[T14-2j][V0.2]

My Hot Secretary Project Manual

			
Timothy Lim Team leader Database	Shekhar Test master Processor/Logic	John Integration Ui & Google Calendar	Kahou Cheong Documentation Command Parser

Contents

User Guide

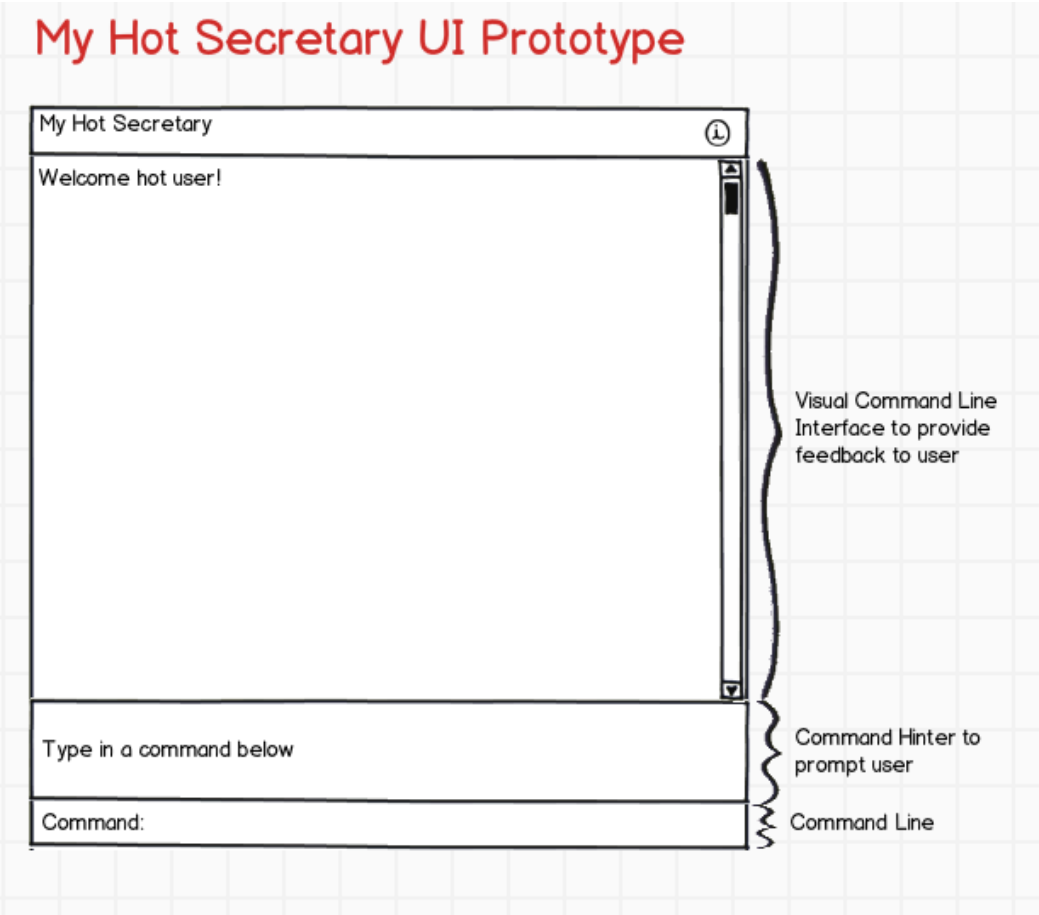


Figure 1 : UI Prototype

My Hot Secretary

My Hot Secretary is a personal text-based scheduler. It helps to schedule and manage tasks in an organized manner. It is made for people who are fast typists and who prefer typing over mouse and voice commands.

How do I install it?

My Hot Secretary does not need any installation, just download and run!

What does My Hot Secretary do?

My Hot Secretary is a scheduler that has support for:

- Timed tasks (Tasks to be done over a period of time)

- Deadline tasks (Tasks to be done before a certain time).
- Floating tasks (Tasks without allocated time or date).

Core Features

Tasks operations:

- Create
- View
- Update
- Delete
- Mark
- Undo

Search

- Search for a task.
- Search for tasks on a day/date.
- Search for tasks within a date range.

