Aw Rui Huan - Project Portfolio

PROJECT: Diary (of the) Undergraduate Kommon Engineer (D.U.K.E.) Manager

About the Project

My team of 5 was tasked to write a Java code to solve problems for our Software Engineering project. We chose to enhance the code into a Java-based electronic diary for engineering students, called DUKE Manager. This application gives engineering students a more efficient platform to record their daily tasks on their devices.

My role was to design and implement the features – Contacts and Detect Duplicates. Contacts consist of 4 other functionalities and Detect Duplicates is meant to automatically run when a new “Todo”, “Deadline” or “Fixed Duration” type of task is added. The following sections shall illustrate each of them in details.

Figure 1 shows a snapshot of our project:

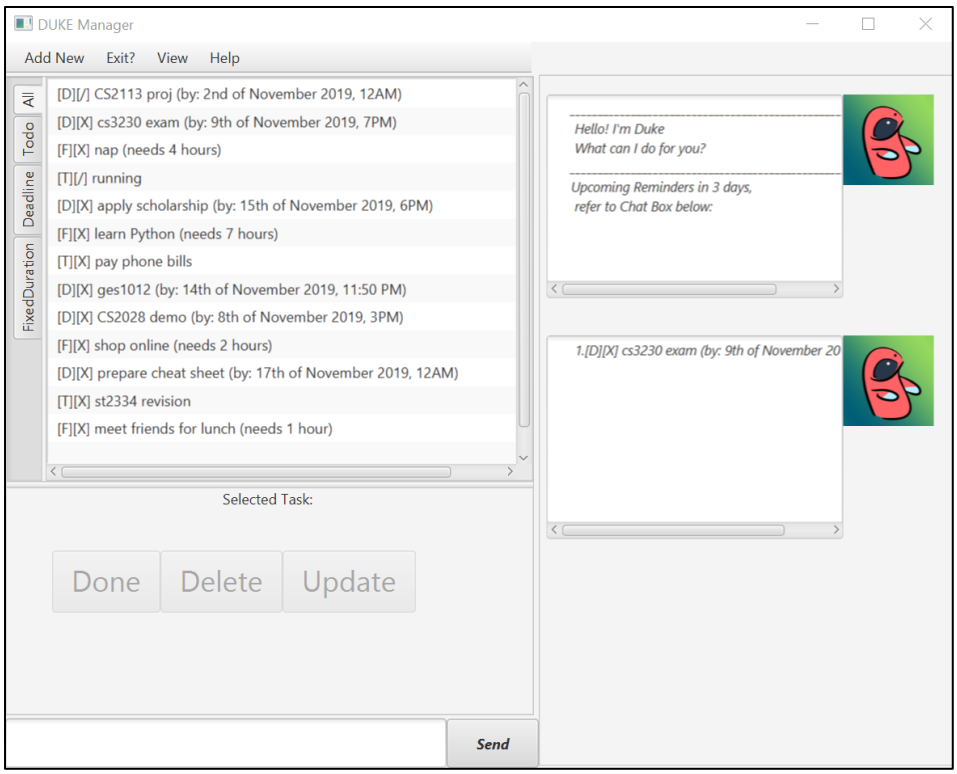

Figure 1. Graphical user interface (GUI) of DUKE Manager.

To note: A text with grey highlight (example) indicates that it is a component, class or object in the program. While a text with yellow highlight (example) indicates it is a command that can be keyed to the command line.

Summary of Contributions

This section shows a summary of the code contributed, documentation and contributions to project management.

Enhancement added: Implementation of contacts management.

- What it does: Allows the user to store contact information such as name, number, email and office. User may also delete, find and view all the contacts after they are stored into DUKE Manager.
- Justification: Even though users may store their contacts in their phones, once they lose their phone or damaged it, all the data, including contacts, may be lost at once. Thus, with this feature, it provides them with an additional platform to safekeep contact information, improving the product remarkably.
- Highlights: This enhancement is an add on to existing commands and was challenging because there is a need to analyse the different ways which users may input their user data into the system so that the program does not point out input errors when it is not. For example, we cannot restrict phone numbers to only include numbers, some international numbers include symbols as well.

