Jel Lim – Project Portfolio

Overview

JelphaBot is a desktop Task Manager. The user interacts with it using a CLI, but it has a GUI created with JavaFX. It is written in Java, and has about 20 kLoC.

Summary of contributions

- Code contributed: [View on RepoSense]
- Major enhancement: Added the start and stop a timer feature for tasks as well as a viewing panel for the user's overall productivity for the day.
  - What it does: `start <index>` command allows the user to start a timer for the task specified by the index provided. The user may stop this timer with `stop <index>` command. The productivity panel will show the total time spent on as well as the number of tasks completed that are due today.
  - Justification: If the user wants to gauge the amount of time spent on their tasks and how many tasks they have completed or an overview of their productivity in that day, it is easy to view all this information in one tab.
  - Highlights: This enhancement works with existing as well as future commands. An in-depth analysis of design alternatives was necessary to decide the layout of the productivity panel and how effective it was in improving overall user experience. The implementation was also challenging as it was difficult to decide how much data to present to users.
  - Credits: Stop and Start command classes were adapted from existing commands such as Edit command. Command line history was adapted from existing implementations.
- Minor enhancement: Added a command history that allows the user to navigate to previous commands using up/down keys.

Other contributions:

- Issues management:
  - Out of a total of 95 issues from versions 1.0 to 1.4. I managed 42 of them on GitHub (examples: #163 #263).
- User stories management:
  - There was a total of 30 user stories, for versions 1.0 to 1.4. I managed 10 of them.
- Documentation:
  - Developer Guide:
    - Section 3.1, Delete Sequence diagram, whole section of 3.3, 4.1, 4.4, 4.7
    - Appendix C, Use Case diagram, whole section of appendix C8, C9, D, E, F7, F8, F9, F10
  - Language improvements to User Guide : PR #283, #300, #316

    - Replaced existing UML diagrams to ones that corresponds with our final product: PR #198, #326
    - Added Use Case diagram and Activity diagrams for start command: PR #326
    - Added Sequence diagram for stop command: PR #328
- Enhancement to existing features:
    - Added DateTime model to Task model to integrate deadlines.
    - Added
- Community:
    - Reviewed Pull Request: #19
    - Reported 13 bugs and offered suggestions during PE dry run: issues
    - Reviewed other team's DG: (examples: 1, 2, 3)
- Tools:
    - Integrated CI/CD tools (Appveyor and Coveralls) to the team repository.

Contributions to User Guide (Extracts)

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

Productivity Tracking (Jel)

JelphaBot also comes with a productivity tracking that allows you track the progress of your tasks in that week. You would be able to see the progress bar as you complete more tasks!

Track productivity : `productivity`

Apart from the function to switch tabs by pressing kbd:[Ctrl] + kbd:[tab] on your keyboard, you can enter the `productivity` command or its shortcuts `:P` or `:p` to manually switch to the productivity tab. The productivity panel will then show you your productivity for the day.