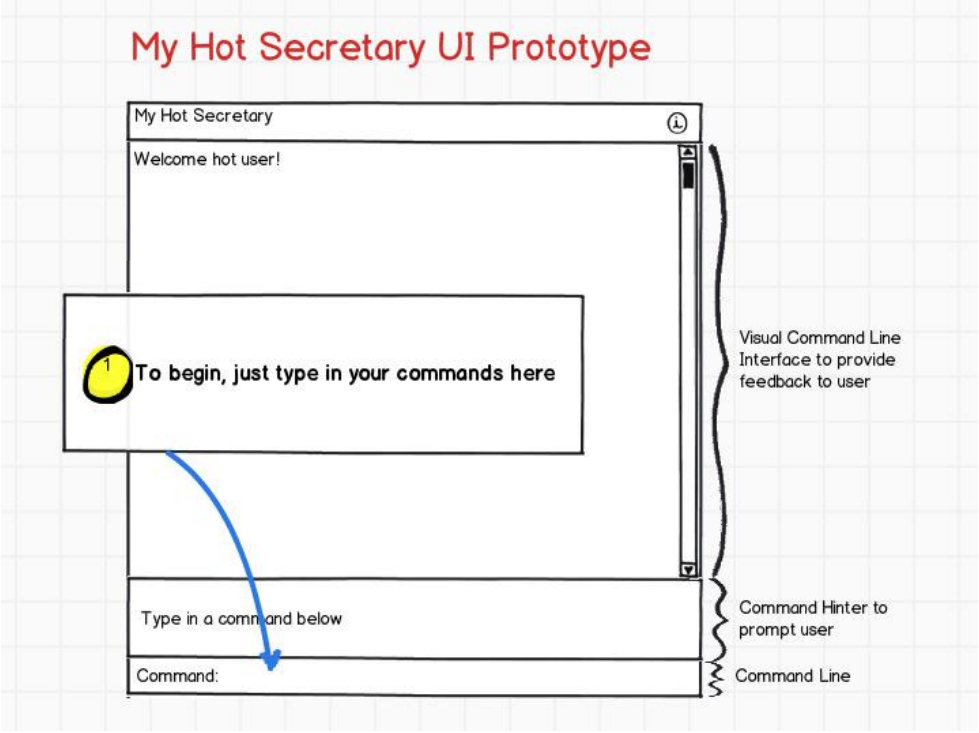
Google Calendar Integration (Requires internet connection):

- Automated synchronization of tasks with Google Calendar.
- Automatic pushing of tasks after every operation.
- Automatic pulling of tasks from Google calendar every 5 minutes.
- Manual pulling of events from Google calendar.

Flexi-commands

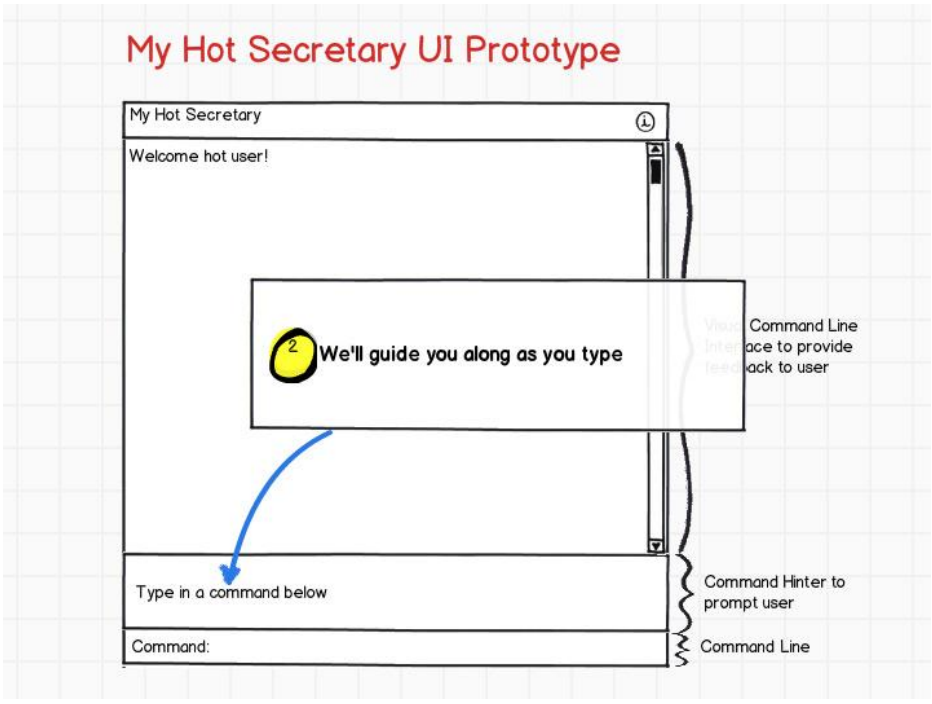
- Natural language recognition
- Recognize multiple command formats
- Support for undo/redo of previous actions

Quick Visual



Guide

Figure 2 : UI Prototype Command input



-

Figure 3 : UI Prototype Command Hinting

[T14-2j][V0.2]

The Command Guide

*Command parameters in [ ] brackets are optional, while parameters in < > brackets are needed to complete the command.

Add

To create an event you need to specify ‘What’ and ‘When’ (Only ‘What’ is required)

- What: This can be any text; the event name is created from this.
- When: This can be any date and/or time expression.

