Kenny Ho - Project Portfolio

PROJECT: Delino

Overview

Delino is a desktop application for couriers to manage delivery tasks. The user interacts with it using a CLI, and it has a GUI created with JavaFX. It is written in Java, and has about 10 kLoC.

Summary of contributions

- **Major enhancement**:
  1. **Added `search` feature** (Pull request #187)
     Functionality of feature can be found here and implementation of it can be found here.
     - **What it does**:
       Allows the user to specifically search for keywords that matches any fields of the parcel. The user are also able to narrow down their search range by providing different flags and prefixes.
     - **Justification**:
       This is one of the key feature of Delino as it allows the user to sieve out parcels of their interest from the large parcels list.
     - **Highlights**:
       This enhancement is affected by the existing `insert` and `return` command as it depends heavily on the fields added in parcel. It also affects the future command to be added in if the future command requires to change the properties of a parcel. Implementing this feature requires the thorough understanding of the existing class in AB3 such as the `ArgumentMultimap` and the `NameContainsKeywordsPredicate`. Furthermore, JavaFX class `Predicate` was used to achieve functionality of this feature.
       Moreover, as this features allows user to search in many different possible way, testing the different scenarios are hard. ParametizedTest of JUnit testing was used to make my test code much cleaner.
  2. **Updated `edit` feature** (Pull request #234)
     Functionality of feature can be found here and implementation of it can be found here.
     - **What it does**:
       Allows the user to edit specific fields of a parcel, Order or Return Order.
     - **Justification**:
       This is one of the key feature of Delino as it allows the user to edit any field in Orders or Return Orders. It is especially important as the user might need to reschedule the delivery date due to unforeseen circumstances.
     - **Highlights**:

This enhancement is affected by the design of parcel class. It affects the future implementation of any future features that will alter the parcel design. This feature requires a thorough understanding of the existing code of AB3 such as the `ArgumentMultimap` class.

- **Minor enhancement**:
  1. Changes to all asiicdoc files title to match our app Delino (Pull request #92).
  2. Improve UI of Delino (Pull requests: #209, #221).
  3. Allow user to easily see the time in Delino by adding a real-time clock (Pull requests: #336).
- **Code contributed**: [Functional code & Test code]
- **Other contributions**:
  - Project management:
    1. In charge of both code integration and testing.
    2. Finalised the pre-releases of Delino v1.2 and v1.3 on github.
    3. Together with Jeremy, we set-up the Travis bot in our team repo.
  - Enhancements to existing features:
    1. UI changes Pull requests : #209, #221
    2. Improve search feature functionality to cover Return Order as well as coverage of test case #187, #248.
    3. Improve edit feature functionality to cover Return Order and improve error message for edit feature #234, #305.
  - Documentation:
    1. Update Edit feature, Search feature, Quick start, Glossary and minor documentation bugs of **User Guide** (Pull requests: #91, #92, #128, #135, #177, #195, #227, #316).
    2. Update Use cases 1-6, Product Scope, Image rendering bug, Search feature and Edit feature of **Developer Guide** (Pull requests: #76, #114, #178, #204, #208, #231, #249, #259, #316).
    3. Added own email in ContactUs, changed title of all documents to match Delino name and Update of own PPP (Pull requests : #80, #92, #321).
  - Community:
    - PRs reviewed (with non-trivial review comments): #81, #116, #121, #171, #181, #194, #309, #318
    - Contributed to forum discussions by asking a few help questions #32, #35, #89.
    - Reported bugs and suggestions for other teams in the class #1, #2, #3, #4, #5, #6, #7, #8.
  - Tools:
    - Integrated a third party library (Travis) to the project (#8f78de0)

Contributions to the User Guide

Given below are sections that I have contributed to Delino's User Guide. They showcase my ability to write documentation targeting end-users.

Editing an order : `edit` [Done by - Kenny Ho Si Chong]

In this section, you will be able to find out how to properly use the `edit` command and the relevant examples.
Why will you want to use the `edit` command? If you wish to edit any field of a parcel, the `edit` command will provide you the means to do so.

How to use the Edit command

Here is how you can edit the details of any parcel by following the steps below:

Step 1 : Type in the keyword `edit`
Step 2 : Provide the `FLAG` corresponding to the parcel order type you want to edit
Step 3 : Provide the `INDEX` of the parcel displayed on the screen that you wish to edit
Step 4 : Provide the `ORDER_ATTRIBUTE_PREFIX` coupled with a front slash `/` and the new value you want to change to.
Step 5 : Press `Enter` on your keyboard to see the magic!

NOTE

- In between each step please put a whitespace!
- If you can't see any orders or returns use the `list` command to view existing parcel! If nothing is showing up means you got to add some `order` or `return` parcel and start doing work!

What constitutes a valid Edit command

The syntax for a valid `edit` command can be seen below:

- `edit FLAG INDEX ORDER_ATTRIBUTE_PREFIX/NEW_VALUE [ORDER_ATTRIBUTE_PREFIXES/NEW_VALUE]…`

NOTE

- `edit` is the command word for this feature.
- `FLAG` is to differentiate the different kind of parcel (orders or returns).
- The `INDEX` given is the parcel you will be editing on.
- `ORDER_ATTRIBUTE_PREFIX` is the field of the parcel you want to change
- `NEW_VALUE` is the new value you want to replace the old value with.

