Amos Cheong - 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**:
  - Added `insert` command (Pull request #189)
    - What it does: Allows the user to insert new delivery order into the Order List. The user can add an order that contains the following attributes :
      - Transaction ID
      - Name
      - Phone
      - Address
      - Email
      - Delivery Timestamp
      - Warehouse location
      - Cash on Delivery
      - Comment (Optional)
      - Type Of Item (Optional)
    - Justification: This is an important feature and it is a must-have. It allows user to insert multiple orders into the list.
  - Refactor the `list` command (Pull request #239)
    - What it does: Instead of simply displaying all the orders, the user can choose to list :
      - Delivered orders
      - Undelivered orders
    - Justification: The feature helps the user, as a courier, to know how many orders have to be completed and how many orders are already completed.
  - Added the `show` command (Pull request #199)
    - What it does: It gives a more detailed information about the orders and return orders in

their lists, in the form of numbers and PieChart. The information shown is :

        - Total Earnings
        - Orders Completed
        - Orders Returned
        - A PieChart to show to the user on how many orders are completed and not completed, returned and not returned.

    - **Highlights**:
      Ability to let user to see information based on the date(s) provided. For example, the user can enter : `show 2020-02-02 2020-09-01` to see all the information about the orders that are from the date `2020-02-02` up to the date `2020-09-01`. In addition, user can show statistics about everything in all the lists or just simply showing the statistics for today.
    - Credits: UI Design ideas for `show` Command from https://www.youtube.com/watch?v=UDi051XyQ-U&t=339s

- **Minor enhancement**:
    - Fix bugs for the Application (Pull requests #309, #233)
        - What it does: Restricts user from entering any invalid values
        - Justification: This is an essential update, as the invalid values will affect the rest of the features.
- **Code contributed**: [Functional code and Test code]
- **Other contributions**:
    - Project management:
        - Responsible for ensuring that the team's code is compliant with the coding standards and in charge of defining, assigning and tracking all project tasks.
    - Enhancements to existing GUI:
        - Make major changes to the UI to change from Window to TabView (Pull request #318)
    - Documentation:
        - Add Manual Testing Instructions for Developer Guide. (Pull request #81)
        - Update Developer Developer Guide. (Pull requests #171, #252)
    - Community:
        - PRs reviewed (with non-trivial review comments): (Pull requests #299, #227, #218, #193, ...)
    - Tools:
        - Integrated a third party library (Netlify) to the project (#1708e0b)

Contributions to the User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Features

Inserting an order: `insert` [Done by Amos Cheong Jit Hon]

This section shows you how to use the Insert Command and the relevant examples.

Whenever you have a new delivery order to make, you have to add it into your list of orders. The Insert Command is in charge of this functionality.

How to use the Insert Command

Here are the steps on using the Insert Command:

Step 1 : Type in the keyword **insert**

Step 2 : Add in the prefixes `TRANSACTION_ID` `CUSTOMER_NAME` `ADDRESS` `PHONE_NUMBER` `EMAIL` `ORDER_TIMESTAMP` `WAREHOUSE_LOCATION` `CASH_ON_DELIVERY` of the delivery orders

Step 3 : Skip Step 4 if the specific order does not have the customer's comment and the type of item.

Step 4 : Add in the Customer comment and type of item using the prefixes `[COMMENT_BY_CUSTOMER]` `[TYPE_OF_ITEM]`

Step 5 : Press `Enter` on your keyboard to see the magic!

What constitutes a valid Insert command

Here is the syntax of the **Insert** Command :

```
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]
```

- The `TRANSACTION_ID` refers to the transaction id of the order.
- The `CUSTOMER_NAME` refers to the name of the recipient of the order.
- The `ADDRESS` refers to the address of the recipient.
- The `PHONE_NUMBER` refers to the phone number of the recipient.
- The `EMAIL` refers to the email address of the recipient.
- The `DELIVERY_DATE_&_TIME` refers to the delivery date and time of the order.
- The `WAREHOUSE_LOCATION` refers to the warehouse that the courier should collect the order from.
- The `CASH_ON_DELIVERY` refers to the money earned from delivering the particular order.

NOTE

- The `[COMMENTS_BY_CUSTOMER]` refers to the comment made by the recipient to the courier.
- The `[TYPE_OF_ITEM]` refers to the type of item that the courier is delivering.
- The prefixes `tid/TRANSACTION_ID n/CUSTOMER_NAME a/ADDRESS p/PHONE_NUMBER e/EMAIL dts/DELIVERY_DATE_&_TIME w/WAREHOUSE_LOCATION cod/CASH_ON_DELIVERY` are compulsory.
- The prefixes `c/COMMENTS_BY_CUSTOMER type/TYPE_OF_ITEM` are optional.
- Any compulsory prefixes that is absent will result in the App displaying an error message.
- There should be a spacing in between every prefixes. For example, `tid/TRANSACTION_ID n/CUSTOMER_NAME` is allowed but `tid/TRANSACTION_IDn/CUSTOMER_NAME` will be an invalid command input.

WARNING

- `TRANSACTION_ID` must be alphanumeric (No numbers or special characters allowed).
- `ADDRESS` must have a postal code.
- `EMAIL` should be a valid email address.
- `CASH_ON_DELIVERY` must start with a dollar sign followed by the value.
- `CASH_ON_DELIVERY` value must be strictly less than $10,000,000,000,000.If there is a need to add decimal places, you can only add two.
- **Do Not add any commas in the value of `CASH_ON_DELIVERY`**
- To add the Delivery Date and Time, first type in the date in this format : YYYY-MM-DD. Followed by adding the time in 24 hour format.
- The value for `DELIVERY_DATE_&_TIME` must be a time in the future. For example, if the date and time now is 2020-04-03 1200, you are not allowed to add 2020-04-03 1159 as the `DELIVERY_DATE_&_TIME`.
- `TYPE_OF_ITEM` must be alphanumeric (No numbers or special characters allowed).
- `CUSTOMER_NAME` mut be alphanumeric(No numbers or special characters allowed).
- `PHONE_NUMBER` should not have a spacing in between. `n/90011009` is allowed, but `n/9001 1009` is not allowed.
- Only numbers are allowed in `PHONE_NUMBER`.