Supported Datetime Formats

- Day: Mon/Monday
- Month: January/jan
- Year: 2012
- Time: 3pm /5am/15:00
- Date: 30/9, 30/9/12, 30/9/2012
- Date Ranges : this weekend/ this month/ this year/ this week
- Other keywords: today, tomorrow

Timed Task

[add] <task name> [from] <date time> <to> <date time>

- add cs1103 tutorial from 10/9 3pm to 4pm
- vacation 10/12 5/1
- shopping Tuesday 3pm - 4pm
- learn java tomorrow (whole day task)

Deadline Task

[add] <task name> [by] <datetime>

- project submission by Wednesday (means by 11:59PM coming Wednesday)
- learn to cook by tomorrow (means by 11:59PM tomorrow)

Floating Task

[add] <task name>

- add play Pokémon
- do laundry

[T14-2j][V0.2]

Display / Search

Display uses the ‘display/ ‘search’ commands to display all matching results.

By Day/Date –

<display> <datetime>

- show 12/7, 12/7, today
- show this weekend
- show today to tomorrow

By Task

<display> <task name>

- show laundry

Update

Rename task

<rename> <task name search string> <to> <new task name>

- rename learn java to learn C++

If multiple tasks match given task name, all matching entries are displayed with numbering.
User can then choose which entry to rename by specifying the number.

Reschedule task

<edit> <task name> [to] <datetime>

- edit laundry to 5pm

If multiple tasks match given task name, all matching entries are displayed with numbering.
User can then choose which entry to edit by specifying the number.

Mark task as done

<mark> <task name>

- mark assignment

If multiple tasks match given task name, all matching entries are displayed with numbering.
User can then choose which entry to mark as completed by specifying the number.

Delete

<delete> <task name>

- delete laundry

If multiple tasks match given task name, all matching entries are displayed with numbering.
User can then choose which entry to delete by specifying the number.

Undo

<undo>

- Undo (undo the previous command).

Redo

<redo>

- Redo the previous undo.

Help

<help>

- Shows a list of the usable commands and date time formats.

Sync (Google Calendar)

[sync]

Sync command allows the user to initiate Google Calendar synchronization with My Hot Secretary**.**

First Time Synchronization

For first time synchronization, My Hot Secretary will prompt for Google Authentication or Google Account registration.
Register - User is redirected to Google Account registration via a browser.
Authentication - User is required to enter his/her email / password.

Synchronization is automated after user's very first sync command, subsequent [sync] forces manual sync.

Developer Guide for My Hot Secretary (MHS)

What is My Hot Secretary

My Hot Secretary is a desktop application that helps a user to keep track of their tasks.

Functionality summary:

1. Natural language command support
2. Command Hinting
3. Created, retrieve, update, display operations for floating, deadline and timed tasks
4. Smart Task Search (name / time)
5. Google Calendar Sync Support

Who is this guide for

This guide is for anyone who wants to work on My Hot Secretary.

Getting Started

My Hot Secretary is hosted on Google Code, the source code can be found at:
http://code.google.com/p/cs2103aug12-t14-2j/

To work on My Hot Secretary, just retrieve the code from the above repo. We recommend using Mercurial as this is the Revision Control System we are currently using.

Program Architecture

MHS utilizes an N-Tier architecture:

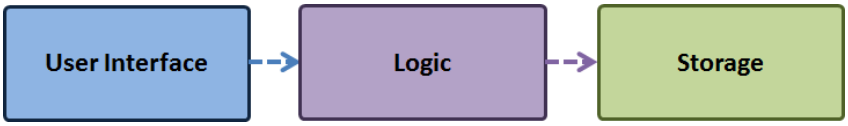


Figure 4 : MHS N-Tier Architecture

Main components are :

- **UI:** The responsibility of the UserInterface is to control interaction between the user and the logic-tier. It updates the logic-tier with user input and updates what is displayed to the user.

- **Logic:** Logic Tier handles the flow of logic between the different components of the architecture. It passes the command string over to the command parser to parse the command. It then interacts with the database to execute this command.

- **Storage:** Storage Tier interfaces persistent data storage mechanism, on local disk and remote (Google Calendar Service) and handles task queries and operations.