These are the possible combinations of the `edit` command:

Table 1. Possible Combinations of Edit command

Scenario	Command	Result
If you want to edit the name of the first return order displayed on returns list	`edit -r 1 n/Xuan En`	The index `1` customer's name of the return order list will be changed to `Xuan En`.
If you want to edit the phone number of the second order displayed on the orders list.	`edit -o 2 p/9999 4444`	The index `2` customer's phone number of the order list will be changed changed to `9999 4444`.
If you want to edit the address of the first order displayed on the orders list.	`edit -o 1 a/Blk 123 Pasir Ris Street 51 #12-21 S510123`	The index `1` customer's address of the order list will be changed to `Blk 123 Pasir Ris Street 51 #12-21 S510123`.
If you want to edit the name, phone number and address of the third return order displayed on the returns list.	`edit -r 3 n/Mr Tan p/0123 4567 a/Blk 141 Yishun st 71 #09-09 S760141`	The index `3` customer's name, phone and address of the return order list will be changed accordingly to the prefix.

WARNING

- The `INDEX` **must be a positive integer**, e.g: 1, 2, 3, …
- The `INDEX` **must be in range** of the number of displayed orders
- Only can be used when there is at least an order displayed.
- The `FLAG` can only be either `-o` or `-r`, please refer to here for more information.

Searching for parcels using their attributes: `search` [Done by Kenny Ho Si Chong]

In this section, you will be able to find out how to properly use the `search` command and the relevant examples.
Why will you want to use the `search` command? If you wish to search for a parcel with specific keywords, the `search` command will provide you the means to do so.

How to use the Search command?

Here is how you can search for any parcel containing the keywords given by the following steps below:

Step 1 : Type in the keyword `search`
Step 2 (optional) : Provide `-o` flag if you want to only search for parcels in the order list. A `-r` flag also can be used to only search for parcels in the return list.

NOTE

If no flag is given in this step, `search` command will be performed on both the order and return list.

Step 3 (optional) : If you wish to only specifically search for keywords in a field, you should

provide the ORDER_ATTRIBUTE_PREFIX coupled with a front slash / and the keyword you want to search for.
Step 4 : Provide any number of alphanumeric words you wish to search for in the parcel.
Step 5 : Press Enter on your keyboard to see the magic!

NOTE

- In between each step please put a whitespace!
- If nothing is displaying means you have no parcel containing the keyword you have given!
- Keyword searches are case-insensitive, e.g: Jeremy matches jErEmY or jeremy or any alphabet casing permutations.

What constitutes a valid Search command?

The syntax for a valid search command can be seen below:

- search [FLAG] KEYWORD [MORE_KEYWORDS]… OR
- search [FLAG] ORDER_ATTRIBUTE_PREFIX/KEYWORD [MORE_KEYWORDS]… [ORDER_ATTRIBUTE_PREFIX/KEYWORD MORE_KEYWORDS]…

NOTE

- search is the command word for this feature.
- FLAG is to differentiate the different kind of parcel (orders or returns).
- The KEYWORD is the word you want to search for in any of the parcel and it is case-insensitive.
- ORDER_ATTRIBUTE_PREFIX is the field of the parcel you want to search for

These are the possible combinations of the search command:

Table 2. Possible Combinations of Search command

Scenario	Command	Result
If you want to search for any return parcel containing the keyword Jeremy or Loh.	search -r Jeremy Loh	Return all return order(s) containing keyword of jeremy, Jeremy Loh or loh or any of the above as long as it appears in any of the parcel field.
If you want to search for any order parcel that contain the transaction id of asj2od3943.	search -o tid/asj2od3943	Return all order(s) with transaction ID of asj2od3943.
If you want to search for any return parcel that contain the phone number of 92039999.	search -r p/92039999	Return all return order(s) with phone number of 92039999

Scenario	Command	Result
If you want to only search for any order parcel that contains either the phone number `92039999`, transaction id of `asj2od3943` or the name `jeremy`.	`search -o p/92039999 tid/asj2od3943 n/jeremy`	Return all order(s) with either phone number of `92039999` or transaction ID of `asj2od3943` or name of `jeremy` or any of the above combinations.
If you want to search for **any** parcel containing the name `Jeremy`.	`search n/Jeremy`	Return all parcel(s) with the name of `Jeremy`

IMPORTANT

- The search is case insensitive. e.g `hans` will match `Hans`
- The sequence of the keywords does not matter. e.g. `Hans Bo` will match `Bo Hans`
- Only full words will be matched e.g. `Han` will not match `Hans`
- Orders matching at least one keyword will be returned (i.e. `OR` search). e.g. `Hans Bo` will return `Hans Gruber`, `Bo Yang`

In this section, you can find out more about the commands supported by Delino (their respective format and example).

If you would like to know more about a specific command, you can view more information by clicking the provided link in the table below.

