Kan Yip Keng’s Projects Portfolio

1. Introduction

Hi, I’m Yip Keng, a Year 2 Computer Science Undergraduate and a part-time tutor at the National University of Singapore. I am passionate about building useful applications which benefits millions of people around the world.

One of my favourite projects is the PDF++, an ambitious file management system aims to replace the current file management system found in most operating systems. This document provides you with a detailed description of the project and my contributions to its creation.

2. Project: PDF++

PDF++ is a desktop application that is meant for anyone who wants or needs a convenient tool that helps them manage all the documents in their computer.

The application has a Graphical User Interface (GUI), built upon the JavaFX framework, that is supplemented with a Command Line Interface (CLI) to accept more specific requests by the User.

The PDF++ offers a wide range of features with potentials for future improvement including:

- **File Manipulation** : Adding, Moving, Merging and Deleting Documents.
- **File Organization** : Labelling tags and Assigning deadlines to Documents.
- **File Extraction** : Finding Documents based on tags and in-text content.
- **File Encryption** : Securing Documents with password encryption.

2.1. Implementation of `deadline` function

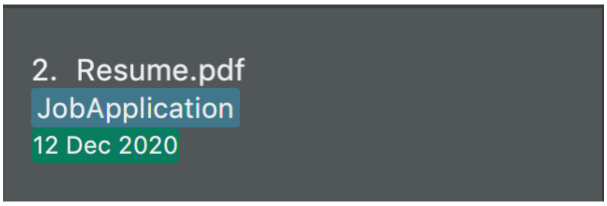
- **What**: This function allows the user to assign a deadline to a document.
- **Justification**: This is an integration of task reminder into our computer files. Users can set a deadline reminder to the documents which have tasks to be met before a specific date.
- **Highlights**: The application labels the deadlines based on the number of days to the document’s due date. Upon launching the application, a list of deadline reminders will be shown in chronological order. This can improve the user’s productivity by keeping track of their upcoming tasks.

2.2. User Guide Contributions

Excerpt from UserGuide: Setting a deadline for a file: `deadline`

Set or remove a deadline for the file, specified by the index of the file that is next to the name of the file in the Files Section.
A file’s deadline is located under its name and has 4 colours to indicate the amount of time you have before it is due.

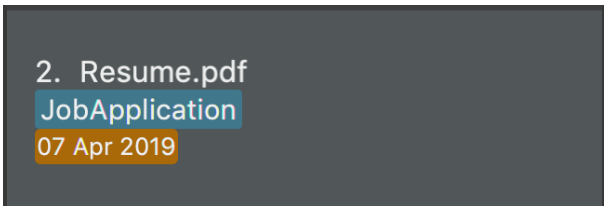
The colour Green, indicates that there are more than `7` days till the deadline is due, as shown below.



2. Resume.pdf
JobApplication
12 Dec 2020

Figure 1. Deadline Colour Green

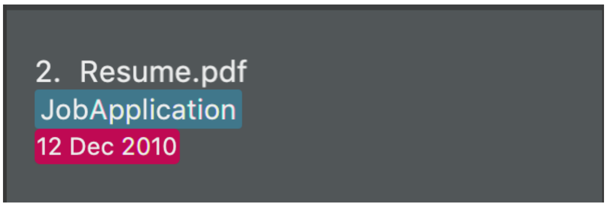
The colour Orange, indicates that you have `7` or fewer days till it is due. As shown below.



2. Resume.pdf
JobApplication
07 Apr 2019

Figure 2. Deadline Colour Orange

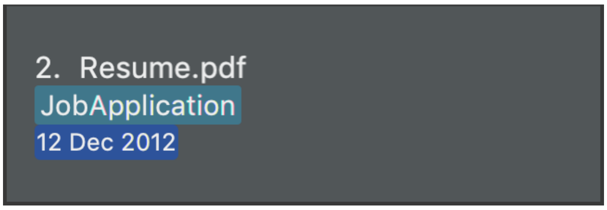
The colour Red, indicates that you have reached or failed to complete the task by the due date, as shown below



2. Resume.pdf
JobApplication
12 Dec 2010

Figure 3. Deadline Colour Red

The colour Blue indicates that you have completed the set task.



