note into the command box and press Enter shown in figure below.



editnote notes

Figure 4: Input editnote notes in the command box



JavaCake

JavaCake

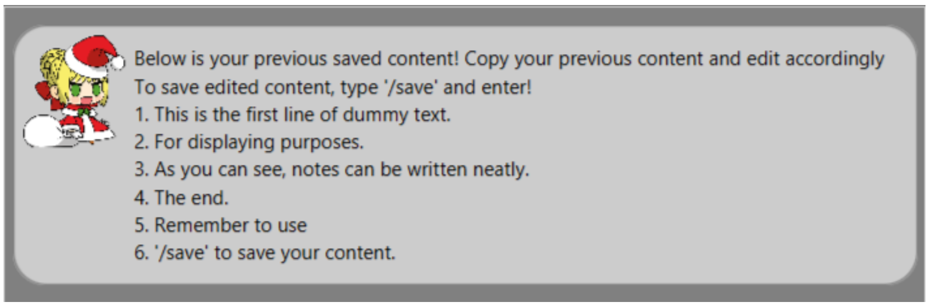
Here are the 4 subtopics available!
1. Java Basics
2. Object-Oriented Programming
3. Extensions
4. Overall Test
Key in the index to learn more about the topic or do the quiz!

~~Upcoming Deadlines!~~
1.[✗] start dieting on java
(by: 01/01/2019)
2.[✗] testmessage to show the39characterlimit
(by: 01 01 2019 0001)
3.[✗] finish javacake
(by: 31-12-19 23:59)

You have 4 note(s) available!
1. helloworld
2. Notes
3. Notes2
4. Notes3

Figure 5: View of the pre-existing notes in JavaCake

2. If you have the particular note contains content, all the content will be displayed for you. You can choose to copy the old content and append the new content before saving as shown below.



Below is your previous saved content! Copy your previous content and edit accordingly
To save edited content, type '/save' and enter!
1. This is the first line of dummy text.
2. For displaying purposes.
3. As you can see, notes can be written neatly.
4. The end.
5. Remember to use
6. '/save' to save your content.

Figure 6: Pre-existing notes stored being displayed

3. After you are done writing new content into the note, save all your content by using `/save`.

Take note that by using editnote, all previous content written will be WIPED by the new content.

If you only wish to read the content without making any changes, you can use the viewnote command instead.

Contributions to the Developer Guide

This section shows my additions to ***JavaCake Developer Guide*** for the `Logic` and `List` feature.

Logic Feature

The browsing of content feature is facilitated by `Logic`, which allows users to dynamically navigate through the content in the content directory without the need to hardcode any of the content in our codebase.

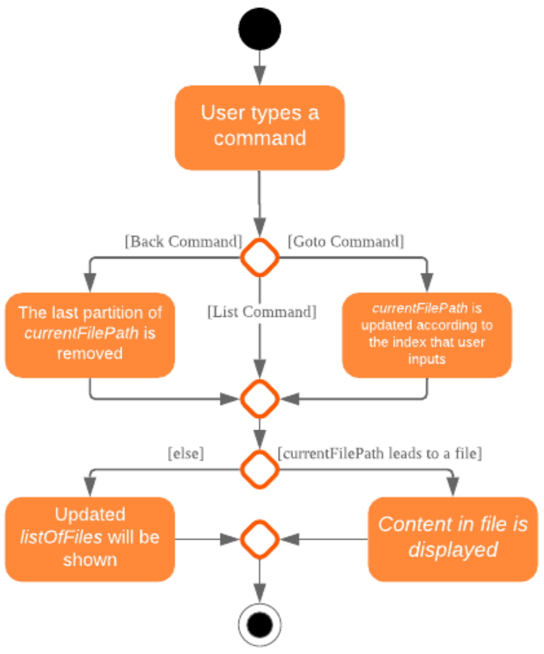

User types a command
[Back Command]
[Goto Command]
[List Command]
The last partition of *currentFilePath* is removed
currentFilePath is updated according to the index that user inputs
[else]
[currentFilePath leads to a file]
Updated *listOfFiles* will be shown
Content in file is displayed

Figure 7: Activity Diagram for Content Browsing in JavaCake, [Activity diagram done by Kishore]

Figure 7 shows the overall activity diagram for content browsing. Two variables of *defaultFilePath* and *currentFilePath* are used, in which *defaultFilePath* stores the file path towards the start of our content directory and *currentFilePath* is used to store the updated file path towards the content requested by the user.

When a command such as `list`, `back` and `goto` that requires the program to traverse through the content directory is called, *currentFilePath* is being updated by concatenating the name of the file to itself.