There are two different scenarios on how to insert the orders :

Table 1. Possible combinations of Insert Command

Scenario	Command	Result
Insert the order without a comment and no item type	`insert tid/A094844 n/John Doe a/Blk 505 Tampines #10-33 S520505 p/98761111 e/johndoe@example.com dts/2020-05-20 1300 w/Yishun cod/$4`	You should be able to see that the order with transaction id 'A094844' will be inserted into the list of delivery orders.
Insert the order with all the order attributes including the non-compulsory ones	`insert tid/C1023456789 n/Amos Cheong a/Blk 571 Hougang st 51 #02-02 S530571 e/amoscheong@example.com p/90010019 dts/2020-05-10 1650 w/Marsiling cod/$5 c/Leave it at the riser type/glass`	You should see that the order with transaction id 'C1023456789' is inserted into the list of delivery orders.

TIP

- List of order attribute prefixes can be found here.

Listing orders : `list` [Done by Amos Cheong Jit Hon]

In this section, you will learn more about the List command and how to use it.

As a courier, you would want to take a look at all the orders that you have in your list of orders regardless of the type of orders or the order status. The List Command will enable you to view all these orders.

How does the List Command works

Here are the steps to execute the List command:

Step 1 : Type in the keyword **list**.

Step 2 : If you want to simply see all your delivery and return orders. Otherwise, please proceed to Step 3. Else, skip to Step 4

Step 3 : Provide the following `[KEYWORD]` : `done` (Showing all your completed orders) or `undone` (Show all your uncompleted orders)

Step 4 : Press `Enter` on your keyboard and see the magic!

What constitutes a valid List command

The syntax of a valid `list` command is as shown: `list [KEYWORD]`

There are three types of list commands that are shown in the examples below :

Table 2. Possible combinations of List Command

Scenario	Command	Result
Display all orders	`list`	Show two lists of all orders. One list for delivery orders, the other for return orders
Display all completed orders	`list done`	Show two lists of all completed orders. One list for delivery orders, the other for return orders
Display all uncompleted orders	`list undone`	Show two lists of all uncompleted orders. One list for delivery orders, the other for return orders

IMPORTANT

- `KEYWORD` can only be either `done` or `undone`.

Showing statistics : `show` [Done by Amos Cheong Jit Hon]

This section will explain more about the Show command and how to use it.

At some point of time, you would want to know how many orders have you delivered or what is your earnings for today. Therefore, you have to use the Show Command to view those information.

How to use the Show command

In this section, you will learn how to use the Show Command.

Here is how you can show the statistics of your orders:

Step 1 : Type `show`

Step 2 : If you want to see your statistics for all the orders, simply type `all` and skip to Step 5

Step 3 : Type `today` to show the statistics for today or simply type in a date in a date format of `yyyy-mm-dd`

Step 4 (Optional) : Type another date in `yyyy-mm-dd` format to see the statistics within the date range. This date must be after or equal to the date provided previously

Step 5 : Press `Enter` on your keyboard to see the magic!

NOTE

The `show` command opens up a new window that displays the following information :

- Earnings
- Orders Delivered
- Orders Returned
- PieChart that display numbers for orders delivered, not delivered, returned and not returned

What constitutes a valid Show Command

In this section, you will learn about the correct syntax for a valid `show` command and all the different combinations of the command.

All the syntax for a valid `show` command is shown below:

- `show START_DATE [END_DATE]`
- `show all`
- `show today`
- `show DATE`

NOTE

- Only one or two arguments is allowed for the `show` command.
- The value of `START_DATE` can be just the word `today` (Showing statistics just for today)
- `END_DATE` is an optional field. If included, the command will show statistics based on the given range (inclusive) of dates
- `START_DATE` cannot be a date after `END_DATE`
- `show all` command is the only command that accepts only one argument. It shows all the statistics in the list regardless of the dates

Here are the different scenarios of showing statistics :

Table 3. Possible combinations of Show Command

Scenario	Command	Result
Showing all the orders statistics regardless of date	`show all`	You will be brought to the Statistics tab and the statistics of all orders will be displayed to you
Showing the statistics for today	`show today`	You will be brought to the Statistics tab and the statistics for today's orders will be displayed to you
Showing the statistics for the date between today and the end date provided	`show today 2020-12-03`	You will be brought to the Statistics tab and the statistics between today's date and `2020-12-03` will be displayed to you
Showing the statistics for just the given date	`show 2020-12-03`	You will be brought to the Statistics tab and the statistics in `2020-12-03` will be displayed to you
Showing the statistics within two given dates	`show 2020-12-03 2021-01-01`	You will be brought to the Statistics tab and the statistics for the date between `2020-12-03` and `2021-01-01` will be displayed to you

Contributions to the Developer Guide

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

Features

Insert Feature

This section, will introduce the Insert Feature. In addition, it will show the expected path-execution, the structure of the of the **InsertCommand** class, structure of the **InsertCommandParser** class and it will also describe the interaction of objects between the **InsertCommand** object and other object classes.

What is the Insert feature

The insert feature allows the user to insert an incoming delivery order into the list using the command line. The order consists of : Transaction ID, Name, Phone, Address, Email, Delivery Timestamp, Warehouse location, CashOnDelivery

The order also consists of two optional fields that can be added:

1. Type of Item
2. Comment for Courier

Structure of Insert feature

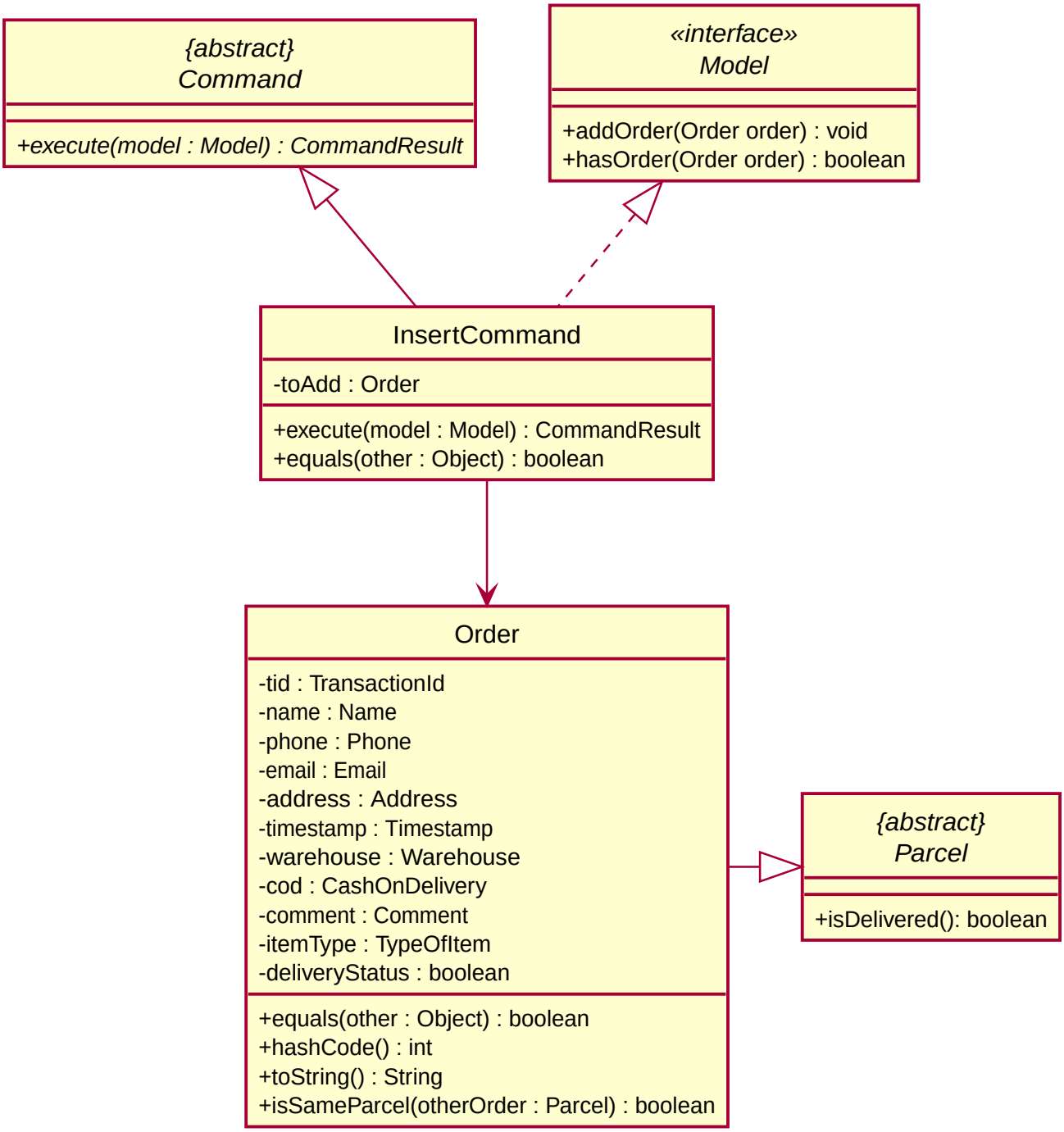


Figure 1. Insert Class Diagram

Structure of InsertCommandParser

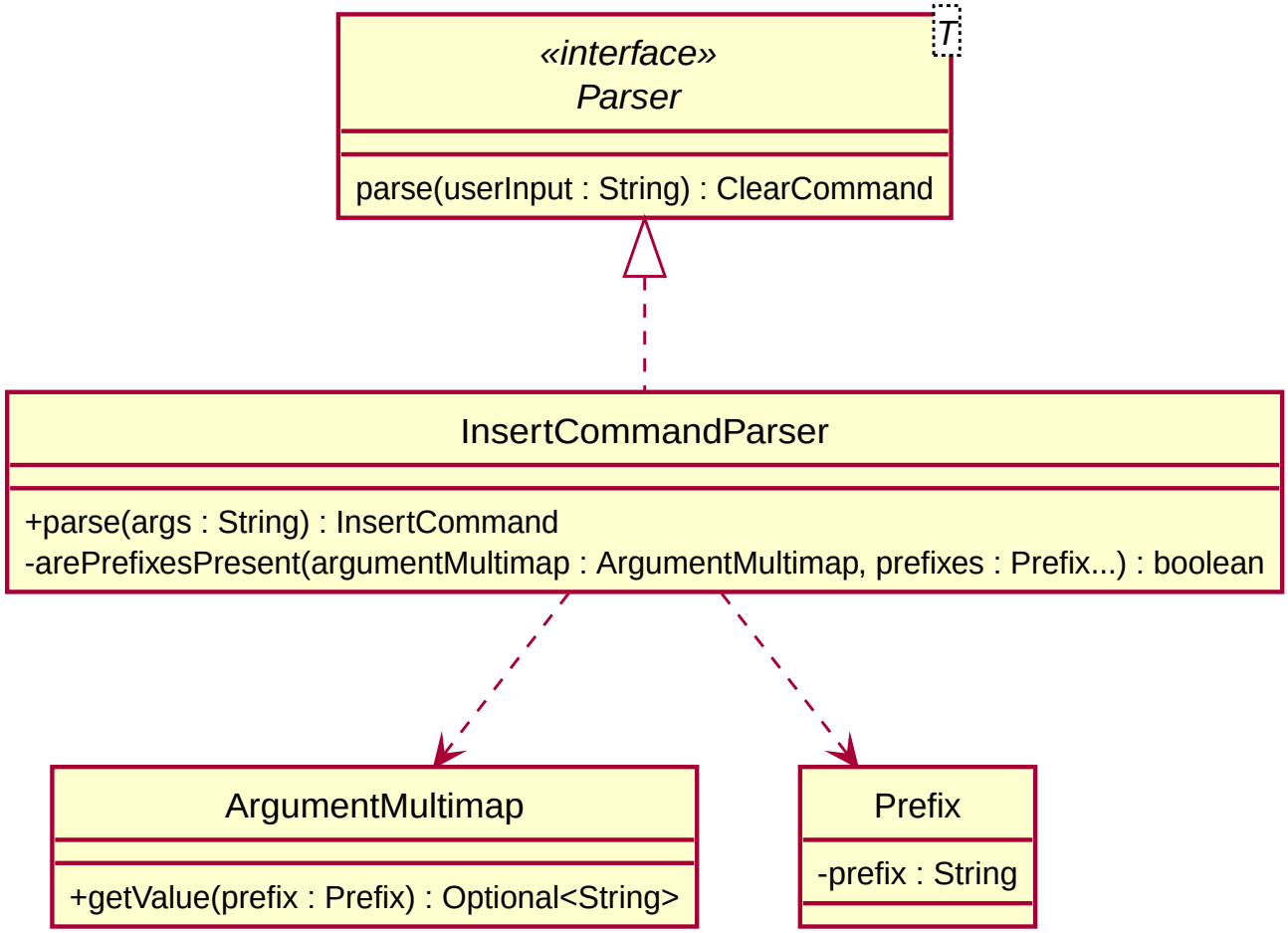


Figure 2. InsertCommandParser Class Diagram

Path Execution of Insert Command

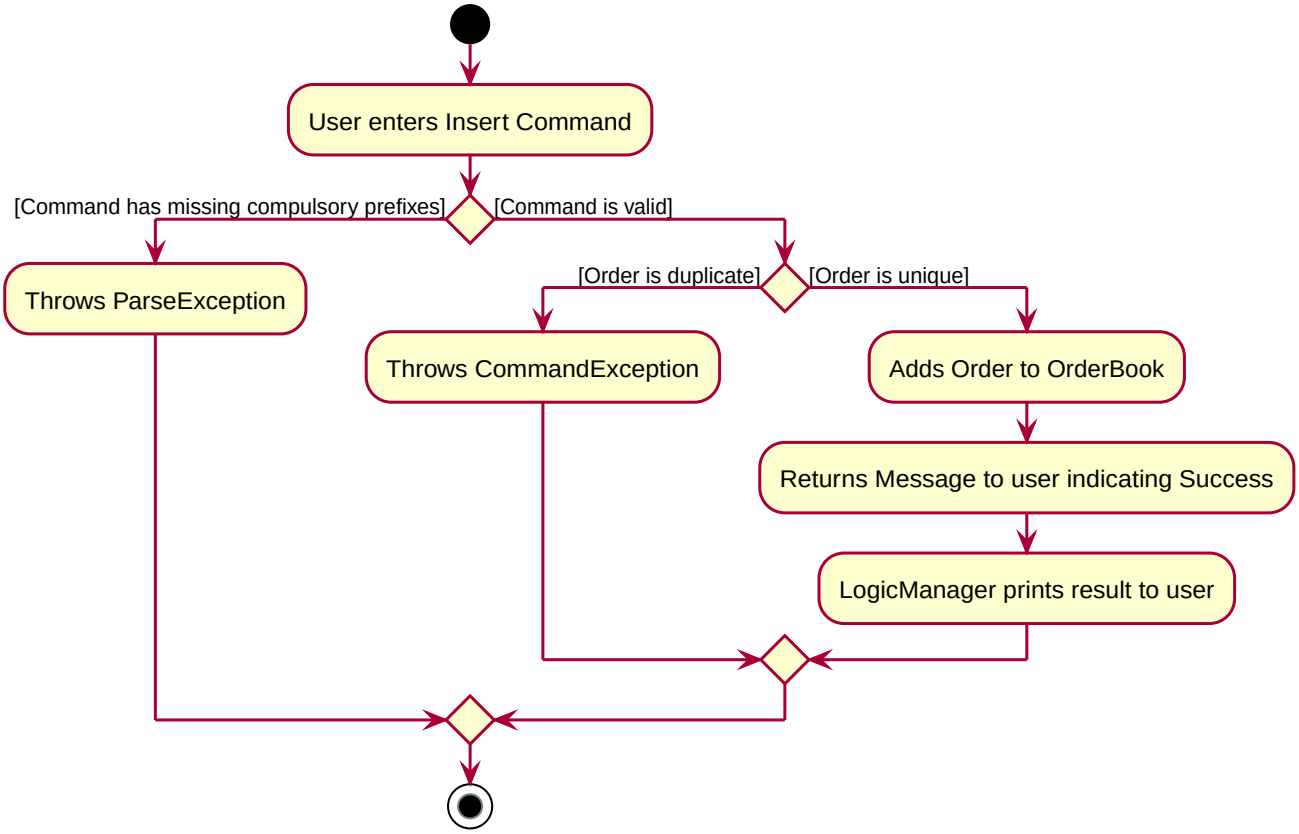


Figure 3. Insert Activity Diagram

Interaction between objects when the Insert Command is executed

Here is the sequence diagram for the **Insert Command** as shown below:

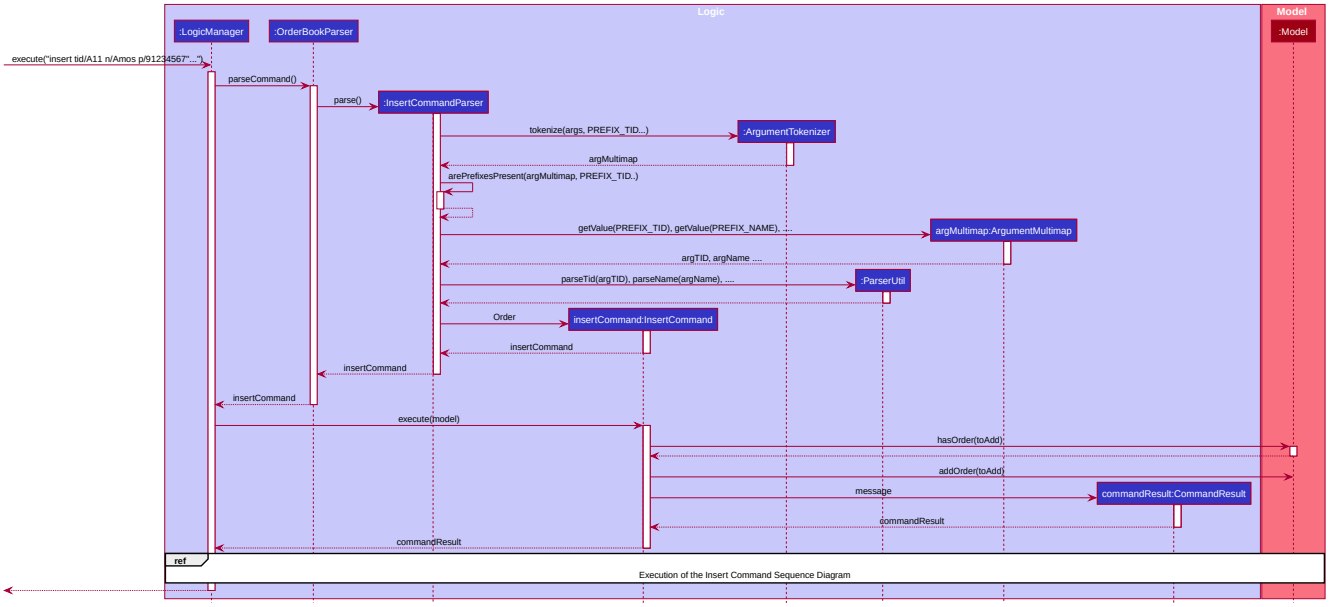


Figure 4. Insert

The arguments of the Insert Command will be parsed using the parse method of the `InsertCommandParser` class.
The `InsertCommandParser` will tokenize the arguments parsed in using the tokenize method of `ArgumentTokenizer` class which returns the tokenized arguments. Using the tokenized arguments, the Parser will check if the arguments parsed in matches with the tokenized arguments using the

arePrefixesPresent method.

