

# Rusdi Haizim – Project Portfolio for JavaCake

## About the project

My team and I were tasked with enhancing a basic command line interface desktop application that serves as a personal assistant for our Software Engineering project. After much deliberation, we decided to shape it into a Java learning assistant application called **JavaCake**, which allows the user to interact with a Command Line Interface (**CLI**) while still viewing a Graphical User Interface (**GUI**).

There is not only learning material to learn Java, but also interactive quizzes and other features. Through this application, prospective students to the CS2113/T course as well as others curious to learn Java can easily learn the fundamentals of the Java programming language without needing to cross-reference with heavily-documented and daunting websites.

This portfolio serves to document my contributions to the project and the role that I played in developing it.

My role was to design and write the codes for the Profile, deadlines and GUI features. The following sections illustrate these enhancements in further detail. I have also included the relevant sections that I have contributed to the user and developer guides with respect to these enhancements.

Note the following symbols and formatting used in the following sections for easier comprehension:



This symbol indicates important information.



This symbol indicates a tip that the user can follow in order to achieve the best result from the given 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.

`ProgressStack`

Blue text with grey highlight indicates a component, class or object in the architecture of the application.

## Summary of contributions

This section showcases a summary of my coding, documentation, and other useful contributions to the team project.

### Enhancement added:

#### ❖ Addition of a Profile feature.

##### ➤ What it does:

This feature allows the user's progress to be tracked and continuously updated throughout the app's runtime.

##### ➤ Justification:

This is needed since whenever the user exits the app and relaunches it, their progress should have been previously saved and restored.

- Highlights:
 

This enhancement serves as the skeleton to other components my groupmates had implemented such as the quiz questions implemented through [Model](#).
- ❖ Addition of the ability to add deadlines.
  - What it does:
 

The `deadline` command allows the user to add a deadline for them to pace their progress when using the app. The user may also set the deadline as `done`, `snooze` or `delete` their deadlines.
  - Justification:
 

This feature works as a progress tracker for the user, where they can add whatever content or quiz they want to get done, and by what date.
  - Highlights:
 

This enhancement adds a more complete user experience as when the user first launches the app, they will be greeted with a list of the deadlines that they themselves had set.
- ❖ Addition of the GUI for user view the [Model](#).
  - What it does:
 

The GUI allows the user to view the current content or go through the quiz while multitasking other tasks, such as creating notes as well as adding deadlines.
  - Justification:
 

With a GUI, the user can still utilise the app's main features while being aware of their current deadlines/notes, compared to a single-view mode utilised by most CLI apps.
  - Highlights:
 

Users can now multitask when using different functions simultaneously, which enhances the user experience. There are also some aesthetically pleasing features like animated pictures and scrolling text that adds to the experience.

### Code contributed:

Please click these link to see a sample of my code: [\[Functional code\]](#)

### Other contributions:

- ❖ Project management:
  - There were a total of 4 releases, from version 1.1 to 1.4. I managed releases versions 1.1 and 1.2 on GitHub.
- ❖ Enhancements to existing features:
  - Updated the GUI to support dual colour schemes (Light and Dark mode) to allow varied user experience. (Pull request [#91](#))
  - Updated the GUI to include animations for
- ❖ Documentation:
  - Converted the Developer Guide from GoogleDocs into AsciiDoc and uploaded onto GitHub for online perusal of the developer guide.
- ❖ Community:
  - Reviewed Pull Requests (with non-trivial review comments): [#74](#), [#101](#), [#121](#)
- ❖ Tools:
  - Integrated a third party library (Commons IO) to the project ([#106](#))
  - Integrated a new GitHub plugin (Travis) to the team repo ([#14](#))

## Contributions to the User Guide

**JavaCake** contains a myriad of commands that the user can execute. These are some of the common commands that the user may execute:

<code>list</code>	Lists down the topics/quizzes that the user can view/do in the dialog box.
<code>goto</code>	Displays the content/directory as specified by the argument that follows the command.
<code>exit</code>	Displays an exit message before closing the application.
<code>deadline</code>	Adds a new deadline for the user with a specified date.

The following section is an excerpt from our **JavaCake** user guide, showing the additions I have made for the deadline feature.

### Adding a deadline: `deadline`

This command allows the user to add a task which is due by the date specified.

#### What's the purpose?

This serves as a form of progress-tracker, as users can set what topics they want to be done or what quizzes they want to finish and by what time.

Example:

Let's say that you want to set a deadline to complete the topic 'Java Basics' by the 11<sup>th</sup> of November.




The `deadline` feature can only accept certain date formats.

Some common acceptable formats include the following examples:

- 01/01/2019
- 1 january 2019
- 01 01 19 2359

*Refer to Developer Guide's Section 3.7 for comprehensive list of acceptable date formats*

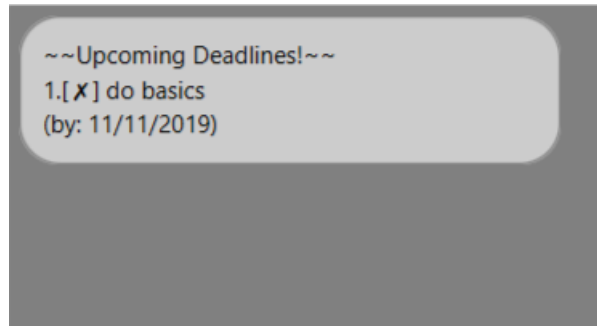
1. Type `deadline do basics /by 11/11/2019` into the command box and press the 'Enter' key inside the command box.



```
deadline do basics /by 11/11/2019
```

*Figure 1: Input deadline in the command box*

2. The newly added deadline will be shown in the following format below, near the top-right side of the app along with the list of deadlines that were previously entered. They are sorted in the order of most recent dates.