2. Resume.pdf
JobApplication
12 Dec 2012

Figure 4. Deadline Colour Blue

All deadlines are also displayed in the Deadlines section of the application.

Format: `deadline INDEX date/DATE` [To Set a Deadline]
Format: `deadline INDEX done` [To Complete a Deadline]
Format: `deadline INDEX remove` [To remove a Deadline]

- `INDEX` refers to the index of the file that you wish to edit.
- `DATE` to the deadline you wish to assign the file.
- `done` is the prefix that tells the application that you have completed the deadline.
- `remove` is the prefix that tells the application that you wish to remove the deadline.

Examples:

- `deadline 1 date/20-02-2019`
- `deadline 1 done`
- `deadline 1 remove`

WARNING The date must be in the format of dd-mm-yyyy. '''

2.3. Developer Guide Contributions

Excerpt From Developer Guide: Deadline feature

Current Implementation

The `deadline` feature is facilitated by both **Deadline**, **DeadlineCommand** and **DeadlineCommandParser** This feature allows you to set or remove deadlines of the file specified by you from PDF++. The deadlines will be recorded and displayed both in the list of files as well as in the information panel for each individual file.

The implementation of the **Deadline** model can be represented in the following class diagram:

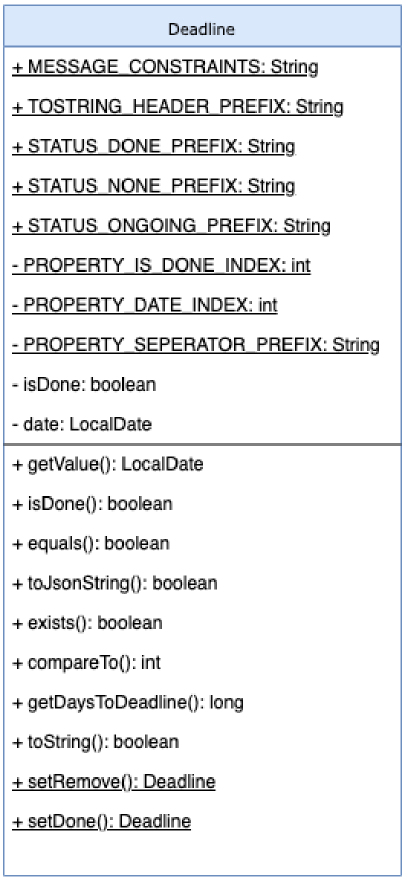

Deadline
+ MESSAGE_CONSTRAINTS: String
+ TOSTRING_HEADER_PREFIX: String
+ STATUS_DONE_PREFIX: String
+ STATUS_NONE_PREFIX: String
+ STATUS_ONGOING_PREFIX: String
- PROPERTY_IS_DONE_INDEX: int
- PROPERTY_DATE_INDEX: int
- PROPERTY_SEPERATOR_PREFIX: String
- isDone: boolean
- date: LocalDate
+ getValue(): LocalDate
+ isDone(): boolean
+ equals(): boolean
+ toJsonString(): boolean
+ exists(): boolean
+ compareTo(): int
+ getDaysToDeadline(): long
+ toString(): boolean
+ setRemove(): Deadline
+ setDone(): Deadline

Figure 5. Deadline Class Diagram

A **Deadline** model has a *Java.time.LocalDate* `date` attribute and a `boolean isDone`. The `date` is the date for the deadline assigned to the file, the `isDone` attribute evaluates to `true` if the deadline is set to be done, `false` if it is not done.