There are two scenarios :

1. Some compulsory prefixes are not present :
   `InsertCommandParser` will throw a new `ParseException` object to the `LogicManager`.
2. All compulsory prefixes are present in the arguments :
   It will the proceed to use the getValue method of the `ArgumentMultimap` class to get the value of the prefix. For example, if the argument parsed in is tid/A12345, the getValue method will get the value 'A12345'. Subsequently, it will use the `ParseUtil` methods to get the corresponding object values and put it into the parameters of the new `Order` object. The order object will be put into the parameter of the `InsertCommand` object and this will be returned to the `LogicManager` class for execution.

`LogicManager` will call the execute() method of this `InsertCommand` object. In the execute() method, it will use the `Model` class to call hasOrder method to check for duplicates, if it is a duplicate, the order will throw a CommandException which indicates that there is a duplicate order in the OrderBook already. Else, it will successfully inserts the new order using addOder method. Finally, it return a new `CommandResult` object, containing a String that indicates a successful insertion.

List feature

This section describes the functionality , the structure, interactions between objects and path the path execution of the **List Command**.

What is the List feature

List feature allows the user to see all the orders from both Delivery Orders and Return Orders.

The user can enter `list` to display all the orders. Besides that, the user can also input `done` to dispay all delivered orders and `undone` to display all orders that are not delivered.

Structure List feature

The structure of the List Feature is as shown below:

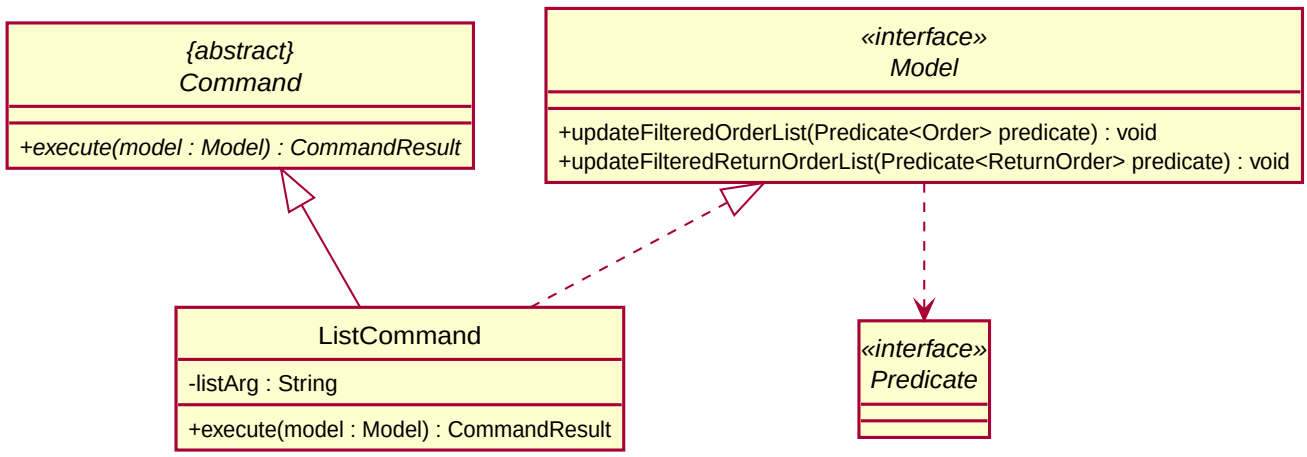


Figure 5. List Class Diagram

Path execution of the List Command

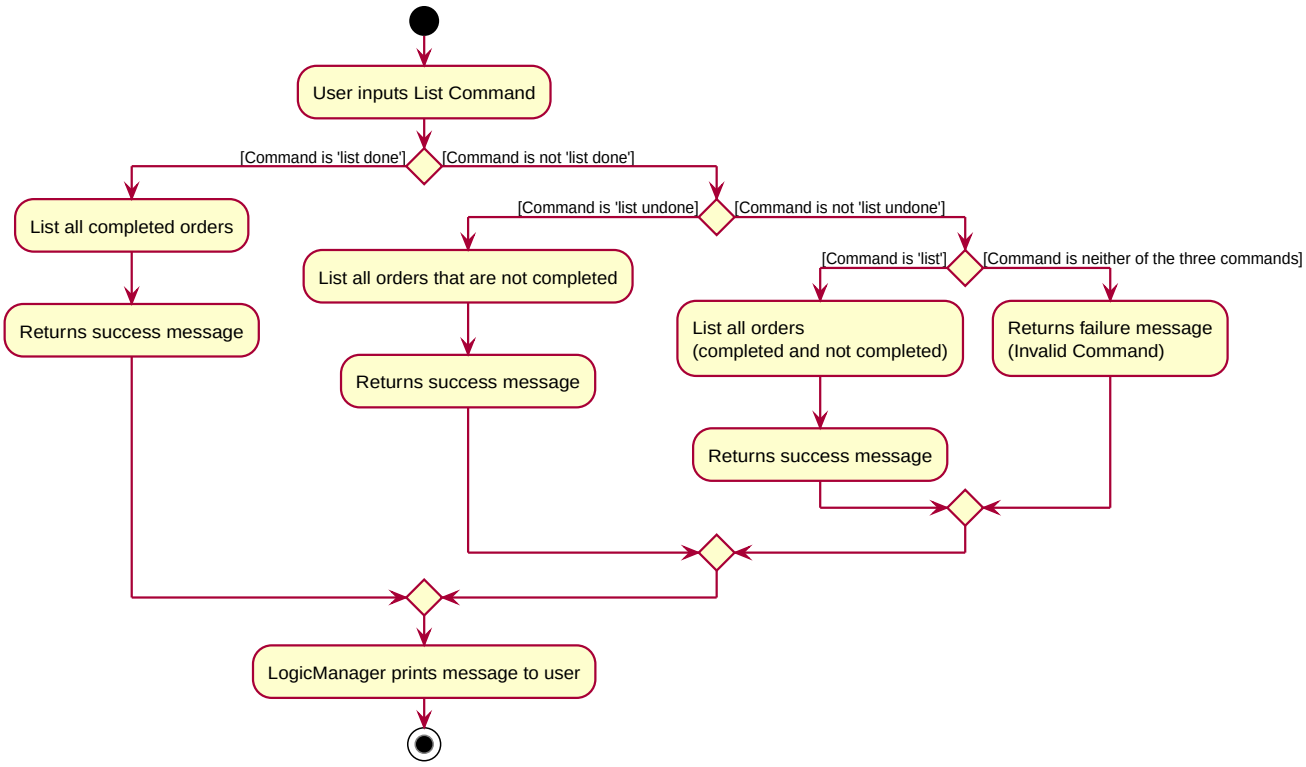


Figure 6. List Activity Diagram

The above activity diagram shows the logic and the path execution when the **List Command** is executed.

