Jérôme Huang (huangje) - Project Portfolio

PROJECT: Le Duc

Overview

Le Duc is a task management application geared to help high school students manage their academic and extracurricular workloads. The user enters commands via the command line, and is able to perform a vast array of operations.

Summary of contributions

- **Major enhancement**: added **the ability to customize the application**
  - What it does: allows the user to customize the application by adding/changing the shortcut and by changing the language. The user can also change the application welcome message.
  - Justification: this feature give the possibility to the user to make the application more personal. Moreover, changing the language allow international student to use it.
  - Highlights: For now, only the language, the welcome message and the shortcut can be customized. But more will come in the future.
  - Credit: The customization of the welcome message was made by SHAH RAHUL.
- **Minor enhancement**: added the possibility to add recurrence for homework and for event.
  - what it does: allow the user to add several times the same task, with only the date apart
  - Justification: sometimes, the user have the same task, but only the day differ, so this feature facilitate the user's life.
- **Minor enhancement**: added **the ability to show task by day/dayofweek/week/month/year**
  - What it does: allows the user to see tasks for a specific day, week, month, years.
  - Justification : this feature make the user's life easier because he can see which day has which task or which week has which task ...
- **Minor enhancement**: added **the ability to see all the command (help command)**
  - What it does: allows the user to see all the existing commands.
  - Justification: This feature help new user to get familiar with the application by showing all the commands with the description.
  - Highlights: This enhancement allow the application to become more user-friendly. As new commands will be added, the help command will be completed.
- **Minor enhancement**: solved the date conflict for the event task.

- **Code contributed**: [Project Code Dashboard][Help feature] [Customize Shortcut feature] [Show feature] [Customize Language Feature 1][Customize Language Feature 2][Customize Language Feature 3][Customize Language Feature 4][Recurrence Feature 1][Recurrence Feature 2]
- **Other contributions**:
  - Project management:
    - Managed releases `v1.1` `v1.3`
  - Review the code:
    - Review the code and modify a big part of it (Pull requests #10)
    - PR review(Pull requests #73)
  - Write test:
    - Junit EventCommandTest (test)
    - Junit ShowCommandTest (test) *
  - Documentation:
    - Wrote contents of the User Guide (most part of the feature)
    - Wrote the Developer Guide (after a meet up)
    - Wrote the AboutUs

Contributions to the User Guide

Overview: For the user guide, I wrote the feature I have implemented : shortcut, language, help and show. Moreover I wrote the FAQ. You can see our User Guide here.

- **Implementation for language:**

Change the language for all the display message : `language LANGUAGE`

`LANGUAGE` can be en or fr (only two language is available). The language will be set after the program is closed and open again. Example :

- `language fr`
- `language en`

- **Implementation for shortcut:**

Give shortcut to command : `shortcut COMMANDNAME SHORTCUTNAME`

COMMANDNAME is the name of the command (like todo, sort, show ...)

SHORTCUTNAME is the new shortcut name for the command

There are three behaviour : One line command, multi-step command, multi-step customize all command

Be careful :

- There can't be 2 same shortcut name.
- The shortcut name can't be the same as one of the default command name (for example, the shortcut name can't be todo because it is a default command name).

Example (One line command) :

- `shortcut todo t`
- `shortcut prioritize prio`

Example (multi-step command) :

- `shortcut todo`
- The program ask to enter the shortcut
- `t`

Example (multi-step customize all command)

- `shortcut`
- The program enter in customize shortcut mode
- The program display the first command with his shortcut and ask to enter for a new shortcut name
- `shortcutname`
- The program display the first command with his shortcut and ask to enter for a new shortcut name
- `shortcutname2`
- ...
- The program display all the shortcut

Be careful :

- There can't be 2 same shortcut name. If we enter a shortcut that already exists, we are in one line command or multi-step command behaviour, it will show an error, and if we are in multi-step customize all command, it will ask again.
- The shortcut name can't be the same as one of the default command name (for example, the shortcut name can't be todo because it is a default command name).

For each command, the console will output the command name.

The user (you) have to input the shortcut you want.

If the shortcut already exists, the console will tell you to assign another shortcut ---

- **Implementation for show:** Display all the task for one particular day/week/month: `show DATETYPE DATE`

DATETYPE is day, dayofweek, today, week, month, year.