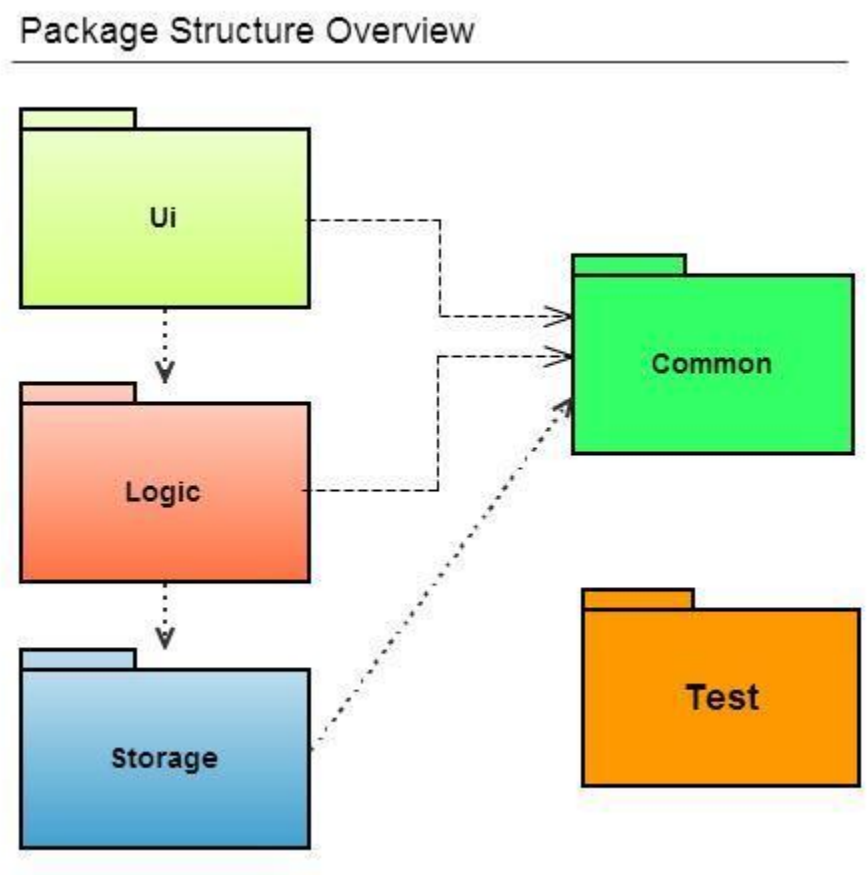


Figure 5 : Package Structure Overview

User Interface

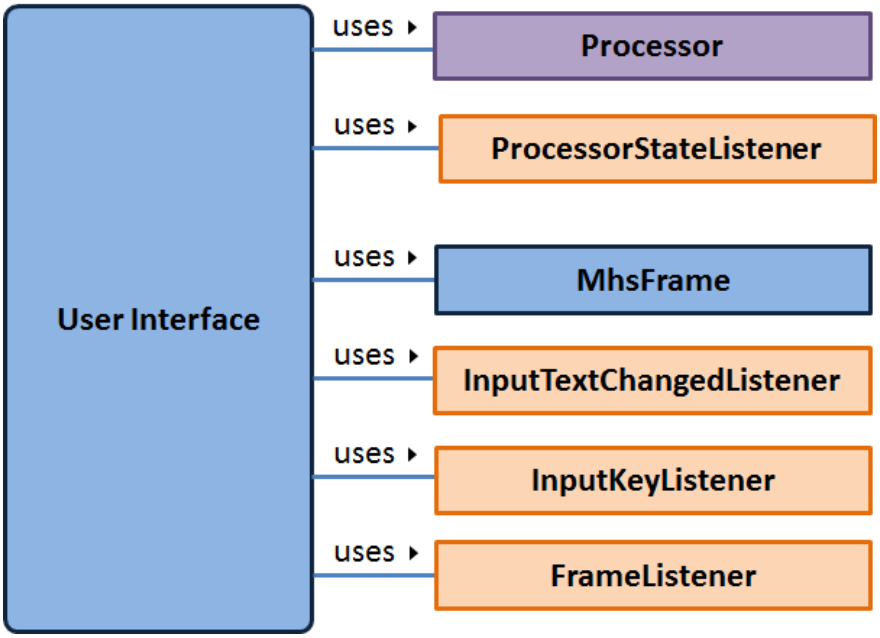


Figure 6 : UI Class Diagram

Responsibility:

The responsibility of the UserInterface is to control interaction between the Processor and the MhsFrame.

MhsFrame the responsibility of this class is to provide the means for the user to interact with the application. It comprises an area to display output, an area to display one line feedback or confirmation messages, and an area for the user to input commands. Two types of input formats are available: password input, where user input is replaced with dots, and plain text input.

Observer Pattern:

The UserInterface makes use of the observer pattern in order to ensure the Processor and MhsFrame are updated at the appropriate times.

InputTextChangedListener is used to observe when the user is typing a command, this allows UserInterface to update the processor as the user types.

InputKeylistener is used to observe when the user hits the enter key, the UserInterface then alerts Processor to execute the current command.

FrameListener is used to observe when the size of MhsFrame has changed, it then updates processor so that the maximum number of lines can be displayed.

ProcessorStateListener is used to observe when the Processor has processed a command or has changed its state, UserInterface then updates MhsFrame to display the updated state.

Logic

This is the tier containing the logic to process the user commands and executes them.