Table 3. Command Summary

Command	Format	Example
Insert	`insert tid/TRANSACTION_ID n/CUSTOMER_NAME a/ADDRESS p/PHONE_NUMBER e/EMAIL dts/DELIVERY_DATE_&_TIME w/WAREHOUSE_LOCATION cod/CASH_ON_DELIVERY [c/COMMENTS_BY_CUSTOMER] [type/TYPE_OF_ITEM]`	`insert tid/0123456789 n/Eng Xuan En a/Tampines St 84 Blk 877 S520877 #01-123 p/87654321 e/xuanen@example.com dts/2020-02-20 1300 w/Yishun industry cod/$4.50 c/please knock the door three times :D type/heavy`

Clear	`clear [FLAG]`	`clear`
		`clear -f`
		`clear -r`
		`clear -f -r`
		`clear -o`
		`clear -o -f`

Delete	`delete FLAG INDEX`	`delete -o 2`
Delivered	`delivered FLAG INDEX`	`delivered -r 2` `delivered -o 1`

Command	Format	Example

Edit	`edit FLAG INDEX ORDER_ATTRIBUTE_PREFIX/VALUE`	`edit -r 2 n/Xuan En`
		`edit -o 2 p/9999 4444`
		`edit -o 1 a/Blk 123 Pasir Ris Street 51 #12-21 S510123`
		`edit -r 3 n/Mr Tan p/0123 4567 a/Blk 141 Yishun st 71 #09-09 S760141`

Exit	`exit`	`exit`

Search	`search [FLAG] ORDER_ATTRIBUTE_PREFIX/KEYWORD [MORE_KEYWORDS]… [ORDER_ATTRIBUTE_PREFIX/KEYWORD MORE_KEYWORDS]…`	`search -r tid/ac1e345x7s`
		`search -r Jeremy Loh`
		`search -o tid/asj2od3943`
		`search -r p/92039999`
		`search -o p/92039999 tid/asj2od3943 n/jeremy`

Help	`help`	`help`
Show	`show`	`show`
Import	`import FILE_NAME`	`import orders.csv`

List	`list [DONE_STATUS]`	`list`
		`list done`
		`list undone`

Return	`return tid/TRANSACTION_ID n/CUSTOMER_NAME a/ADDRESS p/PHONE_NUMBER e/EMAIL rts/RETURN_DATE_&_TIME w/WAREHOUSE_LOCATION c/COMMENTS_BY_CUSTOMER type/TYPE_OF_ITEM`	`return tid/ac17s2a n/BOBBY TAN a/123 Delta Road #03-333, Singapore 123456 p/91230456 rts/12-12-2020 1301 w/Jurong Warehouse c/NIL type/glass`
		`return tid/ac17s2a`
		`return tid/b1230512 n/Aaron Teo a/256 Alpha Road #03-222, Singapore 123567 p/91230456 e/aaron@example.com rts/12-12-2020 1400 w/Jurong Warehouse c/Leave it at the lobby type/metal`

Nearby	`nearby [FLAG] POSTAL_SECTOR`
	OR
	`nearby [FLAG] AREA`
		`nearby east`
		`nearby -o 14`
		`nearby -r north`

Contributions to the Developer Guide

Given below are sections that I have contributed to Delino's Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.

Edit Feature

In this section, the functionality of the `edit` feature, the expected execution path, the structure of the `EditCommand` class and the interactions between objects with the `EditCommand` object will be discussed.

What is Edit Command

The `edit` feature was implemented as `EditCommand` in the `Logic` package.

`edit` feature format : `edit INDEX FLAG ORDER_ATTRIBUTE_PREFIX/NEW_VALUE [ORDER_ATTRIBUTE_PREFIX /NEW_VALUE]`

The `edit` feature allows the user to edit any field except delivery status of the order or the return order. However, user must provide a `FLAG` and `INDEX`.
`FLAG` to indicate which parcel type to edit; `-o` and `-r FLAG` to represent Order or Return Order respectively.
`INDEX` to indicate which parcel the user wants to edit.
The list of the different parcel fields are listed in Appendix E: Glossary.

NOTE	This feature allows user to edit more than one field within a command.

IMPORTANT	**Limitation** • Editing the delivery/return time is that the updated delivery date or return date must not be in the past. • There must be an order first for `edit` command to work.