Interaction between objects during execution of List Command

The sequence diagram for the **List Command** is shown below:

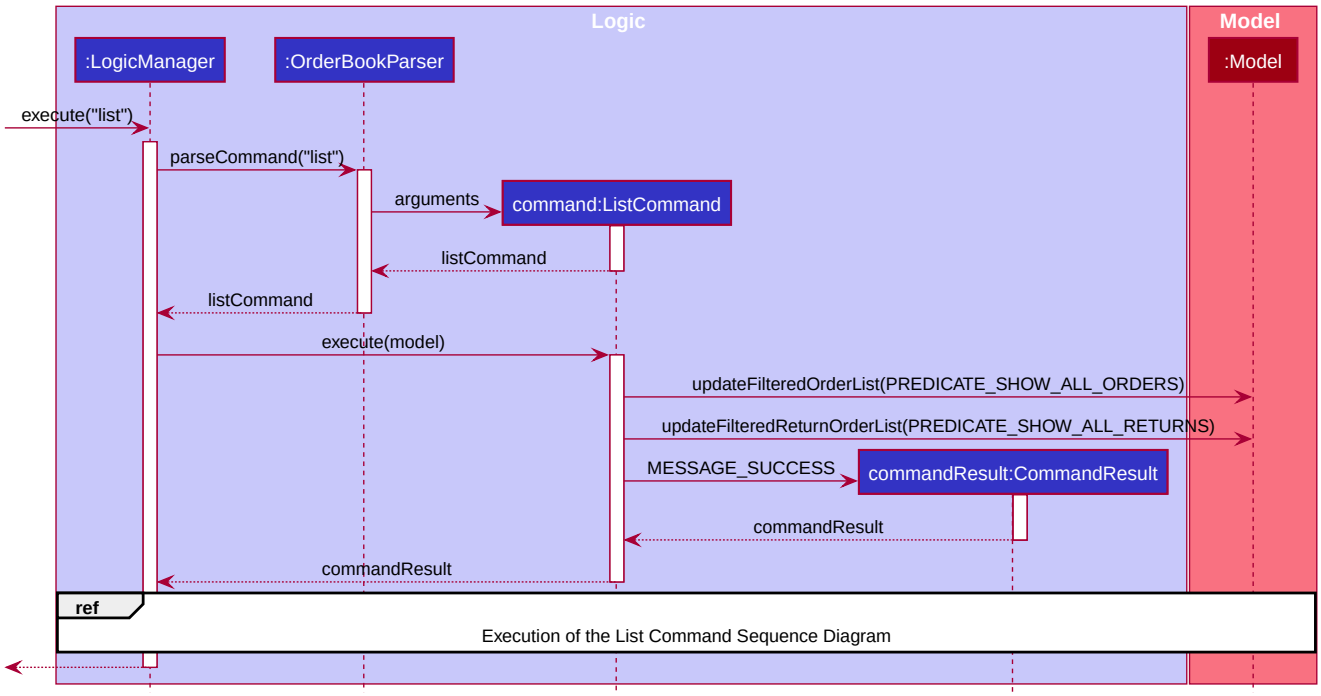


Figure 7. List Command Sequence Diagram

The user first calls the command "list".

NOTE	The second argument of the `list` command can be `done` or `undone` or an empty String.

The LogicManager will call the parseCommand method of OrderBookParser, which then passes the second argument into the `ListCommand` object. This object will then be ultimately returned to the `LogicManager`. Next, the `LogicManager` will call the execute(model) method using the `ListCommand` object. In this method, it wil use the `Model` object to call the methods : updateFilteredOrderList and updateFilteredReturnOrderList. Since in this case, the argument is empty, the predicate that is parsed to the two methods will always result to true, which means to list everything from the order book and return book. When completed, the execute(model) will return a `CommandResult` object to the `LogicManager`, indicating that the command execution is a success.

Show feature

This section describes the functionality , the structure, interactions between objects and path the path execution of the **Show Command**.

What is the Show feature

Show feature allows the user to see the statistical information of all the orders for both Delivery Orders and Return Orders.