The files contained in *currentFilePath* can be either a text file or directories. If the file in *currentFilePath* is a text file, the *currentFilePath* will be updated once more to enter the file in order to read the content stored in the file. The content will then be displayed to the user. If the files contained in *currentFilePath* are directories, the name of the directories will be displayed to the user.

The name of the file(s) found in the current directory will be stored in *listOfFiles*, which is a list container for strings.

The following section highlights the various considerations when designing this feature.

Design Considerations

This section contains some of the design considerations when designing the `Logic` feature. It highlights the various alternative solutions to creating the features and the justification of choosing the current implementation.

When designing the `Logic` feature, I had to make the decisions on how best to process the content we have that are stored in the file. The following is a brief summary of my analysis and reasonings.

Table 3: Design considerations for implementing `Logic` feature

Aspect	Alternative 1	Alternative 2
How reading of content works	Dynamically reads the name of content Pros: Very scalable, no hard-coding required. Cons: Slightly harder implementation of reading content. **(Current Choice)** Since content may need to be constantly updated, having a scalable approach would be more appropriate.	Creating individual classes for each subtopic. Pros: Easier to code since it only requires hard-coding. Cons: Not scalable, expanding content files require redoing of codebase.
Data structure to keep track of current location in program	Storing current file path in a string variable Pros: Very scalable, concatenate string variable with new file path. Cons: Slightly harder implementation since the file locations are harder to find and keep track in Java ARchive (JAR) files. **(Current Choice)** Since content may need to be constantly updated, having a scalable approach would be more appropriate.	Using a stack data structure to store current progress in program. Pros: Easy to implement. Cons: Not scalable especially when content files are expanded since every new path location has to be properly indexed.

<u>List feature</u>

When the command entered by the user is `list`, *currentFilePath* will be reset to *defaultFilePath* in which the names of the directories stored within the start of our content file will be displayed. To make it more scalable, we conveniently renamed our directories to have proper indexing.

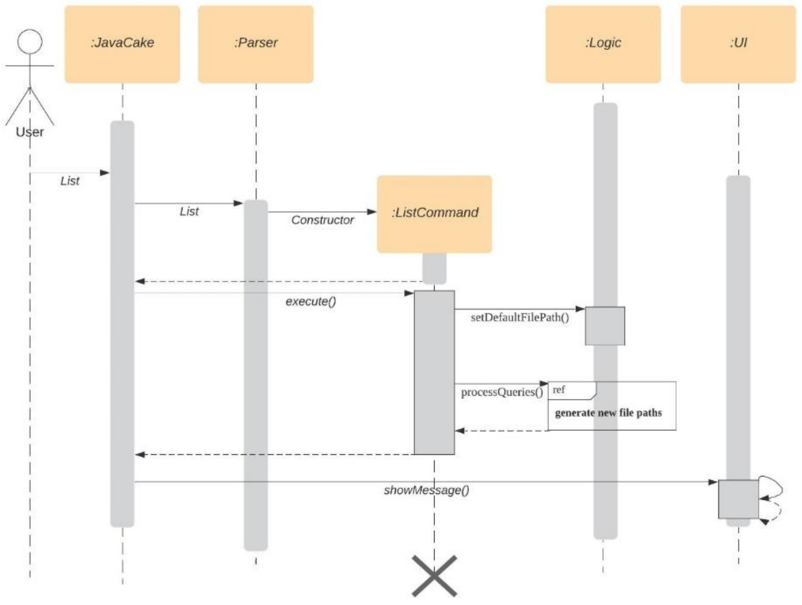


Figure 8: Sequence diagram for ListCommand

`ListCommand` implements the following methods in `Logic` as shown in Figure 8:

- *Logic#setDefaultFilePath()* — Resetting the file path back to default.
- *Logic#processQueries()* — Storing all possible file paths from current directory.

The example below is a usage scenario and how the ListCommand mechanism behaves at each step:

1. User inputs command `list` into *JavaCake*.
2. *JavaCake* passes the list command to `Parser`.
3. `Parser` calls the constructor for `ListCommand`.
4. *JavaCake* calls the *execute()* method in `ListCommand`.
5. When `ListCommand` is executed, `ListCommand` calls the *setDefaultFilePath()* method in `Logic`.
6. *setDefaultFilePath()* in `Logic` will set the static variable *defaultFilePath* in `Logic` to be the root directory of the content files.
7. `ListCommand` also calls *processQueries()* method in `Logic` to instantiate all the names of the directories. The names of the directories will be the names of the sub-topics taught in *JavaCake.*
8. `Logic` class will generate new file path and return the name of content files back to ListCommand.
9. `ListCommand` will return the string of the names of all the content files in main list.
10. *JavaCake* will then send theis string of name to `Ui` for `Ui` to display the content to user.