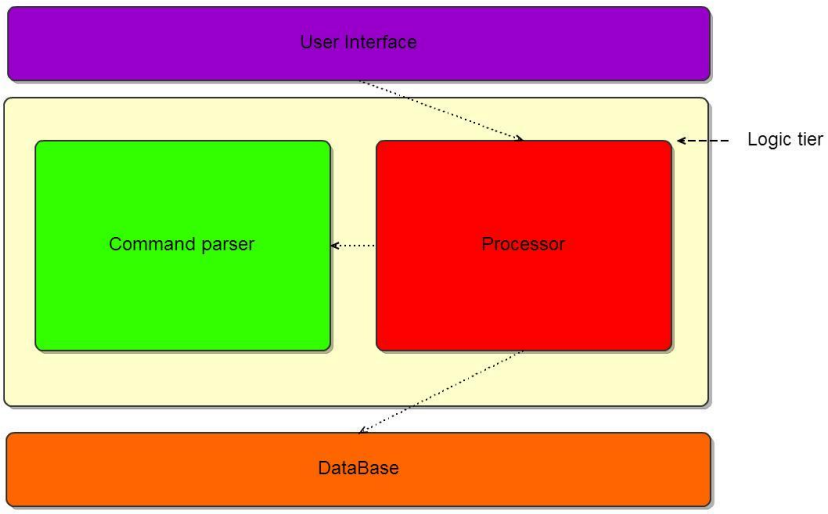


Figure 7 : Logic Overview

Command Parser

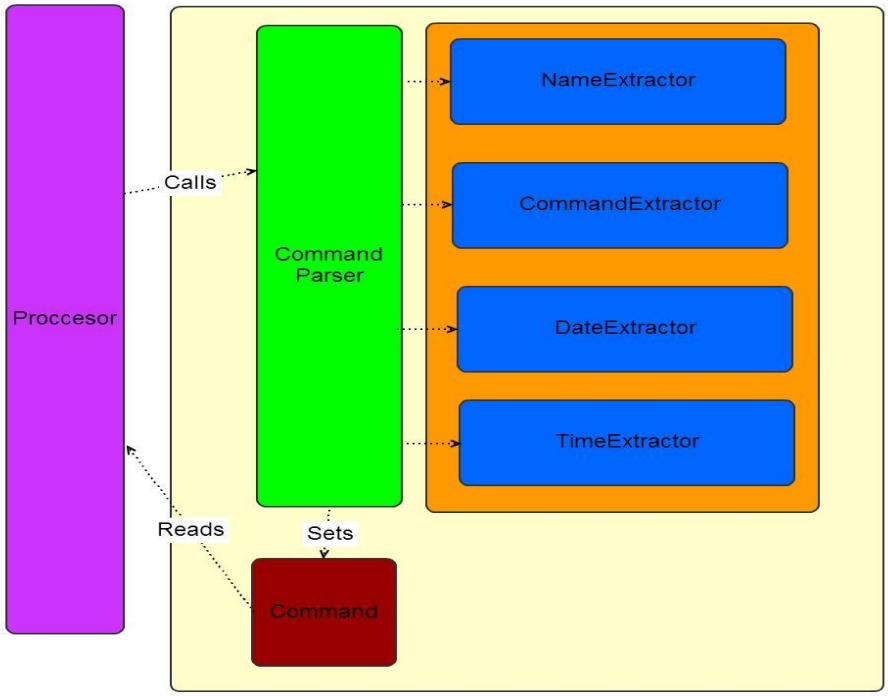


Figure 8 : Command Parser Class Diagram Overview

[T14-2j][V0.2]

Command Parser is the class that is called by the processor to parse a string and extract out the date time , command and name parameters.

Command Parser splits the string into an array of individual Strings.

Command Parser makes use of date extractor, name extractor, time extractor and command extractor to extract the keywords from the string and sets the parameters in Command.

Command Extractor

Returns the first word of the string if it matches one of the commandKeyWords.

Enumeration CommandKeyWord : This is the enumeration of all the keywords that are associated with command that the user can enter.

The String command is the command the program understands that corresponds to each keyword.

Date Extractor

Returns a list of LocalDate objects

Enumeration DayKeyWord : This is the enumeration of all the keywords that correspond to a day of the week.

The int dayOfWeek is the actual day of the week each of the keywords correspond to.

Enumeration MonthKeyWord: This is the enumeration of all the keywords that correspond to a month of the year.

The int MonthOfYear is the actual month of the year that each of the keywords correspond to.

Enumeration UniqueDateTypeKeyWord: This is the enumeration of all the date keywords that depends on the current day.

Date Extractor Flow

1. Check if string matches a date type.
2. When match is found, set up a date queue with all the strings that matches a date type in a row.
3. Stop pushing into the date queue when a non date type is found.

4 Once queue is formed, pop each string one at a time and analyse if it is a day, month , year or unique type.

5 Set the date which corresponds to the queue.

6 For unique types, get the current day and set the date or date range accordingly.

7 Check if date is valid.

8 Rectify the date if it exceeds the boundaries of a month or year.

9 Set the date and push it into a list of dates.

10 Return to the part of the string where the queue stopped pushing.

11 Find the next date type

12 If found go to 2, else return the date list.

Time Extractor

Returns a list of LocalTime objects.

