AB-3

Developer Guide

Setting up, getting started

Refer to the guide *Setting up and getting started*.

Appendix: Requirements

Product scope

Target user profile:

- NUS students
- has a hard time organising and planning what modules to take
- prefer desktop apps over phone apps
- can type fast
- prefers typing to mouse interactions
- is reasonably comfortable using CLI apps
- want to have a good way to check all MCs
- wants to have a good way to check all fulfilled pre-requisites
- wants to have a good way to plan for all their modules

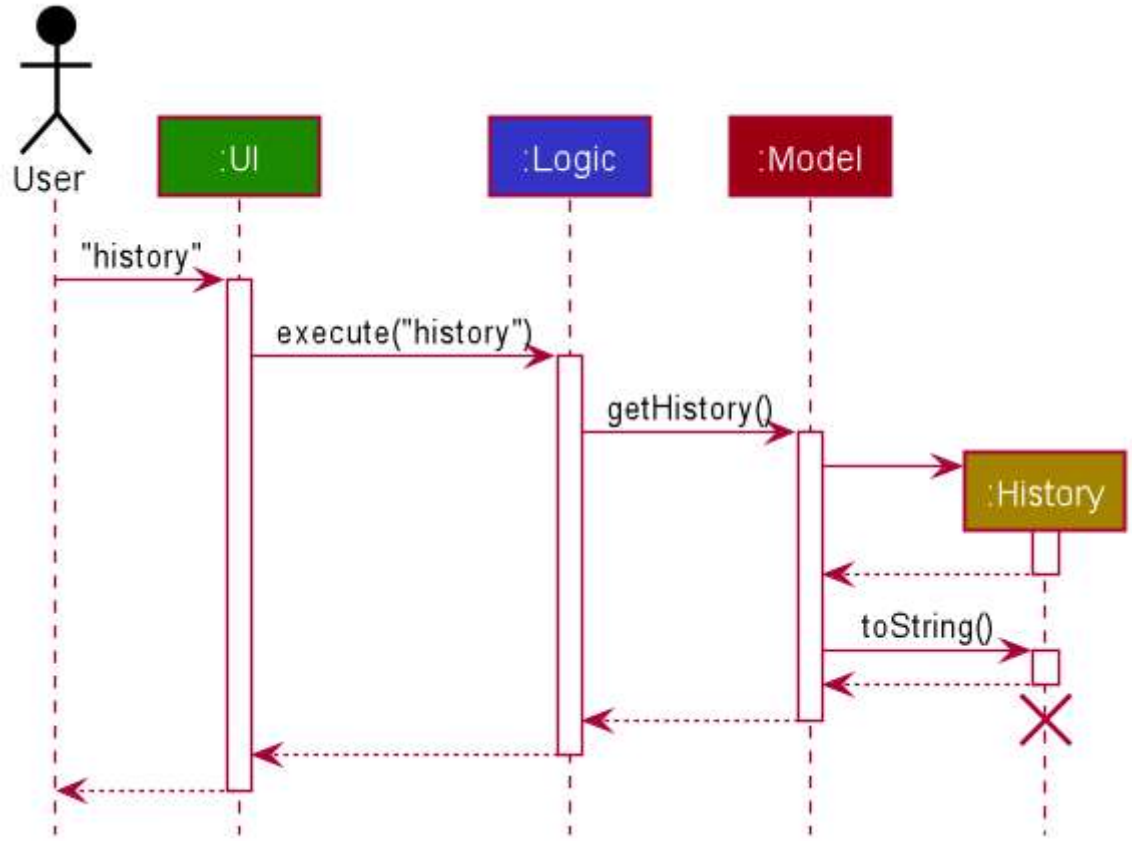
Value proposition:

- managing study plan is much easier than existing choices (i.e. WHAT-IF report)
- planning for modules is more automatic/convenient than manual inputs (NUSMOD)

Each semester's `toString()` method is called which internally calls each modules `toString()` method.