Execution paths of Edit Command

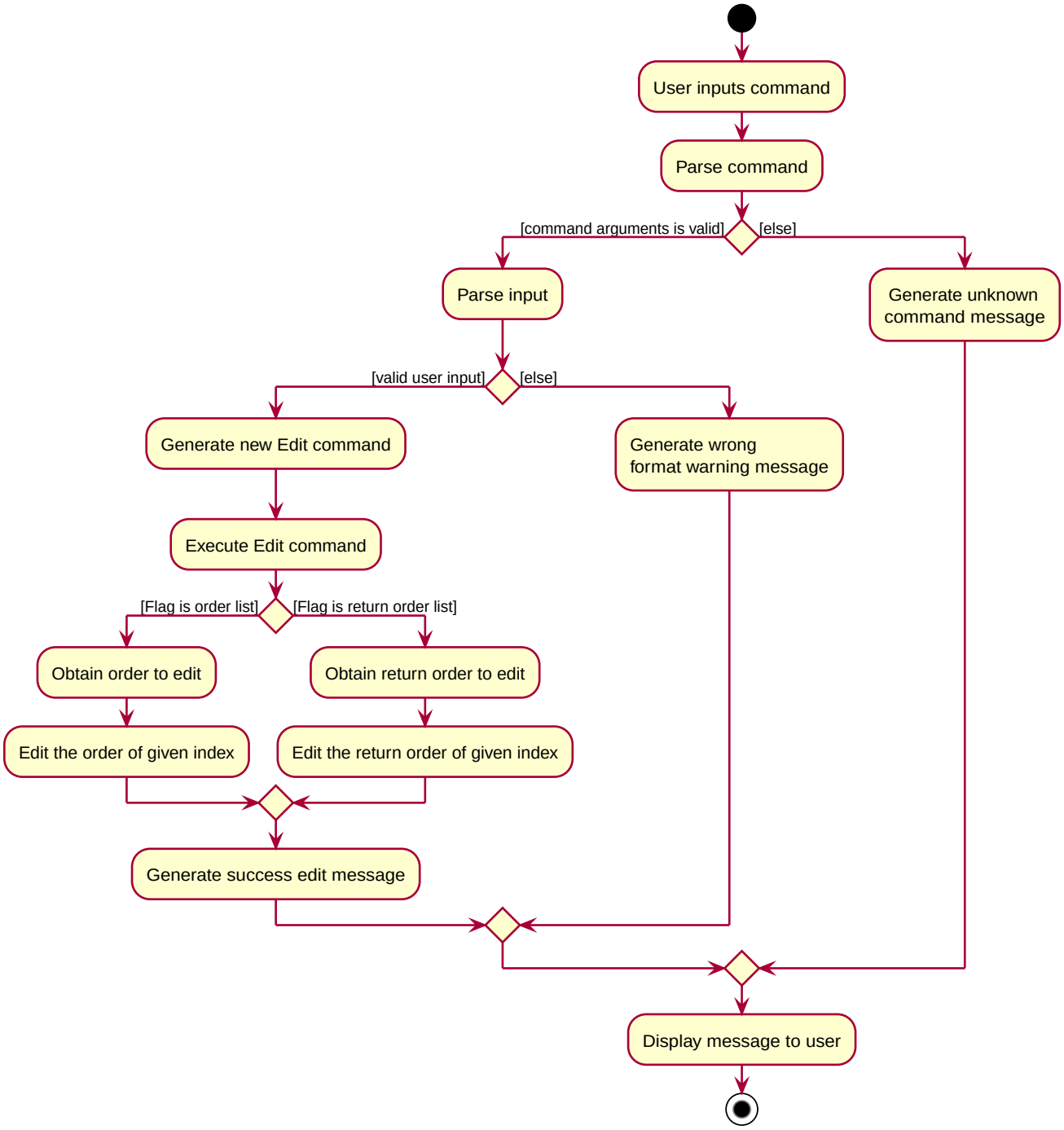

Figure 1. EditCommand Activity Diagram

The above figure illustrates the execution path of `edit` command when performed by the user.

Input when received, will be parsed by the `OrderBookParser`. `OrderBookParser` will check if command word matches any features command word.
In this feature, the command word is `edit`. If no command word is detected, a exception class should be generated for displaying of error message. `CommandException` is used in this feature to achieve that function.

Once validated, user input is once again parse and check for validity. At this step, if user have provided input not matching the valid `edit` format, an exception class is thrown.
Furthermore, if `NEW_VALUE` is invalid an exception should be thrown as well.
`ParseException` class is used in this scenario.

Some invalid `NEW_VALUE`:
1) Editing delivery date or return date to the past.
2) Change the transaction id of one parcel to match another parcel.
3) Violation of any field(s) restriction.

A correct input will prompts Delino to carry out the rest of the steps according.
1) Checking of the `FLAG`
2) Edits the the parcel. 3) Display edit success message.