Time Extractor Flow

1 Check if the string matches a time type through regex.

2 If the string matches, check if its a 12hrs or 24hrs type format.

3 Set the Local Time.

4 Push the LocalTime into the time list.

5 Find the next string that matches a time type.

6 If found go to 2, else return the time list.

Name Extractor

Returns a list of Strings which corresponds to the name types.

Name Extractor uses 2 different types of extraction.

1 Get the name from whatever is within quotation marks to avoid conflict with other types.

2 Extract the name normally.

Name Extractor Flow

For names within quotation marks:

1 Find the String within quotation marks.

2 Push the string into the name list.

3 Remove that string with the quotation marks from the string that needs to be parsed

4 Find the next String within quotation marks.

5 If found go to step 2 else return the parsed String with the strings removed.

For Normal Parsing:

1. Find Strings that are not a date, time , command or special keyword.
2. Set up a name queue with all the name types together.
3. Stop pushing when it is not a name type.
4. Pop from the queue and append the string together,
5. Push the appended String into the name list.
6. Return to the part of the string where the queue stopped pushing.
7. Find the next name type .
8. If found go to 2, else return the name list.

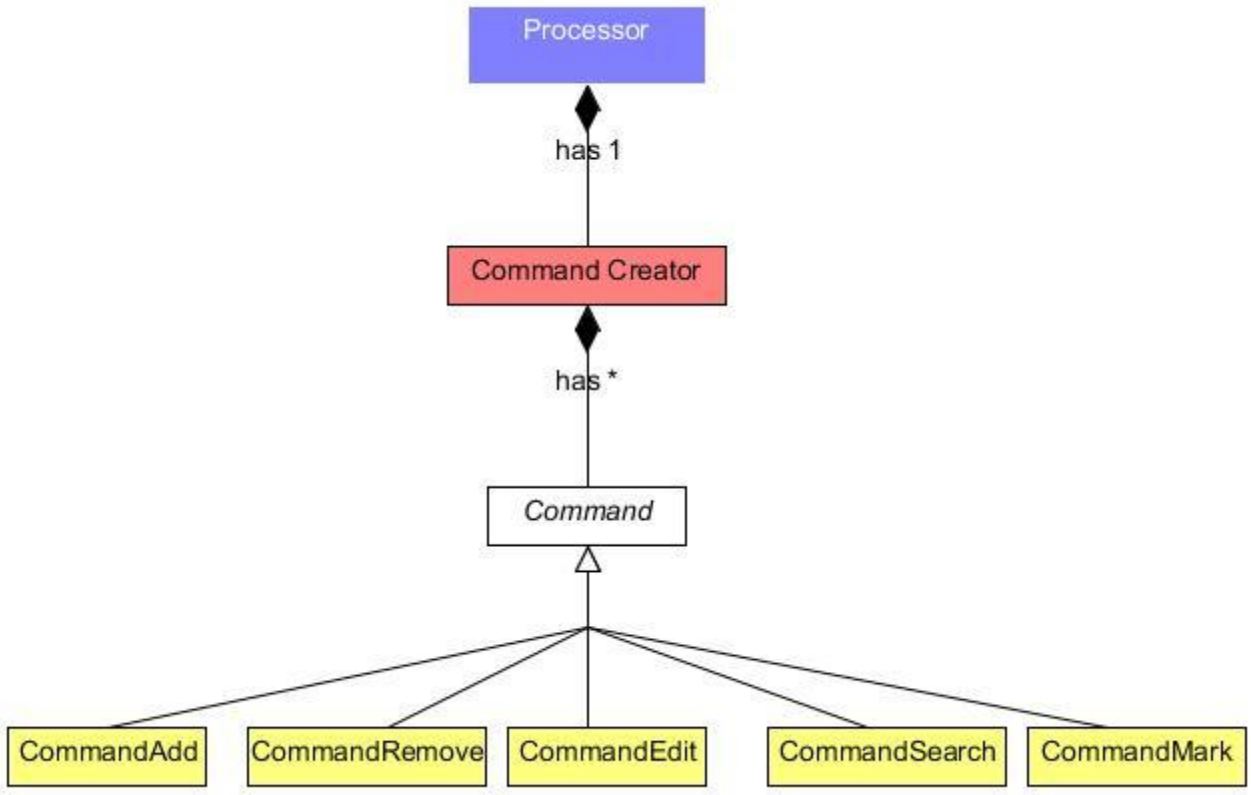
CommandInfo

This is the object that processor reads from

It sets the all the parameters passed in from command Parser and defaults the combines the LocalDate and LocalTime objects into a DateTime object.

The DateTime object will be defaulted to current date if there is a missing LocalTime or LocalDate.

Processor



Responsibility:

The processor class along with its associated Logic classes acts as the controller in MHS. It manages the flow of the user's command and leads it to execution. In this process it interfaces with the UI, Command Parser and Database classes.