The implementation of the **DeadlineCommand** execution can be summarised in the following activity diagram:

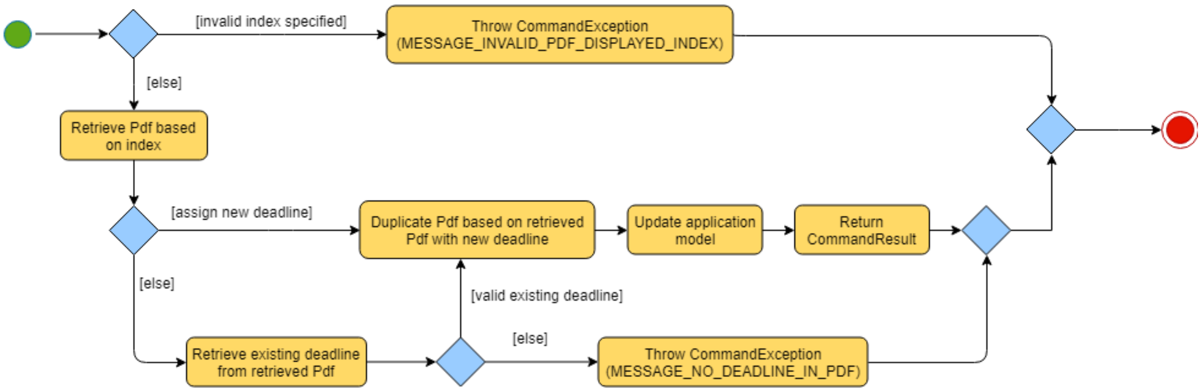

[invalid index specified]
Throw CommandException (MESSAGE_INVALID_PDF_DISPLAYED_INDEX)
[else]
Retrieve Pdf based on index
[assign new deadline]
Duplicate Pdf based on retrieved Pdf with new deadline
Update application model
Return CommandResult
[else]
[valid existing deadline]
Retrieve existing deadline from retrieved Pdf
[else]
Throw CommandException (MESSAGE_NO_DEADLINE_IN_PDF)

Figure 6. Deadline Command Activity Diagram

1. The provided index is checked to be valid i.e. referring to a specific Pdf in the PdfBook.
    a. If the index is invalid, a **CommandException** will be thrown and the execution will be ended.
2. The required Pdf is retrieved from the PdfBook based on the index.
3. A duplicate Pdf of the required Pdf is created.
4. For cases of assigning a new deadline,

   a. The duplicate Pdf is assigned with new deadline attributes.

5. For cases of setting an existing deadline as done or removed,
   a. The existing deadline from the retrieved Pdf is tested to be a valid deadline.
      i. If the existing deadline is a valid deadline, the duplicate Pdf is assigned with new deadline attributes.
      ii. If the existing deadline is not a valid deadline, a **CommandException** will be thrown and the execution will be ended.
6. The duplicate Pdf with new deadline attributes is recorded in the Model and the changes are committed.
7. **CommandResult** is returned upon successful execution.

This sequence diagram demonstrates the interactions involved from start of **DeadlineCommandParser** to end of **DeadlineCommand** execution:

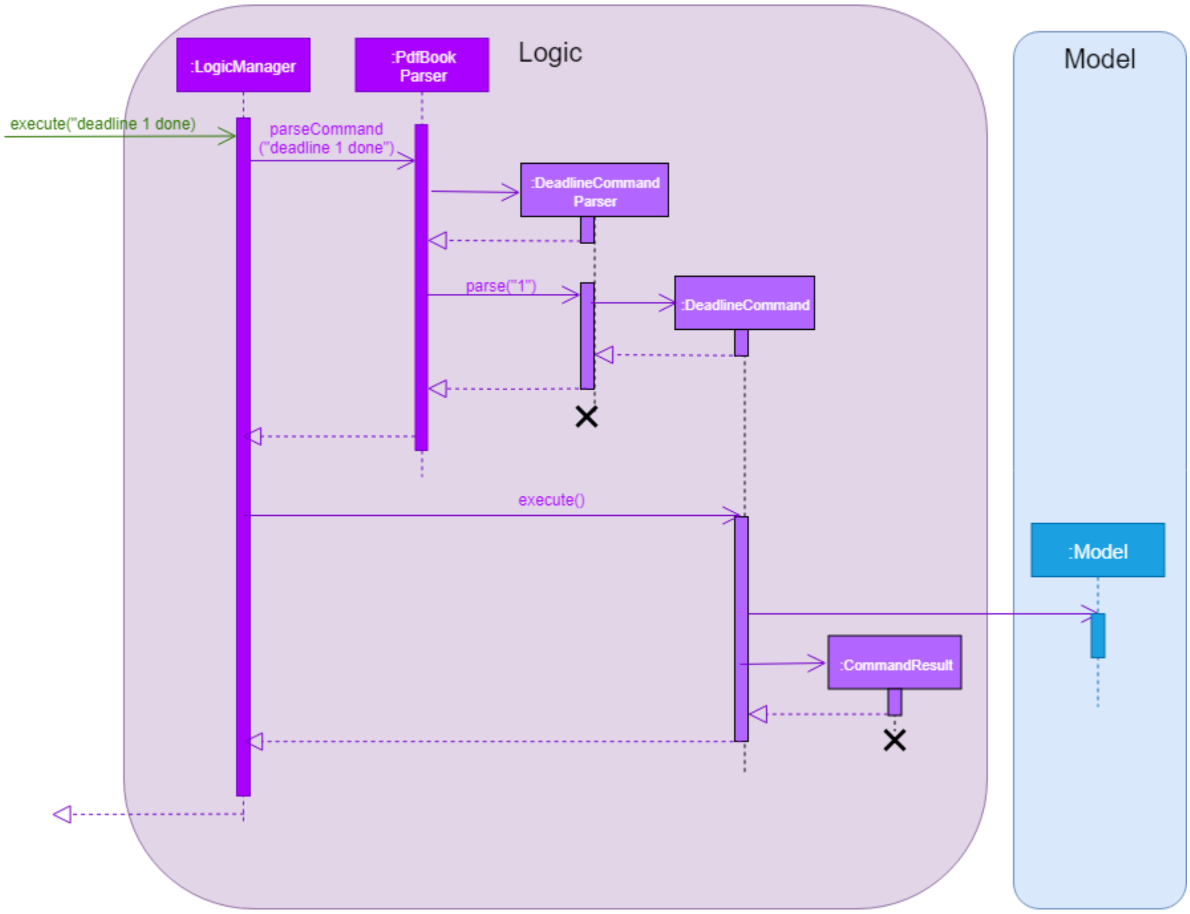


Figure 7. Interactions Inside the Logic Component for the `deadline 1 done` *Command*

TIP After a deadline has been added to the PDF file specified, the date will be color-coded according to days remaining from the current day until the deadline date.

Considerations

There are some discrepancy for the representation of a file without a deadline in the Jackson adapted storage and the Pdf book model. In the Pdf book model, files without a deadline will be assigned with the default deadline whereas the date is set to *LocalDate.MIN*. In the Jackson adapted storage, we simply set the deadline attribute of a file without a deadline as empty. In our previous implementation, we used to assign the default date LocalDate.MIN to the deadline attribute in the Jackson adapted Storage as well, however this might confuse the users if they read the *pdfplusplus.json* and find out the non-existence deadline. Besides, this implementation also makes the displaying process of deadlines through the UI tedious.

In order to comply with the two distinct representation of deadline model, we implemented a default deadline `toString` method that prints the Pdf book model version, and a modified `toJsonString` that prints the Jackson adapted storage version of deadline.

Future Implementation

Our current color-coded deadlines are predefined based on the due date from the current date. Suggested improvement for this area would be providing user-defined color-codes for enhanced user experience.

2.4. Summary of Minor Contributions

- Refactored project from an existing project AddressBook Level-4 to our current application #307
- Implemented the `sort` Command for both **name** and **deadline** mode. #166
- Implemented the `tag` Command. #175
- Reported bugs and fixes to the relevant issues #287
- Updated UI user feedback message for better user experience #296
- Other minor contributions to autotests and the source code