*Figure 2: List of deadlines at top-right side of the app*

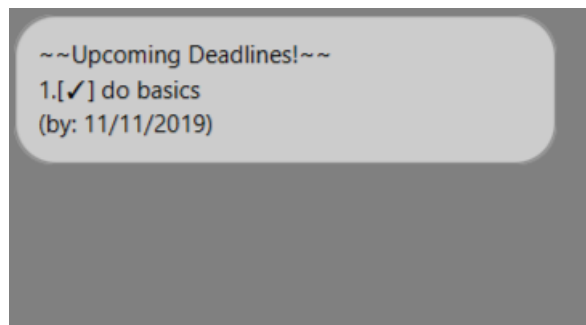
### Meeting a deadline: **done**

This command allows the user to set a deadline as done.

Example:

Let's say that you have just finished browsing through the 'Java Basics' content and want to set your previous deadline as done.

1. Following from the previous input, type **done 1** into the command box and press the 'Enter' key or 'Send' button on the right of the command box.
2. Your list of deadlines will now be updated, with the selected deadline having its 'X' symbol replaced with a '✓' instead, indicating that the deadline has been met.



*Figure 3: Selected deadline is labelled as 'done'*

Refer to Sections 3.9 and 3.10 of the User Guide for similar instructions on how to use the **delete** and **snooze** commands respectively.

## Contributions to the Developer Guide

The following section shows my additions to the *JavaCake* developer guide for the Profile and GUI features.

### Profile feature

This is a backend feature that keeps track of many details of the user such as their username, individual quiz scores and overall quiz scores. It is used in conjunction with `Model` objects like `QuizSession` and `ReviewSession` that my other groupmates have implemented. The following shows some of the common operations:

<code>profile#getUsername()</code>	Gets the username of the user.
<code>profile#resetProfile()</code>	Resets the current user's profile, along with their respective data after calling the <code>reset</code> command.
<code>profile#setIndividualMarks()</code>	Sets the marks of the user for a particular quiz in a particular difficulty (specified in the function parameters).
<code>profile#getIndividualContentMarks()</code>	Gets the marks of the user for a particular quiz in a particular difficulty (specified in the function parameters).
<code>profile#getTotalProgress()</code>	Gets the overall marks of the user for all the quizzes.

These operations are exposed in the `StorageManager` class, which is passed as a parameter when executing various other commands from `Logic`.

### GUI feature

This is a frontend feature which mainly serves to allow the user to multitask while using the application. Tasks such as viewing content/doing the quiz, creating deadlines and creating notes are designed to be mostly mutually exclusive, so as to allow the user to focus more on the content, which is the *JavaCake's* main feature.

Currently, the GUI has a total of 5 viewable interfaces, along with an additional *textfield* for users to input their commands.

View	Description
<code>TopBar</code>	This view contains the title of the app, <i>JavaCake</i> , as well as the individual and total quiz scores at the side.
<code>ContentDialogBox</code>	This view displays the main content and quizzes, which mainly changes according to user commands like <code>list</code> and <code>goto</code> .
<code>DeadlineDialogBox</code>	This view displays the list of deadlines, which is automatically updated after every deadline-related command or after the <code>reset</code> command is called.

NoteDialogBox	This view displays the list of notes, which is automatically updated after every note-related command or after the <code>reset</code> command is called.
AvatarScreen	This view displays our app's avatar. The avatar changes its animation to blink every few seconds as well as when the user gets different quiz scores. There is also an additional dialog box below which displays some motivational quotes for the user.



`TopBar` and `AvatarScreen` are highlighted as such since they themselves are Java classes, as compared to the other views which are implemented using the `DialogBox` class.

## Design Considerations

When designing the `MainWindow#handleUserInput`, I had to make a decision on how best to parse and handle the inputs which cause the user to enter the quiz, review and other modes which block the usual input operation by the user. I also had to deliberate on how to display the user quiz scores at the `TopBar`. The following is a brief summary of my analysis and decision.

Aspect	Alternative 1	Alternative 2
User input for Quiz and Review	<p>Implement a <i>textfield</i> swap and new methods just to handle the <code>QuizSession</code>, <code>ReviewSession</code>, and other commands that block the initial textfield.</p> <ul style="list-style-type: none"> <li>• <b>Pros:</b> Easy to implement</li> <li>• <b>Cons:</b> Will be more memory-intensive and not scalable.</li> </ul>	<p>Keep track of different modes of operation via Boolean variables like <i>isQuiz</i>, <i>isReset</i> and such.</p> <ul style="list-style-type: none"> <li>• <b>Pros:</b> Easy to implement</li> <li>• <b>Cons:</b> Normal commands will usually not be executed when these Boolean variables are set.</li> </ul> <p><b>I decided to proceed with this option</b> because it has a lesser burden on the user's CPU and the execution of normal commands can be resolved when the order of if-else statements are implemented properly.</p>
Display of Quiz Scores in TopBar	<p>Create a fixed number of progress bars and progress wheel objects to display.</p> <ul style="list-style-type: none"> <li>• <b>Pros:</b> Easy to implement</li> <li>• <b>Cons:</b> Not scalable, since if another developer wants to add another quiz, they have to modify the source code directly.</li> </ul> <p><b>(Current Choice)</b></p>	<p>Dynamically create progress bars and progress wheel objects according to the number of Quiz text files found in the resources folder.</p> <ul style="list-style-type: none"> <li>• <b>Pros:</b> Scalable for future implementations of additional quizzes</li> <li>• <b>Cons:</b> Hard to implement and prone to errors if quiz files are not found or properly formatted.</li> </ul>

	For the scope of this project, I have decided to limit the number of quizzes to be only 4.	
--	--	--