Format: `productivity`
Shortcut: `:P` or `:p`

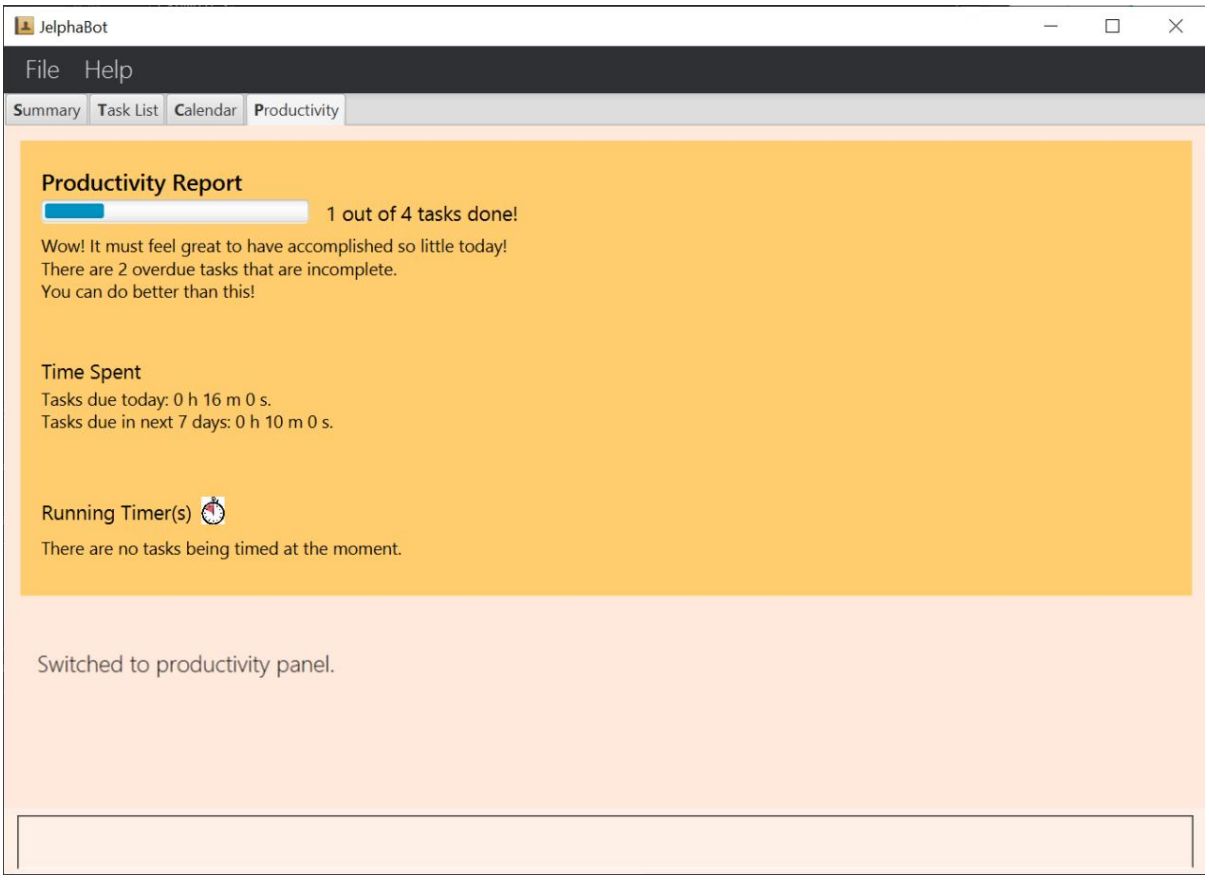

Figure 1. Example of expected result after running `productivity`

NOTE	The progress bar and the text following it only shows tasks that are due on the day JelphaBot is running.