Other Features Implemented: Detect Duplicates in Task List

- Detects for duplicated task input so that the user is notified when they enter the same “Todo”, “Deadline” or “FixedDuration” task twice. This is to ensure that when they mark a task as done, they would not get confused when they see another same task in the list which is not marked as done.

Code contributed: [Project Code Dashboard]

- Contacts Related Code: [Contacts] [ContactList] [AddContactsCommand] [ListContactsCommand] [FindContactCommand] [DeleteContactCommand] [ContactStorage]
- Detect Duplicate Related Code: [DuplicateFoundCommand]
- Test Code: [ContactsCommTest] [DetectDuplicateTest]

Other contributions:

- **Project Management**
  - Labelled user stories and placed them in the correct column.
  - Ensured that appropriate labels have been assigned to each issue in the issue tracker.
  - Solved bugs raised by other teams in the issue tracker: 6 closed
- **Community**
  - PR reviewed #209, #168, #141, #125, #119, #42
  - Reported bugs and documentation improvements for another team

- **Documentation**
    - Updated command instructions in User Guide to make it clearer: #187
    - Created Setting Up and architecture components in developer guide.
    - Updated contacts and detect duplicates with the latest functionalities and explanation in both user guide and developer guide.
    - Added examples to ensure consistency across all the different features' description in user guide.
    - Added manual testing for contacts and detect duplicate.

Contributions to the User Guide

This section is an excerpt from our DUKE Manager User Guide which shows my contributions for the Contacts and Detect Duplicate features.

Adding a contact : addcontact

This command allows the user to add a new contact that stores name, number, email and office. If any of the details are unavailable, simply leave a blank space.

Example:
Let's say you have the tendency to drop your phone and are concerned that one day you might damage your phone and it can never power on again, losing precious data especially a professor's contact number. Thus, you decided to store important contact information in DUKE Manager instead.

- addcontact Christian, 89974554, christian@u.nus.edu, E7-8-2
  This adds to the list of contacts.

Format: addcontact <name>, <number>, <email>, <office>

- <name> refers to the name of the contact to be added.
- <number>, <email> and <office> of the contact is to be input in this format.
- <email> must contain an "@" to qualify as an appropriate address.
- For details that are not known, simply omit it.

Listing all contacts : listcontacts

This command shows the user all the contacts that have been saved.

Example:
Let's say you want to see all the contacts saved thus far and double check to see what you entered is accurate.

Format: listcontacts

Finding contacts : findcontact

This command finds and displays relevant contacts stored inside DUKE Manager.

Example:
Let's say you have stored a lot of contact and would like to quickly find a specific person's details,

but you only remembered part of the person's name. Thus, you find by inputting that detail and DUKE Manager will find it for you. This also works for numbers, emails and office location.

- findcontact Tan
  Finds the list of contacts to search for any input with "Tan", regardless of it being upper or lower case.
- fc 91237978
  Shortcut to search through the contact list for this phone number and displays it accordingly.

Format: findcontact <keyword>

- <keyword> is not case sensitive.
- Can search for name, number, email or office.
- Partial words will be matched. Eg. Tan will match Prof Tan.
- Only contacts will be searched.

Deleting specific contacts : deletecontact

This command allows the user to delete a specific contact inside DUKE Manager.

Example:
Let's say the contact details are outdated and you would like to remove it totally from the system since you do not have that person's new contact details.

Format: deletecontact <index>

- Deletes the contact at the specified <index>.
- The index refers to the index number shown in the displayed list contact.
- The index must be between 1 and the total number of contacts in the contact list.

Detect duplicates in task list

This feature finds and alerts the user of duplicated tasks which may be re-entered. This works for Todo, Deadline and FixedDuration only.

Example:
Let's say you have entered a task to do homework but after keying in other tasks, you have forgotten that homework was already entered and thus, you enter the same task again. Without this feature, when you mark the first homework task as done, you may get confused as to why it still appears in the list of undone tasks. Thus, detect duplicates prevents that from occurring.