Structure of Edit Command

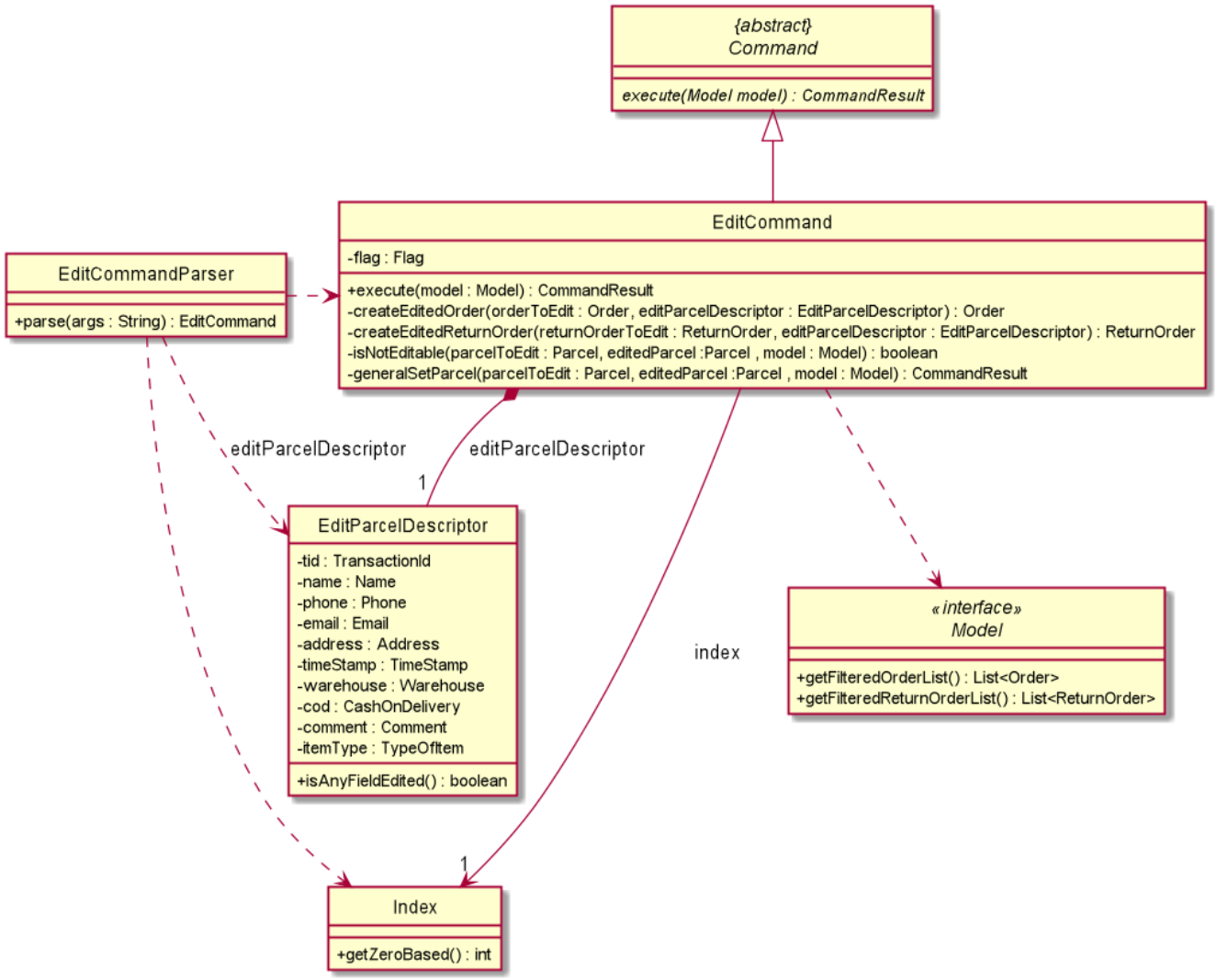

Figure 2. Edit Command Class Diagram

The class diagram above depicts the structure of `EditCommand`. As per any `Command` class, `EditCommand` needs to extend the abstract class `Command`.
Information that are left out in this class diagram are the common messages used in `EditCommand`.