Starting timer for a task : `start`

You can start a timer running for a task.
Format: `start INDEX`

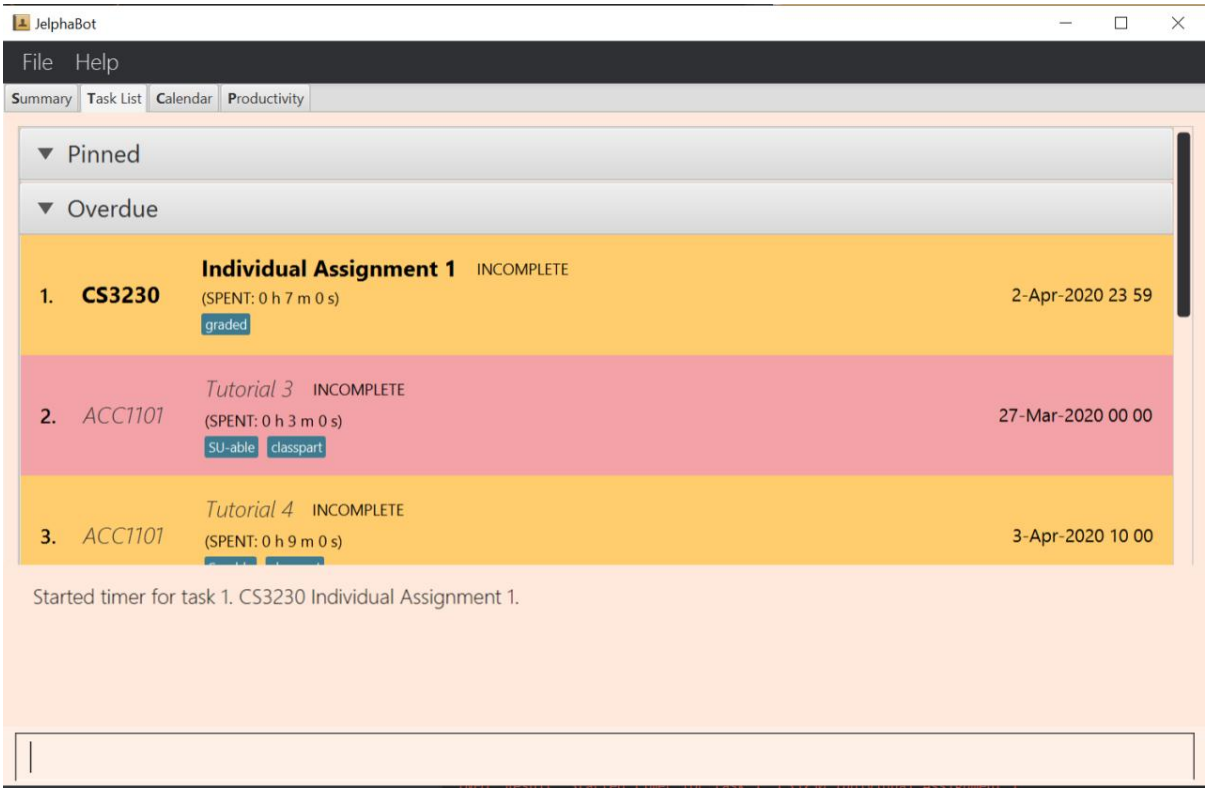

Figure 2. Example of expected result after running `start 1`

- Each task can only have 1 running timer.
- Starts the timer for the task at the specified `INDEX` if timer was not already running.
- The index refers to the index number shown in the displayed task list.
- The index **must be a positive integer** 1, 2, 3, …
- The task to be timed cannot be a completed task.
- New timer entry under "Running Timer(s)" in the productivity tab will be added if execution is successful.