General Logic Flow

Processor receives a command string from the UI. If this is a Google Login Command then the processor handles it internally and interacts with the Database to complete login and sync. Other commands are processed in associated processor classes and finally the command is completed using the database. The confirmation message flows back through all the associated classes and is updated on the user's screen.

Command Pattern

To ensure scalability and ease of development Command Pattern has been implemented in the processor and its associated Logic classes.

Flow of Logic in Command Pattern

When a command is received by the processor which is not a login/sync command, the Command Pattern logic comes into play.

1. Input string is passed to Command Parser for parsing which returns an object of CommandInfo.
2. This object is passed to a method in CommandCreator. CommandCreator creates a new Command based on the input command type.
3. Each subclass of Command has two methods execute() and undo() that handle the execution and undo operations respectively.

- Adding new Command Types
  - Add a subclass to command and implement all the abstract methods inherited from Command.
  - Add a case for that command type in Command Creator
- Reason for excluding a command queue from the command pattern
  - Since this software is being used only by one client at a time, there is no need to queue commands.

Storage

Storage's main role is to perform CRUD operations on tasks and any other persistent data. It interfaces persistent data storage operations on local disk as well as remote storage(Google Calendar Service), including syncing.

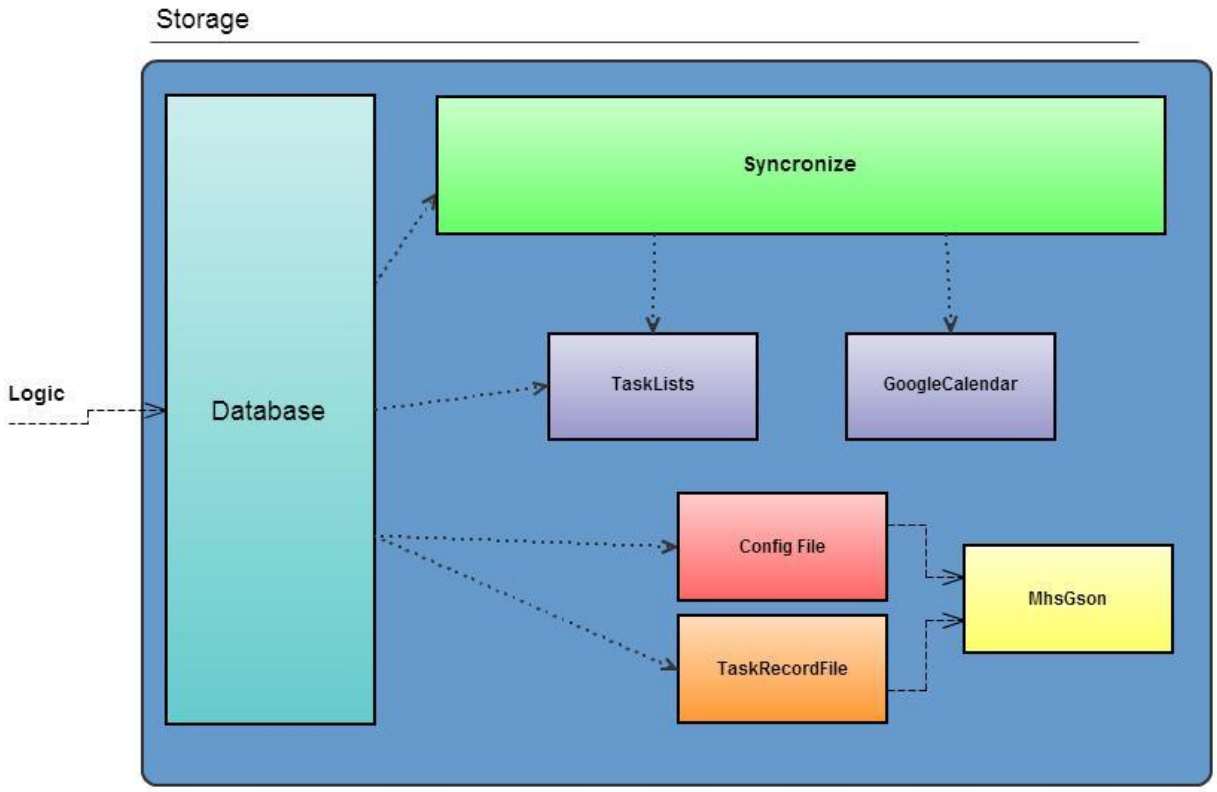


Figure 9 : Database tier overview

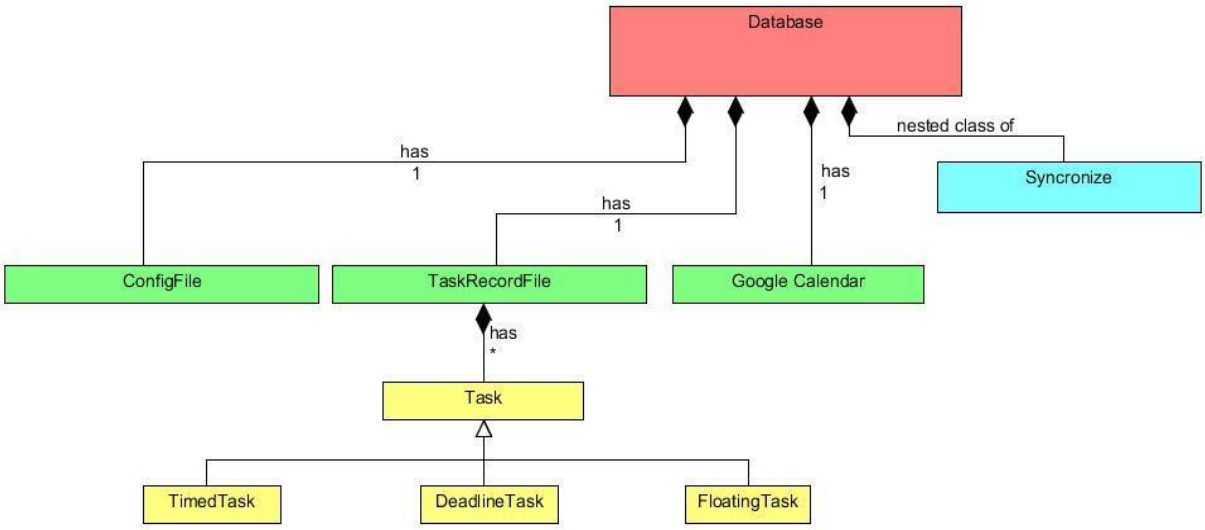


Figure 10 : Database Class Diagram

Database

Database processes incoming requests from logic and performs operations on tasks.

Tasks are stored in lists as ‘views’:

1. taskList - Mapped to task id
2. gCalTaskList. - Mapped to google calendar unique identifier (iCalUid)

Tasks are saved to file after every operation.

Syncronize

Syncronize performs synchronization of tasks between local storage and Google Calendar through a pull/push sync mechanism.

1. Synchronize pulls events from GoogleCalendar into a list (googleCalendarEvents) and this is compared with a local task list (gCalTaskList).
2. Pull Sync is performed by iterating over googleCalendarEvents list.
    a. Create new local task if no existing iCalUid exists.
    b. Update existing local task if Google calendar event is newer.
        i. Delete existing local task if Google calendar event is deleted.
3. Push Sync is performed by iterating over gCalTaskList.
    a. Create new GoogleCalendarEntry if no existing google calendar unique identifer (iCalUid) exists.