Overview: History command

The following presents a final overview of how the `history command` is used:



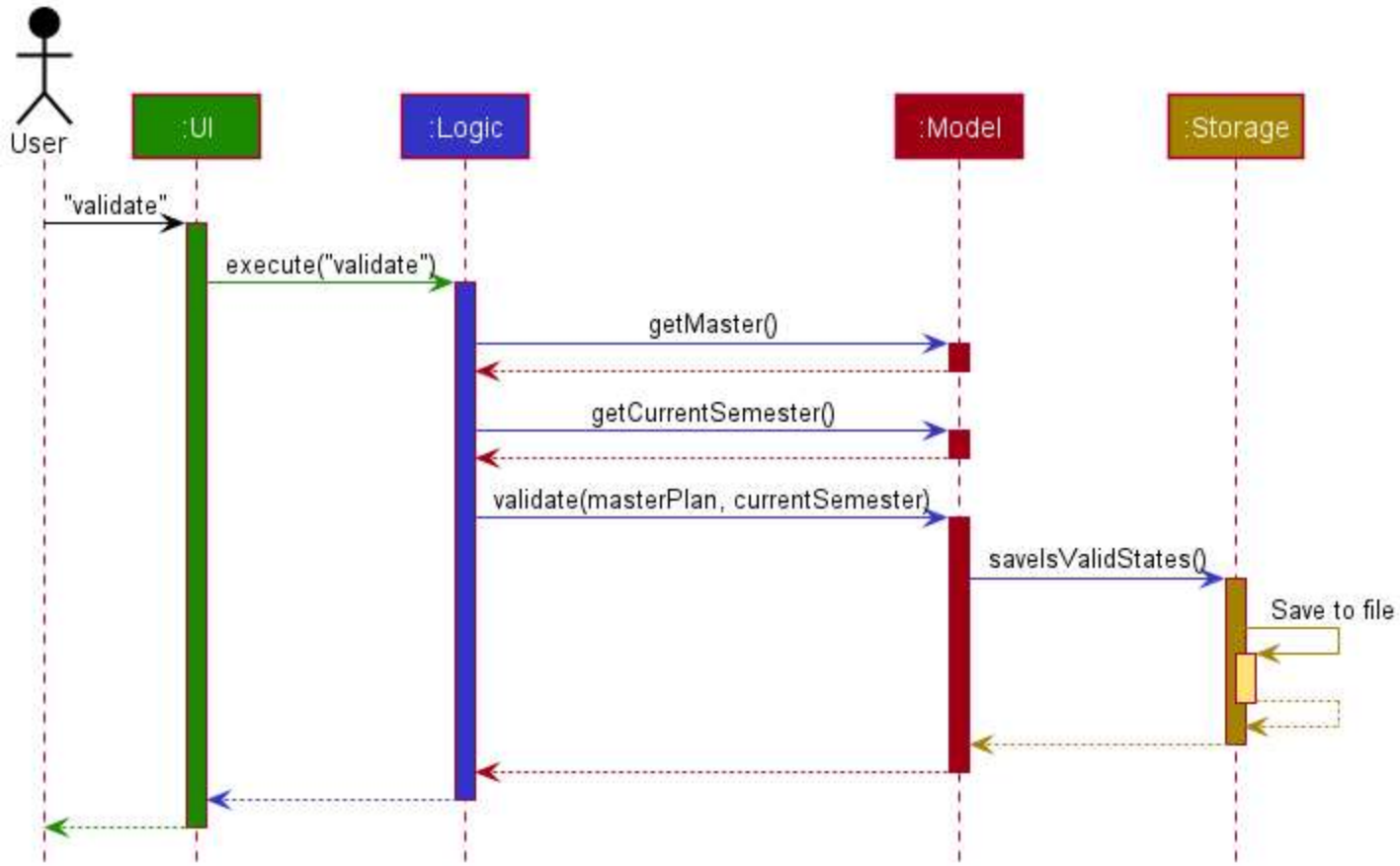
Do note that the current implementation always creates a new `History` instance whenever the `history command` is provided by the user, to ensure that users are presented with their most updated information.

Validate feature

The `validate` command looks at all modules from the `master` plan, specifically all semesters from the first to `current`. Every other plan is then looked at the same way up to whatever semester that `current` is set to. For example, if `current` is set to semester 5, validate takes the first 5 semesters of modules from the master plan and compares it to every other plan. If the other plans have every module that was found in the master plan, it is valid.