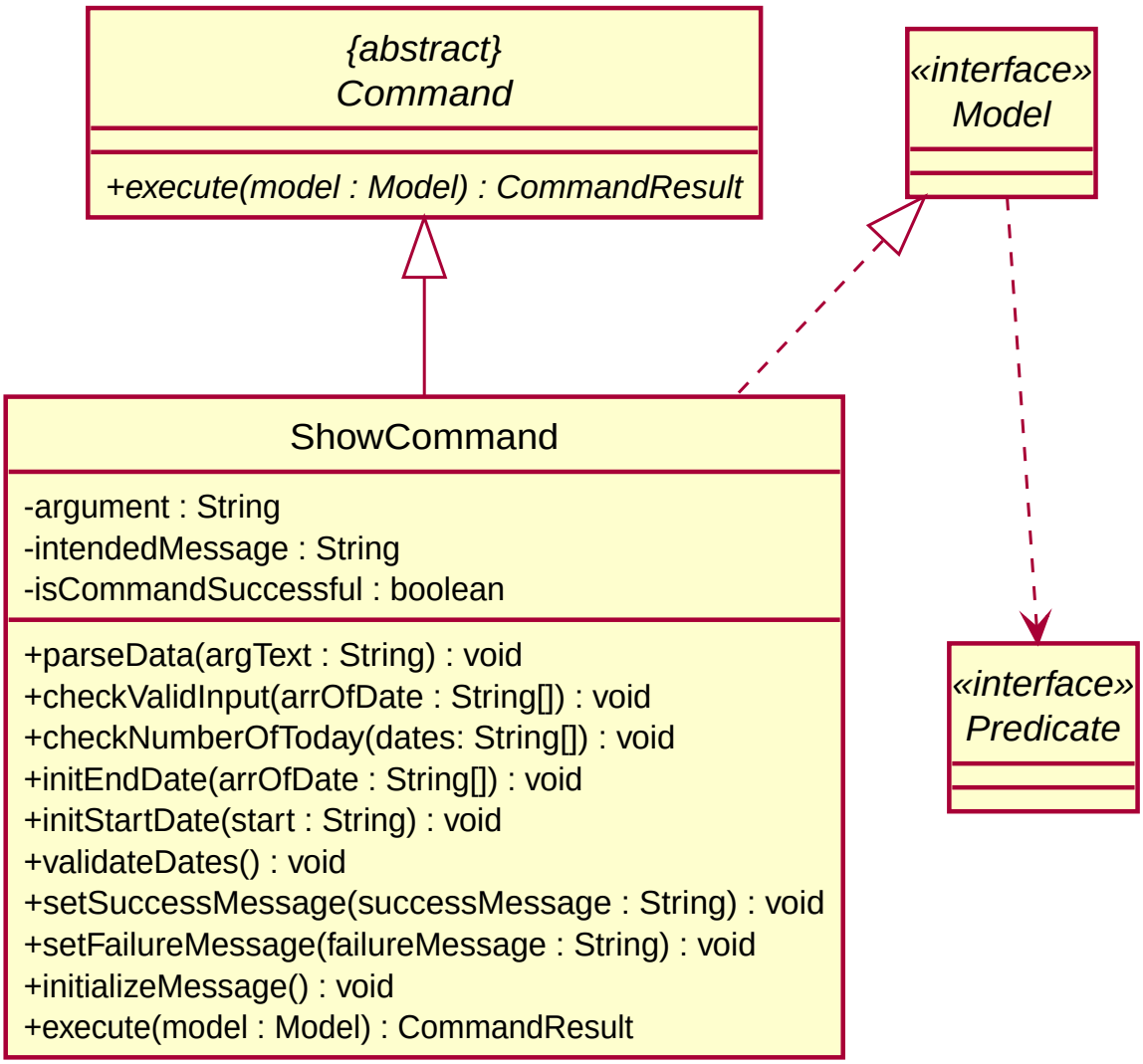
There are a few ways in which the user can input to the command box to execute the **Show Command** :

- `show START_DATE [END_DATE]`
- `show all`
- `show today`
- `show DATE`

Structure Show feature

The structure of the List Feature is as shown below:

Show Command Class Diagram



Path execution of the Show Command

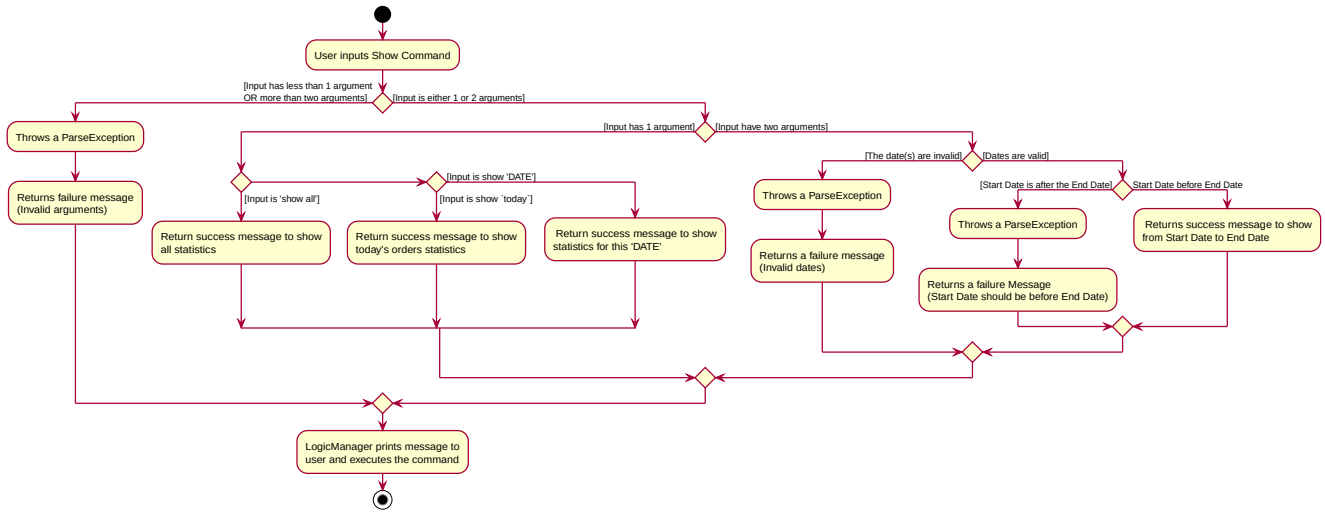


Figure 8. Show Command Activity Diagram

The above activity diagram shows the logic and the path execution when the **Show Command** is executed.

Interaction between objects during execution of Show Command

The sequence diagram for the **Show Command** is shown below:

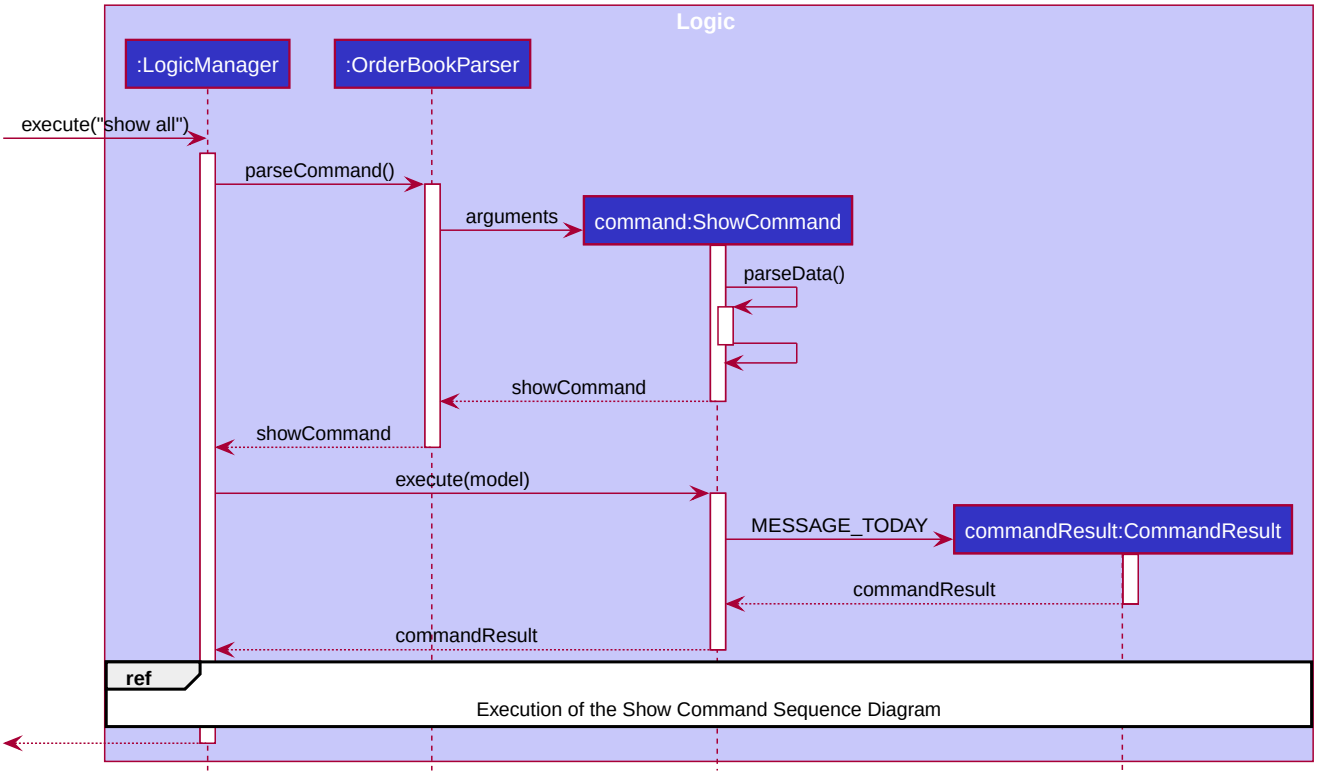


Figure 9. Show Command Sequence Diagram

The user first calls the command "show all".

NOTE	It can accept either one or two arguments.

The LogicManager will call the parseCommand method of OrderBookParser, which then passes the second argument into the `ShowCommand` object. Within the object, it will call the parseData method to make sense of the dates given. After that, it returns the object to the `LogicManager`. Next, the `LogicManager` will call the execute(model) method using the `ShowCommand` object. When completed, the execute(model) will return a `CommandResult` object to the `LogicManager`, indicating that the command execution is a success. In this case where the input is "show all", it will have a message that indicates that the command is showing all information.

Use Cases

Use case: UC01 - Insert an order

MSS

1. User key in the order details.
2. Delino inserts the order details.
3. Delino displays order added.

Use case ends.

Extensions

1a. Delino detects invalid syntax.

1a1. Delino shows an error message.

Use case ends.

Use case: UC14 - Show statistics

MSS

1. User requests to see the statistics of orders.
2. Delino opens a window that contains the statistics.

Use case ends.

Use case: UC10 - Listing all orders

MSS

1. User requests to view the list of orders.
2. Delino display list of orders.

Use case ends.

Extensions

1a. Delino detects invalid syntax.

1a1. Delino shows an error message.

Use case ends.

2a. Delino detects no orders.

2a1. Delino shows empty order list message.

Use case ends.

Glossary

Command Prefix

Table 4. Command Prefix

Prefix	Order Attributes	Used in the following Command(s)
ot/	Order Type	Import
tid/	Transaction ID	Edit, Insert, Return, Search, Import
n/	Customer Name	Edit, Insert, Return, Search, Import
a/	Address	Edit, Insert, Return, Search, Import
p/	Phone Number	Edit, Insert, Return, Search, Import
e/	Email	Insert, Edit, Return, Search, Import
dts/	Delivery Date And Time	Edit, Insert, Return, Search, Import
rts/	Return Date and Time	Return, Search, Import
w/	Warehouse Location	Edit, Insert, Return, Search, Import
cod/	Cash On Delivery	Edit, Insert, Search, Import
c/	Comments by Customer	Edit, Insert, Return, Search, Import
type/	Type of Item	Edit, Insert, Return, Search, Import

Appendix G: Instructions for Manual Testing

Inserting an order

1. Insert a minimum of 2 orders
    a. Insert command format: `insert tid/TRANSACTION_ID n/CUSTOMER_NAME a/ADDRESS p/PHONE_NUMBER e/EMAIL ts/DELIVERY_DATE_&_TIME w/WAREHOUSE_LOCATION cod/CASH_ON_DELIVERY [c/COMMENTS_BY_CUSTOMER] [type/TYPE_OF_ITEM]`
    b. Test case: `insert tid/9876543210 n/John Doe a/Blk 572 Hougang st 51 #10-33 S530572 p/98766789 e/johndoe@example.com ts/2020-02-20 1300 w/Yishun cod/$4`
    Expected: Inserts an order with the above details to the list and displayed on the GUI
    c. Test case: `insert tid/1023456789 n/Amos Cheong a/Blk 572 Hougang st 51 #11-37 S530572 p/9001 0019 e/amoscheong@example.com ts/2020-03-10 1650 w/Marsiling cod/$5 c/Leave it at the riser type/glass`
    Expected: Inserts the order to the list, including the item type and the order comment
    d. Test case: Invalid Syntax

Expected: No order is added. Error details shown in the response message. A help message displayed for user to insert accordingly. Status bar remain unchanged

e. Test case: Insert order with existing Transaction ID in list
Expected: An error will occur and a message will be displayed, stating that order with duplicate ID cannot be inserted into the list

Show

1. Opens a window which shows the statistics of the current lists of orders. It displays information such as earnings made, orders delivered and orders returned (Including a PieChart).

   a. Test case: `show all`
   Expected: All statistical information of all the orders shown in the statistics tab.

   b. Test case: `show today`
   Expected: All statistical information today shown in the statistics tab.

   c. Test case: `show today 2020-12-03`
   Expected: All statistical information between the dates shown in the statistics tab.

   d. Test case: `show 2020-12-03`
   Expected: All statistical information for the given date shown in the statistics tab.

   e. Test case: `show 2020-12-03 2021-01-01`
   Expected: All statistical information within the dates shown in the statistics tab.

   f. Test case: Invalid syntax
   Expected: An error will occur and the response box will show an error message.

List orders

1. List all the delivery orders for the user. The type of orders to be listed is dependent on the command input from the user

   a. Test case: `list`
   Expected: List all the delivery orders, showing all completed and uncompleted orders.

   b. Test case: `list done`
   Expected: List all completed delivery orders.

   c. Test case: `list undone`
   Expected: List all uncompleted delivery orders.

   d. Test case: `list ANY_WORD_OTHER_THAN_UNDONE_AND_DONE`
   Expected: An error will occur, a message will appear in the response box, indicating an invalid list command