Stopping timer for a task : `stop`

You can stop a running timer for a task.
Format: `stop INDEX`

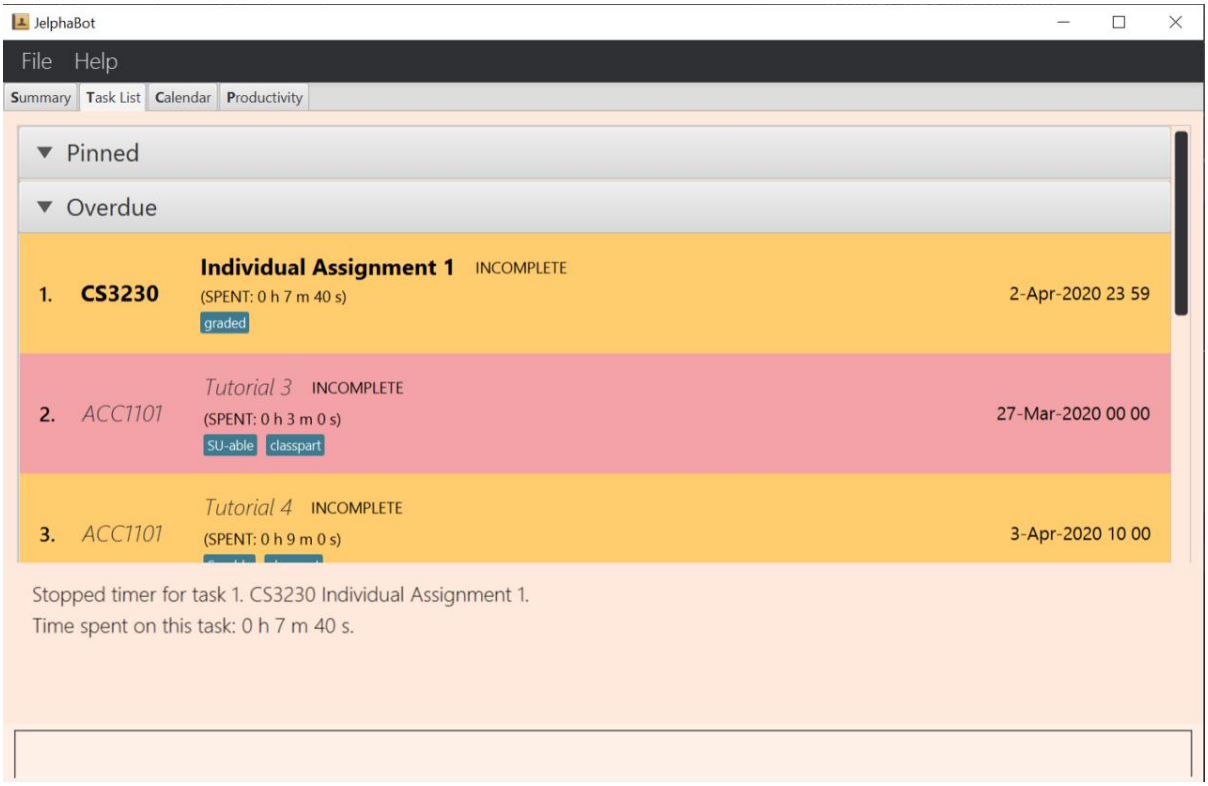

Figure 3. Expected result after running `stop 1`

- The task has to have a running timer.
- Stops the timer for the task at the specified `INDEX` if timer was running.
- The index refers to the index number shown in the displayed task list.
- The index **must be a positive integer** 1, 2, 3, …
- Timer entry under "Running Timer(s)" in the productivity tab will be removed if execution is successful.
- Time spent on timed task will be added to the respected time spent section in the productivity tab.

Receiving encouragement and criticism

JelphaBot automatically tracks the user's productivity in a day and outputs the appropriate response to the user's achievements and task completion rate.
There is no need to manually request for compliments or criticism.

Contributions to Developer Guide (Extracts)

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

Productivity feature (Jel)

JelphaBot has a productivity panel of this feature which provides an overarching view of user's overall productivity.

This feature offers two main functions and one panel for visualisation:

- Start timer for a task.
- Stop running timer for a task.
- Productivity panel under Productivity tab.

Implementation

Function 1: Starts timer for a specified task
In order to start timing a task, the user enters `start INDEX` command (e.g. start 1)

Upon successful execution of the command, the productivity tab displays the task being timed under the Running Timer(s) header.

The following diagram shows the sequence flow of `start` which modifies the current Productivity List:

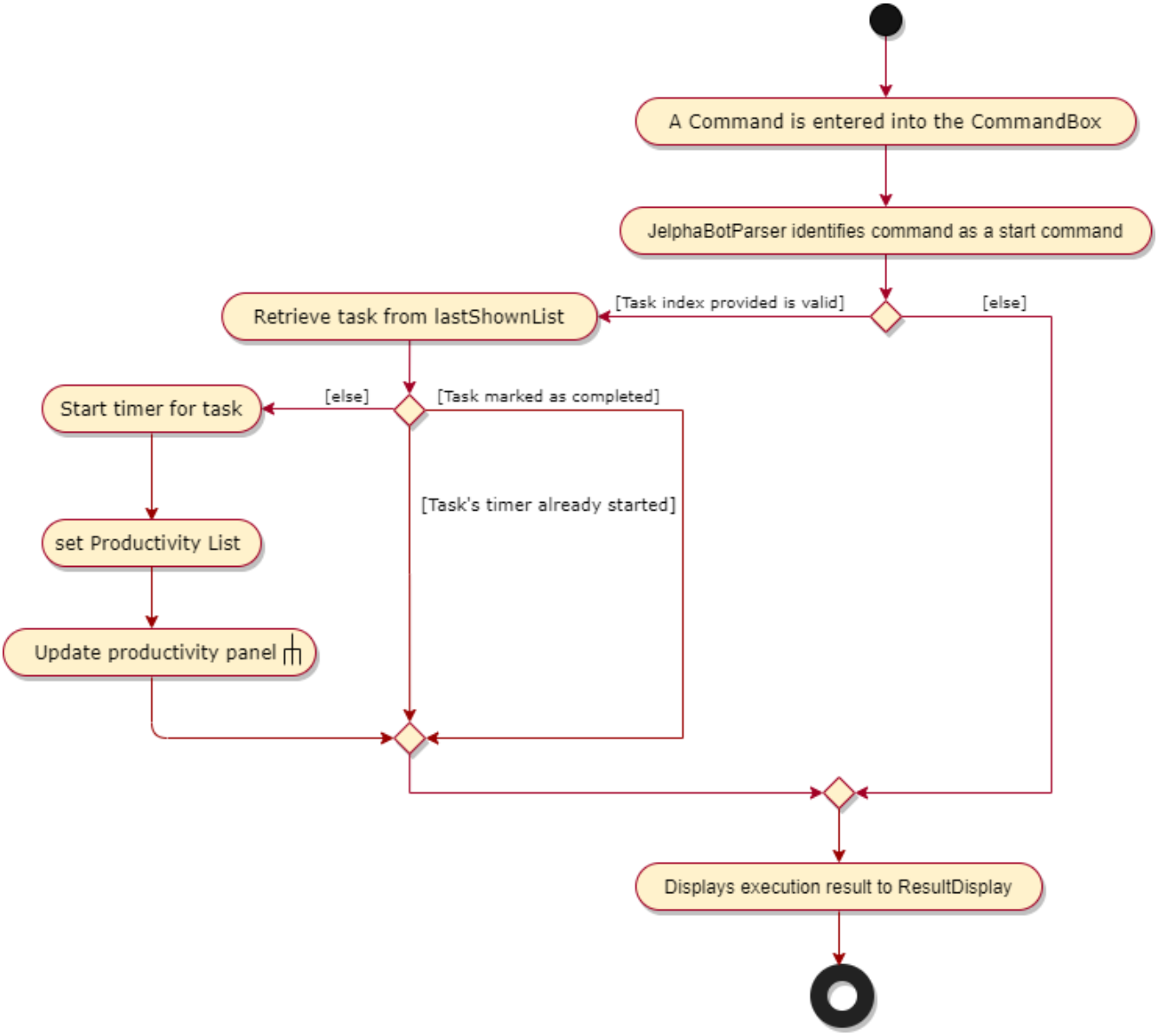

Figure 4. Activity Diagram showing the setting of Productivity in the Productivity List

Update productivity panel:

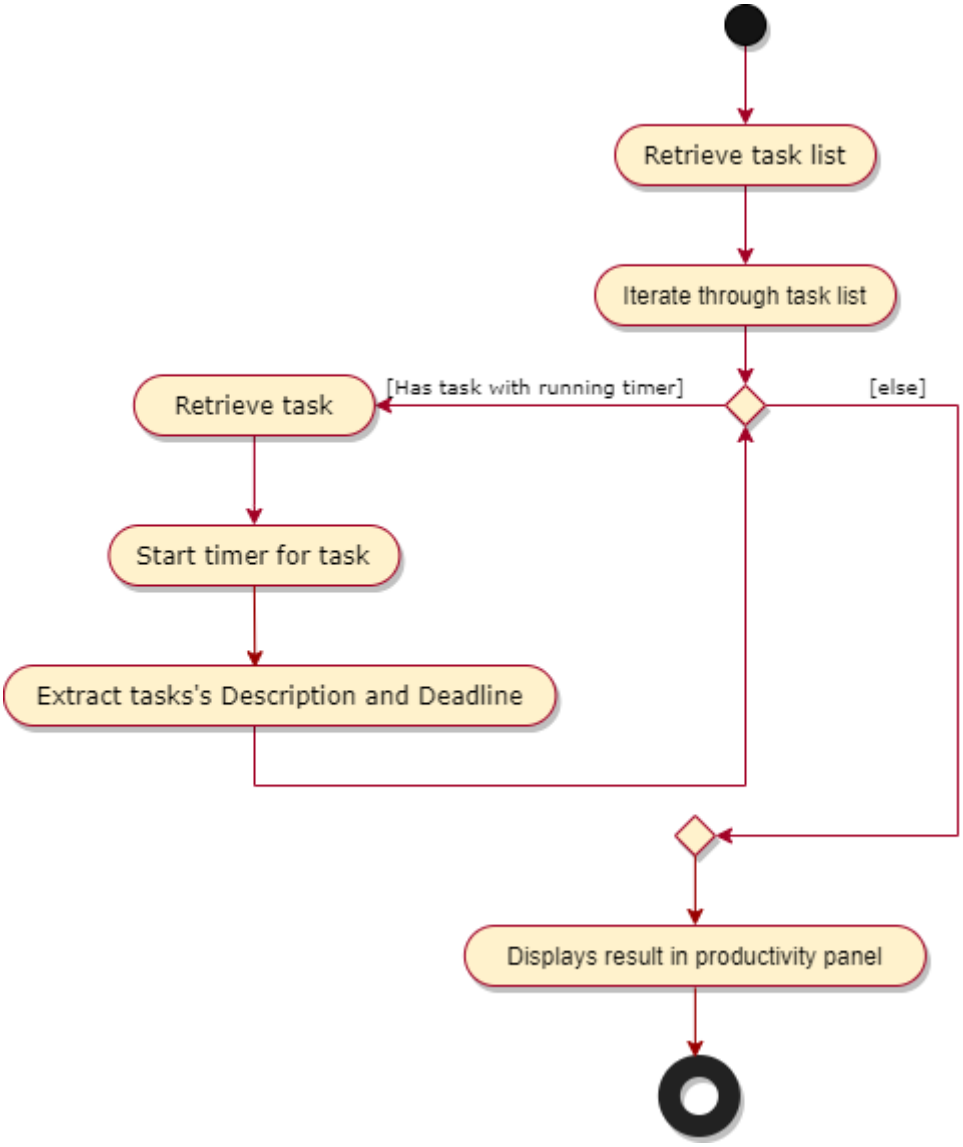

Figure 5. Activity Diagram showing the updating of the productivity panel

Function 2: Stops timer for a specified task
In order to stop timing a task, the user enters `stop INDEX` command (e.g. stop 1)

Upon successful execution of the command, the productivity tab removes the task being timed under the Running Timer(s) header. Under the Time Spent header, the total time spent will be increased depending on the date that the task is due.

NOTE	If the user attempts to start timer for a task marked as completed or stop a task that does not have a running timer, the command fails its execution so that it does not execute that start or stop operation to start or stop the timer for that task.