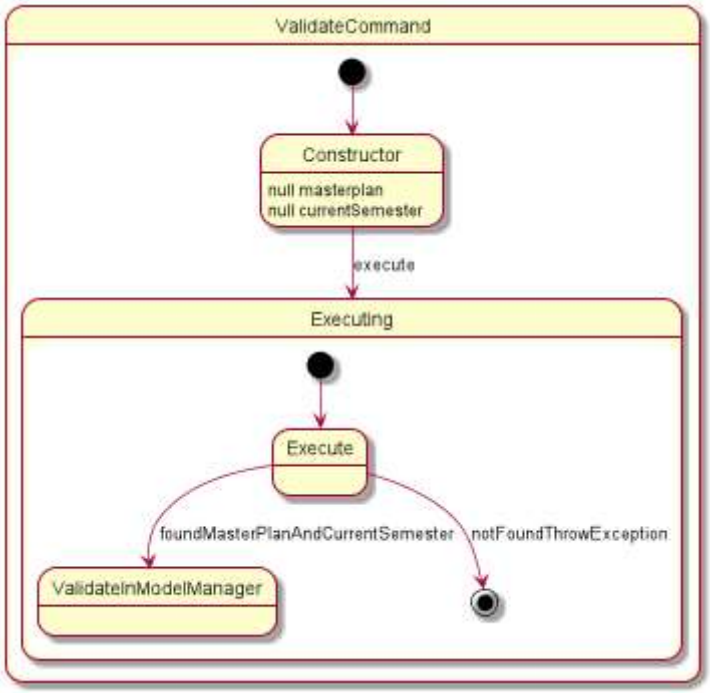
Implementation

Overview: Validate command

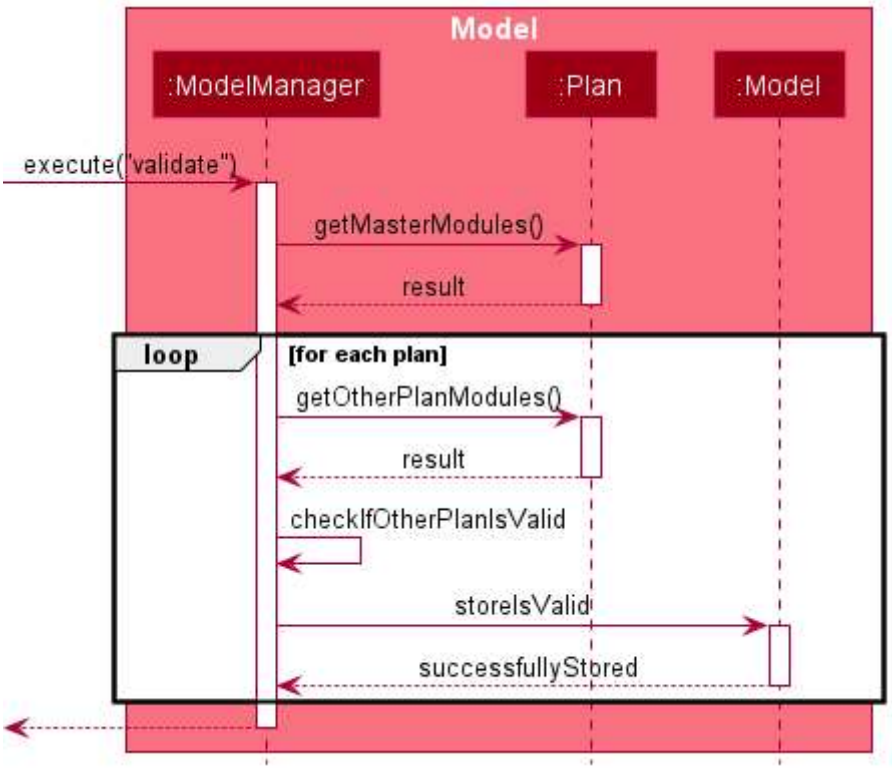


The `validate` command makes use of the `ModelManager` class which contains the necessary list of `plan` objects, which contains the relevant `semester` and `module` objects. After accessing the ValidateCommand logic component, if `master` and `current` is set, the `ModelManager` is accessed.

State Diagrams



The `validate` command does not create other objects. It initializes with a null `masterPlan` and `currentSemester`, and updates them in the `execute()` method, throwing exceptions if they have not yet been set. If set successfully, validation continues.



In the `ModelManager`, the `validate(masterPlan, currentSemester)` method references every other existing `Plan` from the base master plan. The method then sets whether or not they are valid.

[Proposed] Validating using History object