Interactions between Edit Command and it's associated objects

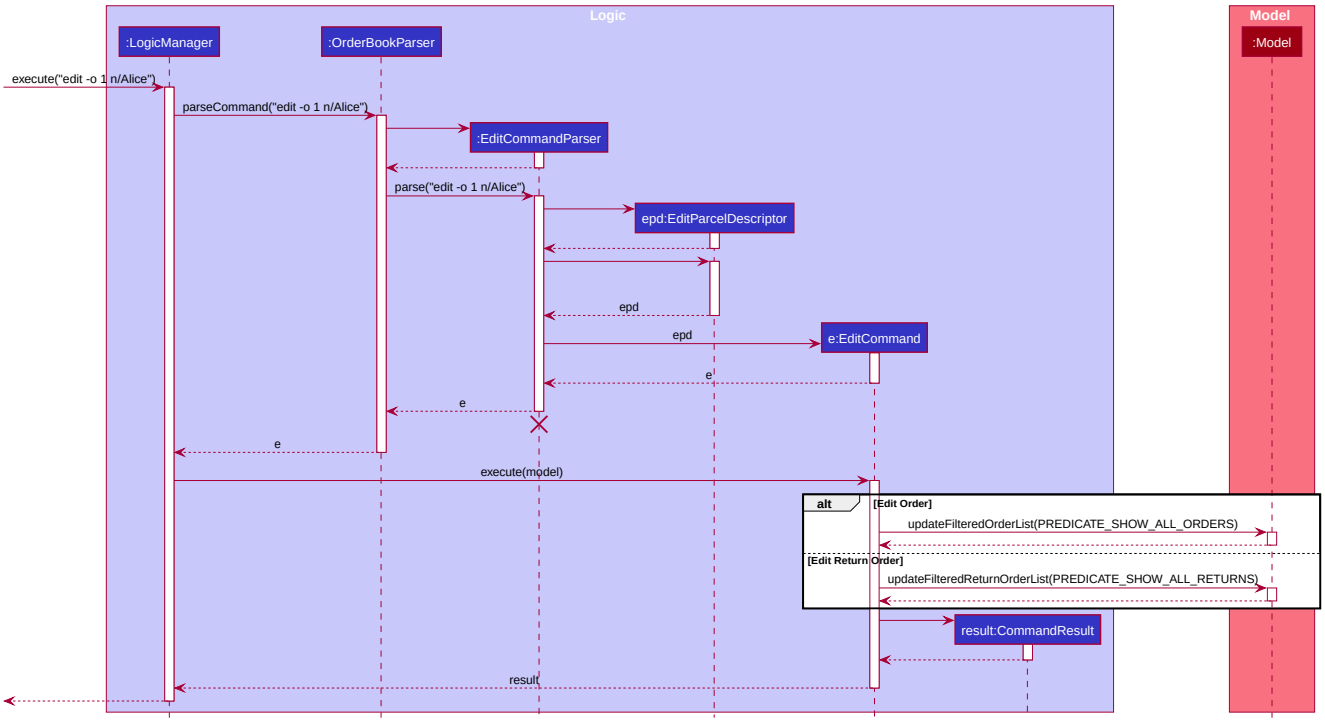

Figure 3. Edit Command Sequence Diagram

The above figure illustrates the important interactions of `EditCommand` when the user successfully edit the first displayed order name to `Alice`.

The handling of breaking down the user input is done in the `EditcommandParser` class which is called upon by the `OrderBookParser` after an initial check for correctness of the command input.

The `EditParcelDescriptor` class is a static class contained in the `EditCommand` class. It act as a helper class to allow the setting of all the `NEW_VALUE` to the corresponding `ORDER_ATTRIBUTE_PREFIX` in the `EditCommandParser` class. The `EditParcelDescriptor` object is then passed back as a parameter to instantiate an `EditCommand`. In the diagram above, the `EditParcelDescriptor` object is named as `epd`. The `EditCommand` object is then passed back as `e` to the `LogicManager` which will then call the `EditCommand#execute`. This execute method mainly calls the 3 helper method, not shown, `EditCommand#createEditedOrder`/`EditComand#createEditedReturnOrder` and `EditCommand#generalSetParcel`. The main function of these methods are to help `EditCommand` in updating the `ObservableList` in the `Model` class which is responsible for the updating of list displayed.

The `ObeservableList` is a JavaFX function which listens and automatically changes the list once an update is performed.

Search Feature

In this section, the functionality of the `search` feature, the expected execution path, the structure of the `SearchCommand` class and the interactions between objects with the `SearchCommand` object will be discussed.

What is the Search feature

The `search` feature was implemented as the `SearchCommand` in the `logic` package.

The search function allow users to search for any orders according to the provided input.

`search` feature format: `search [FLAG] [ORDER_ATTRIBUTE_PREFIX]/[KEYWORD]`

IMPORTANT	A space is needed in between each word.

NOTE	Keyword search is case-insensitive. E.g: Given `Jeremy` it matches `JeReMy`, `jeremy` or any permutations of alphabet casing.

There are two mode of searching, **general search** or **specific search**.
If the user does not provide any `ORDER_ATTRIBUTE_PREFIX`, a **general search** mode will be performed on orders, return orders, or both depending on the `FLAG`.

The `[FLAG] -o` when given, searches only for parcels in the order list.
The `[FLAG] -r` when given, searches only for the parcels in the return list.

- **General search** will search for all fields in an order/return orders/both that have any matching fields.

If the user provide any `ORDER_ATTRIBUTE_PREFIX`, a **specific search** will be performed.

- **Specific search** will search orders/return orders/both based on the given `ORDER_ATTRIBUTE_PREFIX`.