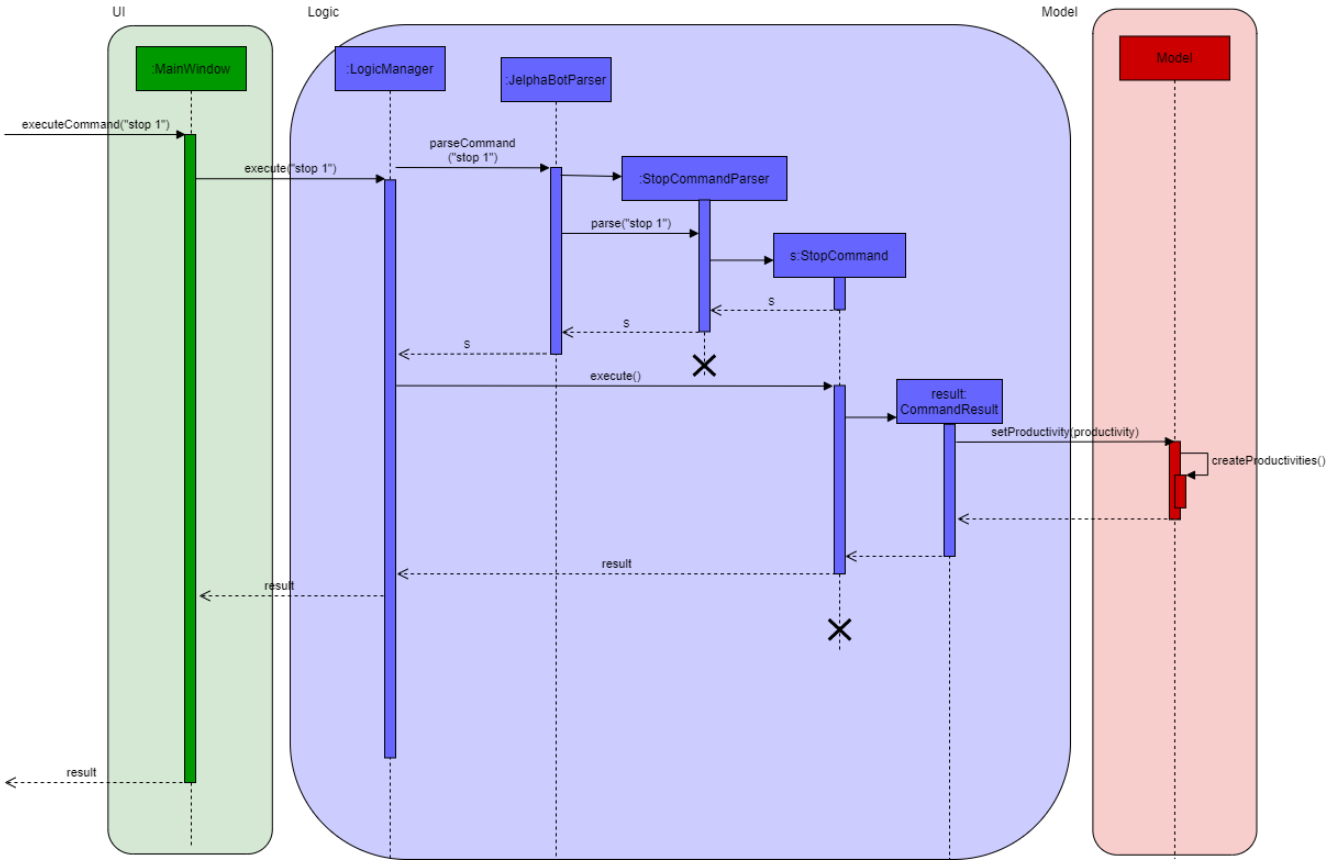

Figure 6. Sequence Diagram after running `stop 1`

Design Considerations

Aspect 1: Rendering sub-parts of productivity panel

- **Current solution**: Render each sub-part (i.e. task completion rate, time spent and running timers) only when that part needs to be updated. All 3 parts are rendered on to the same card.
  - Rationale: No need to re-render all 3 parts when changes are made to only one part.
  - Pros: Easy to implement and reduces waste of computational power.
  - Cons: As all parts are displayed on the same card, if there happens to be problem in other parts of the card, all parts will be affected.
- **Alternative 1**: Abstract each part to a separate card and render all cards onto the same panel.
  - Pros: Allows other parts to be rendered even when there is error on one part. Additionally, it is easier to identify bugs when there is an error in displaying.
  - Cons: Difficult to implement as current view is generated from a ListView but with a single card. Thus, abstracting and refactoring will be costly and hard to debug.
- **Alternative 2**: Employ multi-threading for rendering each sub-part.
  - Pros: No need to use 3 different booleans when updating view. Code base will be cleaner and more readable.
  - Cons: Unsure if cost of multi-threading less then of constructing 3 instances for rendering the productivity panel view.

Aspect 2: Allowing tasks to be added, deleted or edited while timer is running

- **Current solution**: Adding and deleting of tasks are allowed. However, tasks cannot be edited.
  - Rationale: Adding and deleting tasks does not affect the task being timed.
  - Pros: Other functionality are still available for use. Thus, user's experience is not affected.
  - Cons: User is unable to make changes to the task being timed.
- **Alternative 1**: Allow users to edit task while timer is running.
  - Pros: User is able to use all features without restriction.
  - Cons: Difficult to implement as the Task model requires a new Task to replace the old Task when edit command is executed.

Aspect 3: Productivity panel visualisation

- **Current solution**: Separating sub-parts by paragraphs and including progress bar for tasks completed.
  - Rationale: Paragraphing increases readability and the progress bar provides visual aid.
  - Pros: Easy to see at a glance which parts are which.
  - Cons: Text under Running Timer(s) can appear wordy. As number of running timers increase, more text is added under Running Timer(s).
- **Alternative 1**: Highlight displayed module code and deadline in alternating colours
  - Pros: Visually more appealing and looks less like a long list is tasks thus motivating the user to complete his/her tasks.
  - Cons: Does not resolve the issue of having too many words under the sections.
- **Alternative 2**: Only show 3 tasks whose timers were started in order of priority and time when timers were started.
  - Pros: Allows user to focus on tasks at hand.
  - Cons: User might forget about other tasks whose timers were started and not complete them on time.