- todo MA1508E homework
  The same task is already in the list!

Contributions to the Developer Guide

This section is an excerpt from our DUKE Manager Developer Guide which shows my contributions for the Contacts and Detect Duplicate features.

Contacts Feature

The contacts feature is built to allow engineers to manage and store their contacts. They can add, view, find or delete their contact list.

Implementation for contacts feature

The following is an activity diagram to show the execution of all events.

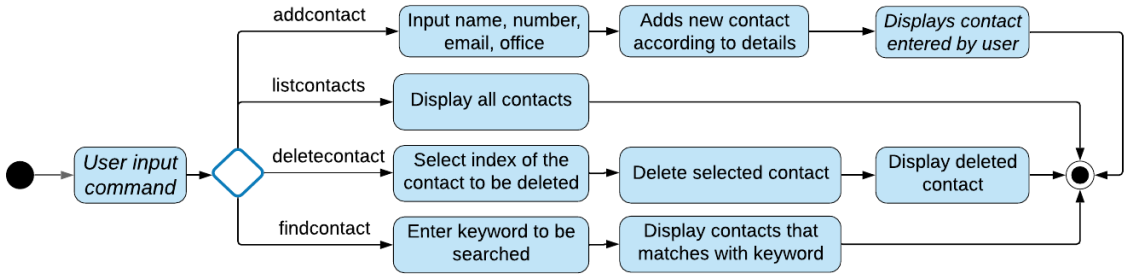

Figure 2. Activity diagram for contacts manager.

1) addcontact()

The addcontacts() feature allows the user to store details about a person's name, number, email and office location.

The following sequence diagram shows how addcontact() operation works:

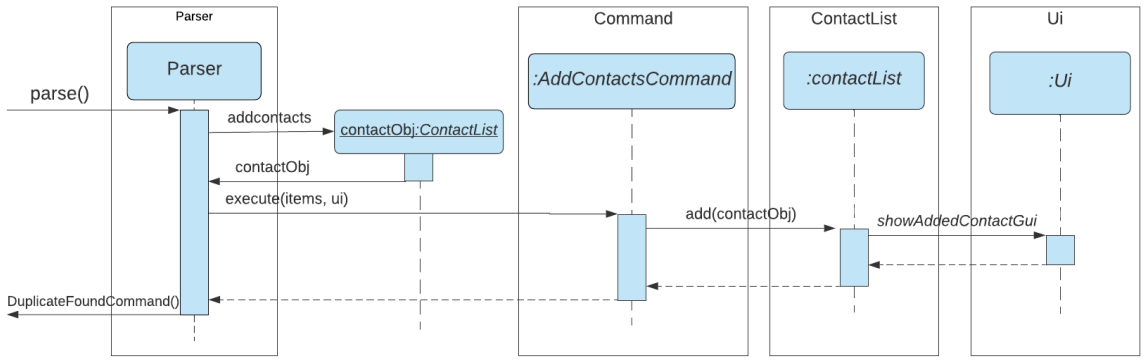

Figure 3. Sequence diagram of addcontact().

Step 1. DUKE Manager is launched for the first time. It attempts to read data stored in the `ContactStorage()` and since the file to store the contacts is not found, DUKE Manager automatically creates an empty file. It then waits for user to command `addcontact` before executing the function.
Step 2. The user executes `addcontact` command and inputs the contact details to be added. The Parser passes the details into `Contacts()` and retrieves `contactObj` which is then passed into `AddContactsCommand()`. `AddContactsCommand()` performs the `executeGui()` method which adds the `contactObj` into a list and shows the user the contact that has just been added.

2) listcontacts()

The listcontacts() feature allows the user to view all stored contacts available in DUKE Manager.