Execution paths of Search Command

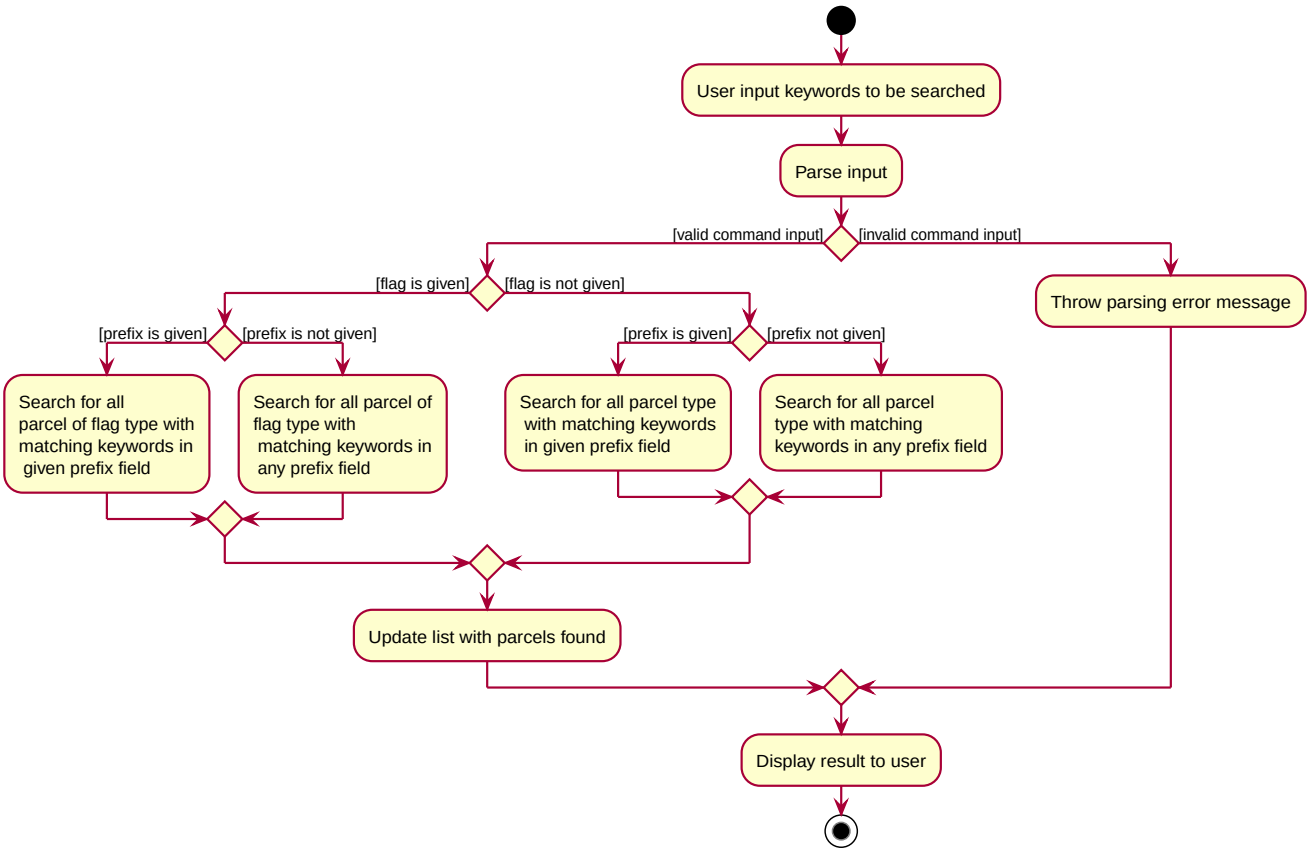

Figure 4. Search Command Activity Diagram

The above activity diagram illustrates the different execution paths of `search` command.
Whenever a user keys in an input with the `search` keyword, the `SearchCommandParser` class will handle the parsing of input.
User input will be validated in the `SearchCommandParser` class.

Input is deemed as invalid and `ParseException` is thrown under these scenarios:
1) `FLAG` given is not `-o` or `-r`.
2) Multiple `FLAG` detected.
3) No `KEYWORD` is given after `search`.

View the list of allowed prefixes in this `search` command here.

Structure of Search Command

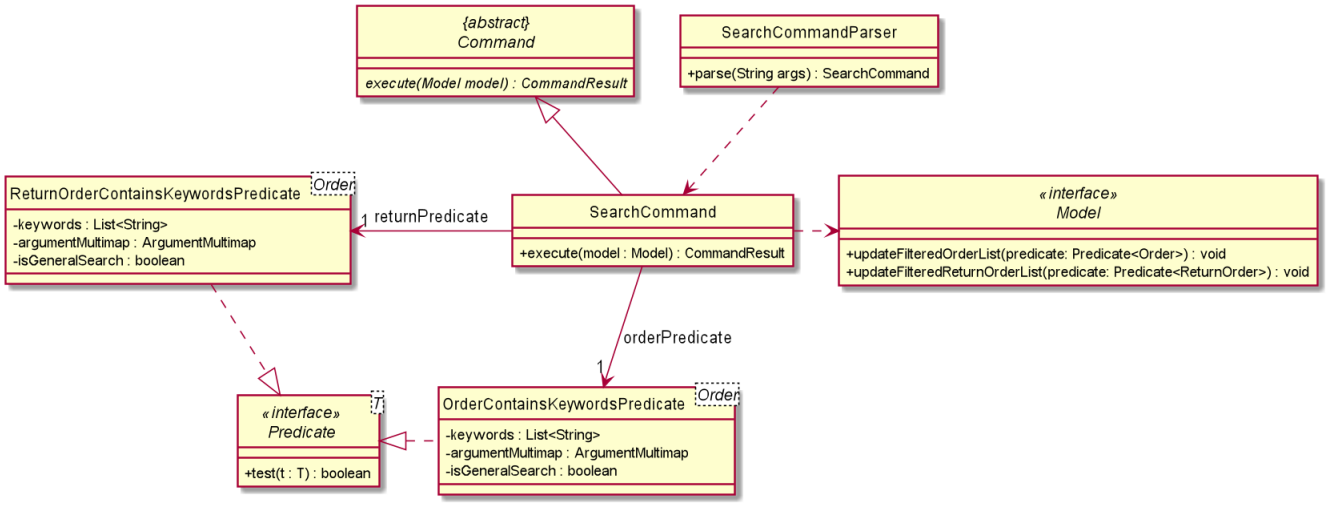

Figure 5. Search Command Class Diagram

The above class diagram depicts the structure of the class `SearchCommand`. As per any `Command` class, `SearchCommand` needs to extend the abstract class `Command`.
Information that are left out in this class diagram are the common messages used in `SearchCommand`.