The DATE argument depends on the DATETYPE

- day :
  - shows all the tasks for the given date
  - DATE : DD/MM/YYYY
- dayofweek :
  - shows all the tasks for the given day of week
  - DATE : monday, tuesday, wednesday, thursday, friday, saturday, sunday
- today :
  - shows all the tasks for the user's today
  - DATE : nothing should be written !!
- week :
  - shows all the tasks, starting from user's today to 7 days later (the last days is not included)
  - DATE : nothing should be written !!
- month :
  - shows all the tasks for the given month
  - DATE : MM/YYYY
- year
  - shows all the tasks for the given year
  - DATE : YYYY

There are two behaviour, one line command and multi-step command

Example (one line command) :

- `show day 29/10/2019`
- `show dayofweek monday`
- `show today`
- `show week`
- `show month 10/2019`
- `show year 2019`

Example (multi-step command) :

- `show day`
- The console ask to enter the day :
- `29/10/2019`

or

- `show dayofweek`
- The program ask to enter the day :
- `monday`

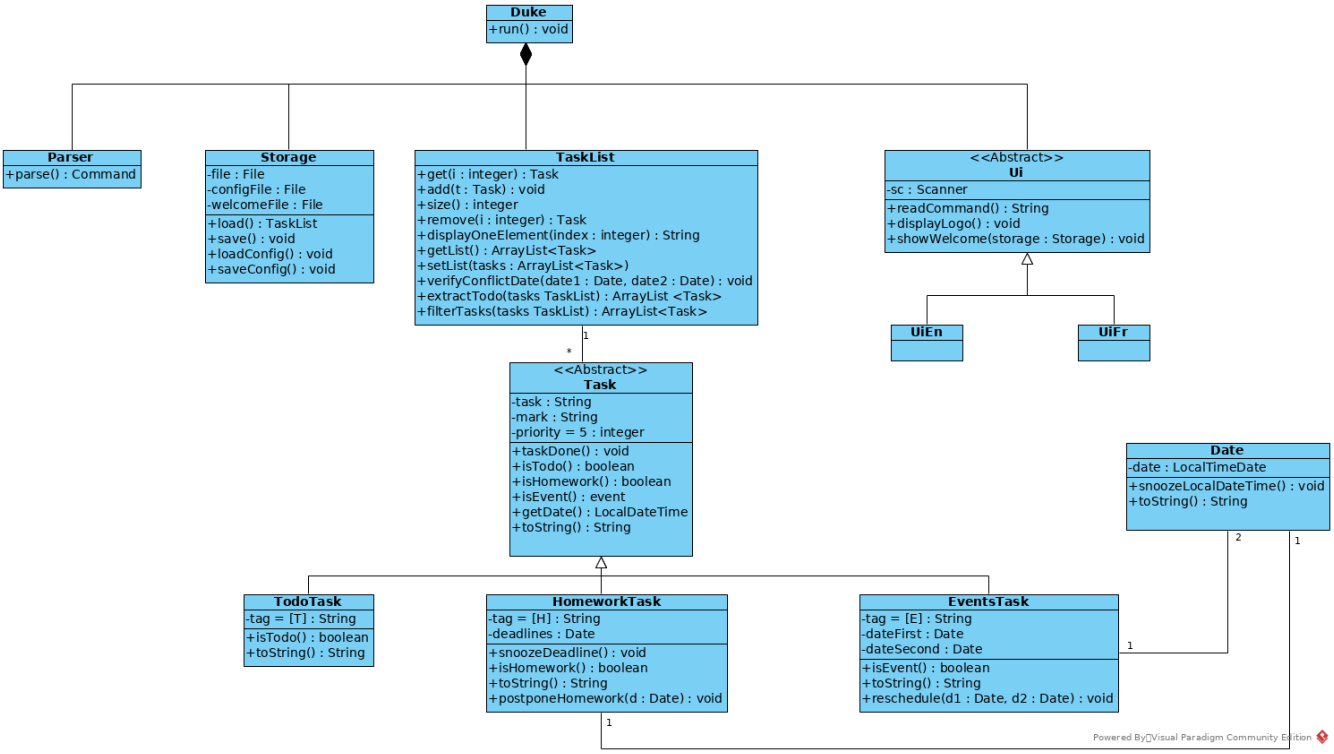
Contributions to the Developer Guide

Overview: For the developper guide I wrote the `Target User`, `User Stories`, `Use Cases`, `Non-functional requirements` and the `Glossary` after brainstorming with the team. I also wrote `Class Diagram` part and the `Customization` part.

- **A part of `Class Diagram`:**

Le Duc main class, called `Duke`, is composed of 4 classes : `Storage`, `Ui`, `Parser`, `TaskList`.

- `Storage` deals with saving and loading files such as the file containing the config or the file containing all the tasks.
- `Ui` deals with the interaction between the user and the program.
- `Parser` given an user's input (through Ui), the Parser will return the corresponding command
- `TaskList` represents the list containing all the tasks.