The following sequence diagram shows how the listcontacts() operation works:

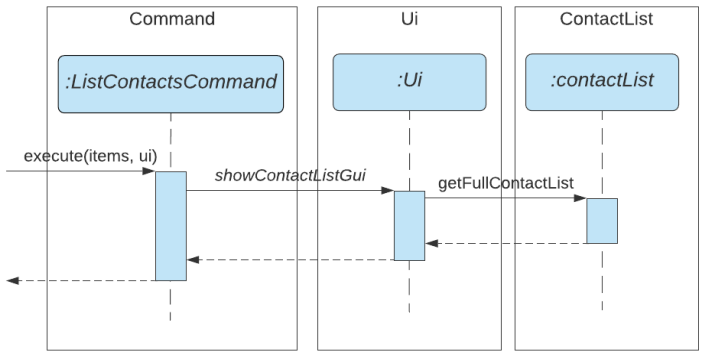

Figure 4. Sequence diagram of listcontacts().

Step 1: DUKE Managers waits for an input by the user. If listcontacts is entered, Parser calls for ListContactsCommand() class which then calls out the executeGui() method.

Step 2: In executeGui(), the showContactList() method in Ui is called to display the contacts list.

3) findcontact()

The findcontact() feature allows the user to quickly find a contact using a keyword, this saves the trouble of having to navigate through the list and finding a specific contact.

The following sequence diagram shows how the findcontact() operation works:

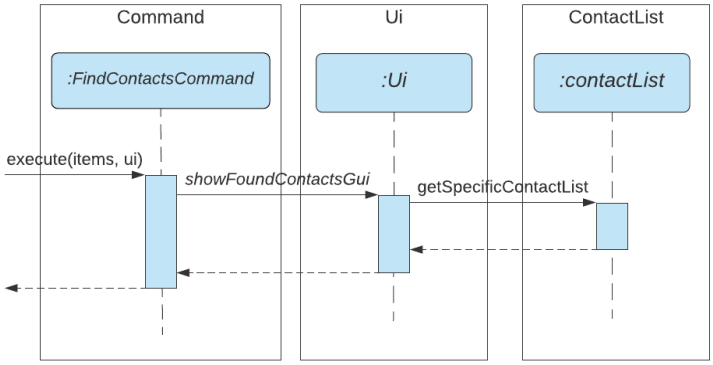

Figure 5. Sequence diagram of findcontact().

Step 1: User inputs a keyword to be searched across all contacts. Parser converts the keyword into lowercase and passes them into FindContactCommand() class which calls out execute() method.

Step 2: The executeGui() method calls out showFoundContacts() method inside the Ui class. In the method, it extracts the details from the contactList getOnlyDetails() method and removes the punctuation and converts them to lowercase. It then proceeds to find keywords that matches inside the list.

Step 3: If there are no contacts found, it will display that no matching tasks were found, else, it will show the user a list of contacts found to be matched.

4) deletecontact()

The deletecontact() feature allows the user to remove a specific contact from its contact list.

The following sequence diagram shows how the deletecontact() operation works:

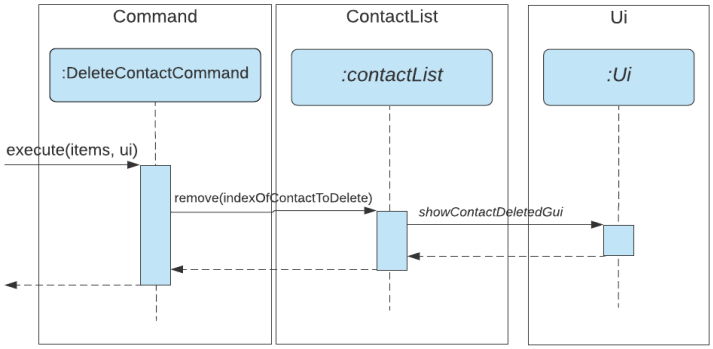

Figure 6. Sequence diagram of deletecontact().

Step 1: User inputs an index to delete a contact. Parser passes the index into DeleteContactCommand() class which calls out executeGui() method.

Step 2: The execute() method calls out getAndDisplay() method inside the contactList class to show the user the contact that has been removed. It then proceeds to remove it by calling remove() in the contactList class. If there are no contacts or an invalid input has been detected, it will display an error message accordingly.