Interactions between objects when Search Command is executed

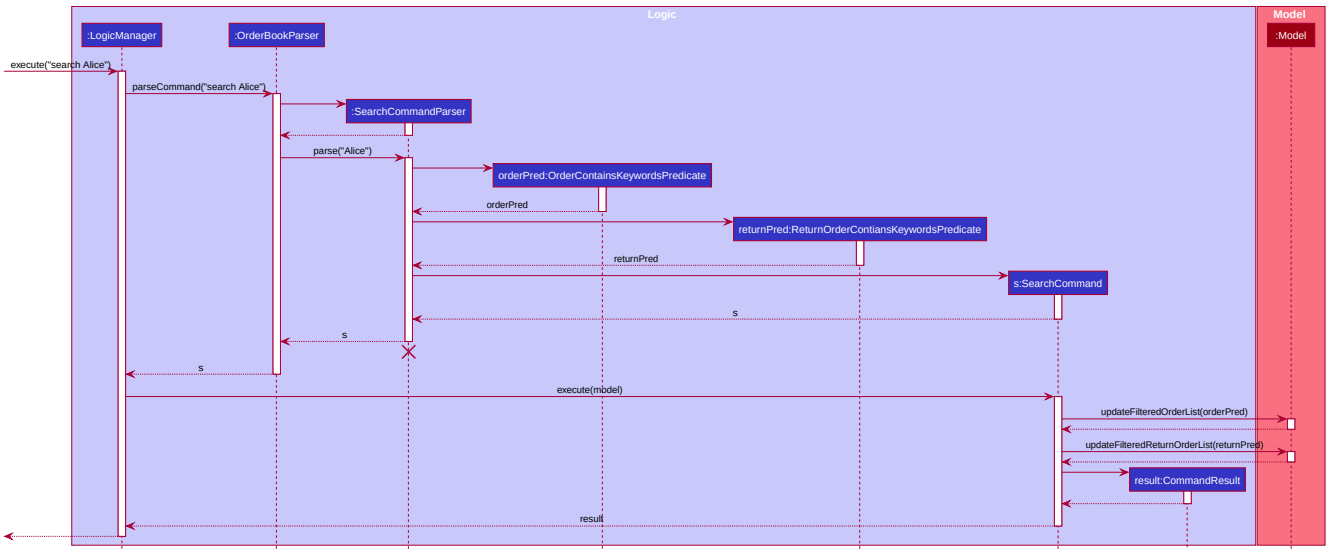

Figure 6. Search Command Sequence Diagram

The sequence diagram above illustrates the interactions between objects when `search` command is performed by the user.
Particularly, the interactions shown is a success `search` command executed by the user and only an abstract view is shown.

`LogicManager` first calls `parseCommand` with arguments representing the user input, `Alice`. The `SearchCommandParser` will then check for any invalid arguments passed by the user.

TIP

- If the given arguments are valid, `SearchCommandParser` will return a new `SearchCommand` object.
- If the given arguments are invalid or empty, a `ParseException` object will be thrown (not shown in the diagram).

The `SearchCommandParser` will then checks for the presence of any `FLAG`. The presence of one will result in different `SearchCommand` constructor being called.
The `SearchCommandParser` will call the both the `OrderContainsKeywordsPredicate` constructor and the `ReturnOrderContainsKeywordsPredicate` if no `FLAG` is given.
However, if a `FLAG` is given, the corresponding predicate will be instantiated and passed as an parameter for the `SearchCommand` constructor with the other left as null value.

IMPORTANT

- What is not shown is that optionally, either `OrderContainsKeywordsPredicate` or `ReturnOrderContainsKeywordsPredicate` can be null if a `FLAG` is given. However, under no circumstances should both be null.

The parsing of user input utilises `ArgumentTokenzier` (not shown in sequence diagram) to process and split each `KEYWORD` to it's corresponding `ORDER_ATTRIBUTE_PREFIX`, if given any.

If the preamble to any `ORDER_ATTRIBUTE_PREFIX` is not empty, a **general search** will be performed in which `KEYWORD` will be searched through all fields of parcel.
However, if `ORDER_ATTRIBUTE_PREFIX` is given and the preamble is empty, the **specific search** will be performed. Only parcel fields that correspond to the given `ORDER_ATTRIBUTE_PREFIX` will be searched and matched with the `KEYWORD`.

The order and return order list updates automatically as the JavaFX class `ObservableList` is used to listen to any changes.

Use case: UC05 - Editing order details

MSS

1. User request to edit order details.
2. Delino edit the order details
3. Delino display changes made.

   Use case ends.

Extensions

1a. Delino detects invalid syntax.

1a1. Delino shows an error message.

Use case ends.

1b. Delino unable to detect any order with the transaction id.

1b1. Delino shows no order found message.

Use case ends.

Use case: UC07 - Search an order

MSS

1. User request to search specific order by transaction id
2. Delino display the requested order.

Use case ends.

Extensions

1a. Delino detects invalid syntax.

1a1. Delino shows an error message.

Use case ends.

1b. Delino unable to find order with the transaction id.

1b1. Delino display order not found message.

Use case ends.