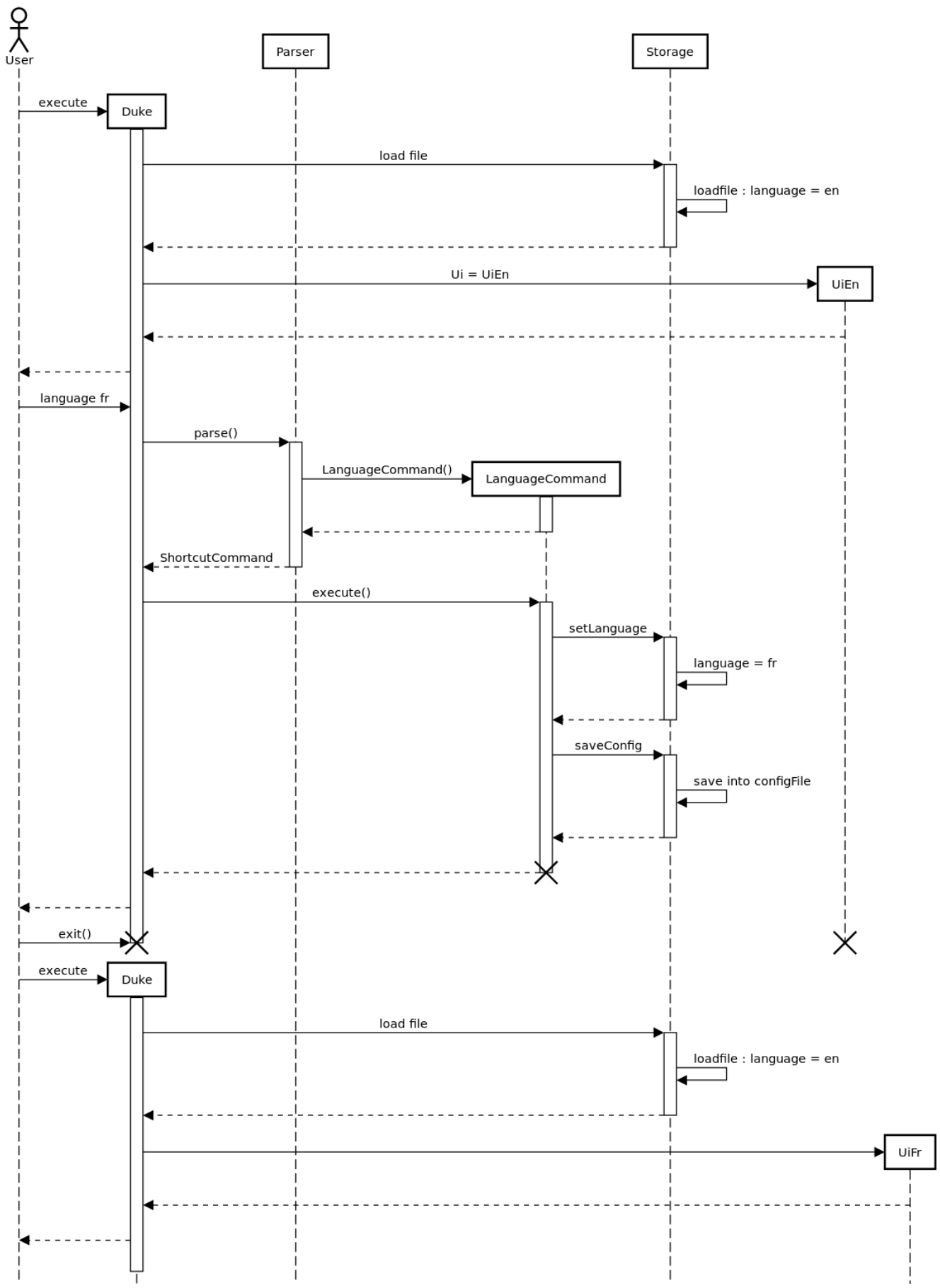


- **A part of `Customization`:**

Changing the language mechanism is done by the `LanguageCommand`. For the moment two languages are available : french and english. Only the return message after a command and the error message

are changed. After typing the command to change the language, the language is changed at the next execution of the program.

The following are the steps to change a language :



- The user open Le Duc (the program).
- Le Duc create the object `ui` as an instance of `UiEn`.

- The user type `language fr` (the program is previously in english)
- The program will change the config file.
- The user exit the program.
- The user reopen Le Duc.
- Le Duc load the config file with the new language.
- Le Duc create the object `ui` as an instance of `UiFr`.
- The language of Le Duc is french.

In the sequence diagram, `Parser` and `Storage` should be created and destroyed when Duke is created or destroyed, but for more clarity, it was not represented.

Consideration

- (Current implementation) Each message displayed to the user (error or a message returned by a command) correspond to an abstract method in `Ui` and an override method in `UiFr` and `UiEn`. It was done so because it is easier to add a new language because it is sufficient to create a new class and override the method.
- (Atlernative) Make an if statement for each new language and an static attribut in `Ui`. There are less methods and less classes but if a new language is added in the future, every single command and every single exception have to be edited.