Detect Duplicates

DetectDuplicate() automatically checks if the todo, deadline or fixedduration task added by the user matches with any existing tasks that have already been stored.

Implementation of Detect Duplicates

The following is an activity diagram to show the events when a duplicated task is found.

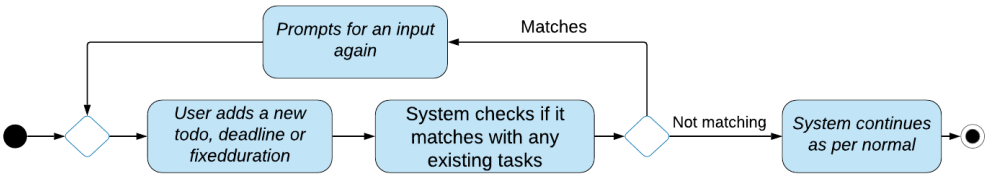

Figure 7. Activity diagram of detect duplicates.

Below is the sequence diagram of Detect Duplicates:

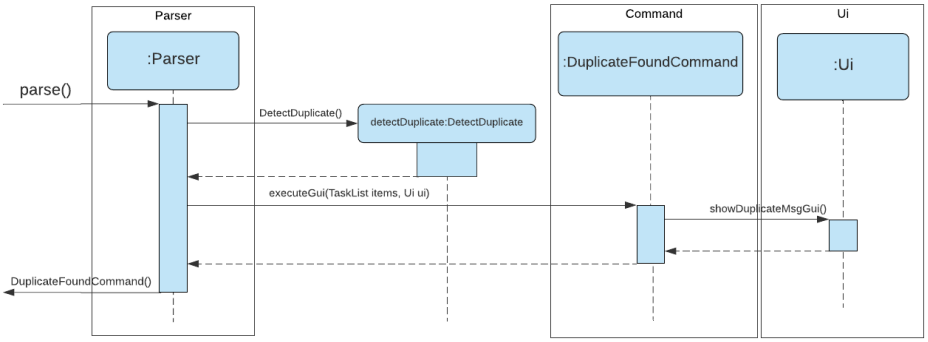

Figure 10. Sequence diagram of detectduplicates().

Step 1. When the user enters a command to add a todo, deadline or fixedduration task, `DetectDuplicate()` is created in the system and `TaskList` is passed into it.
Step 2. `isDuplicate()` checks to see if the input matches with any tasks in the `TaskList` and returns a true value if there is, if not, it will return a false value.
Step 3. If true is sent back to the Parser, `DuplicateFoundCommand()` is called and displays an alert to show the user, preventing them from adding the same task again. If false is sent back, the respective commands proceed as per normal.

Design Considerations

Design Considerations for Contacts

Aspect: How the command for deleting contact is executed

- Alternative 1 (Current Choice): Checks for any error before going to the main code which deletes and displays the contact removed.
    - Pros: This uses the code quality of "make the happy path prominent" and gives a very clear view of the execution path for a successful scenario.
    - Cons: Multiple return statements are needed.
- Alternative 2: Use if-else to check if the conditions are met else it will check for the mistake and display the error message accordingly.
    - Pros: The main execution is at the top of the code.
    - Cons: Method is longer and looks more cluttered due to multiple if-else statements.

Design Considerations for Detect Duplicates

Aspect: Detecting duplicates across the different types of tasks (Types like todo, deadline and fixedduration)

- Alternative 1 (Current Choice): It will detect the description and alert the user even if the type of task differs.
    - Pros: Able to detect if the same task is added to a different type of tasks.
    - Cons: Users are not able to add the same task across the different types of input.

- Alternative 2: Detect duplicate to not activate if the same description is found in the different types of input.
    - Pros: Allows users to store the same description in for example todo and deadline.
    - Cons: Calling the detect duplicate method will look lengthier and users may get confused with their own input as for example, the task in deadline has a time but the same task in todo does not. Making hard to spot the difference in the tasks.