    b. Update existing GoogleCalendarEntry if local task is newer.
        i. Delete existing GoogleCalendatEntry if local task is deleted.

FailSyncSilentlyPolicy: Only UnknownHostExceptions due to internet connectivity are thrown during sync operations, other ServiceExceptions are caught and silently handled.

TaskLists

TaskLists abstracts operations on the taskLists which contains and contains for CRUD on tasks in all task lists.

TaskRecordFile

TaskRecordFile handles CRUD operations of tasks on the task record file (taskRecordFile.json) as well as file operations, such as creation, saving and loading of tasks. (from object to json and vice-versa).

ConfigFile

ConfigFile handles CRUD operations of configuration parameters on the configuration file (configFile.json) as well as file operations such as creation, saving and loading.

GoogleCalendar

This class provides methods for the application to easily create, retrieve, update and delete events in a user's Google Calendar. It utilizes the Google Calendar Data API and java code provided from:

http://code.google.com/p/gdata-java-client/downloads/list

To use this class, the user's email and password is passed into the class, this email and password is used to retrieve an access token from Google's Calendar API. The retrieved access token then associates and authorizes all future method calls with the specified user's Google Calendar.

MhsGson

MhsGson is a singleton class that creates and uses a single instance of Gson configured to work with Tasks (Timed, Deadline, Floating) and JodaTime DateTime objects.

Database Policies

Database public methods that have return type of tasks / list of tasks returns clones of task(s) so that any modification on those task(s) do not change the original task.

Adding new task

Creates new task and assigns a new unique task id, created time and updated time.

InvalidTaskFormatException thrown when task specified does not meet the required format.

Updating Task

Updates existing task's editable fields and updates created time and updated time.

Non-editable fields : CalTaskId, TaskCreated, TaskLastSync are preserved so as not to interfere with synchronization.

TaskNotFoundException thrown when task specified by taskId does not exist.
InvalidTaskFormatException thrown when task specified does not meet the required format.

Deleting Task

Deletes existing task and updates created time and updated time.

Tasks are marked as deleted and not removed for syncronization purposes.

TaskNotFoundException thrown when task specified by taskId does not exist.
InvalidTaskFormatException thrown when task specified does not meet the required format.

Querying Task(s)

IllegalArgumentException thrown when parameter is illegal.
TaskNotFoundException thrown when task specified by taskId does not exist.

Logging

Logs

Levels

Testing

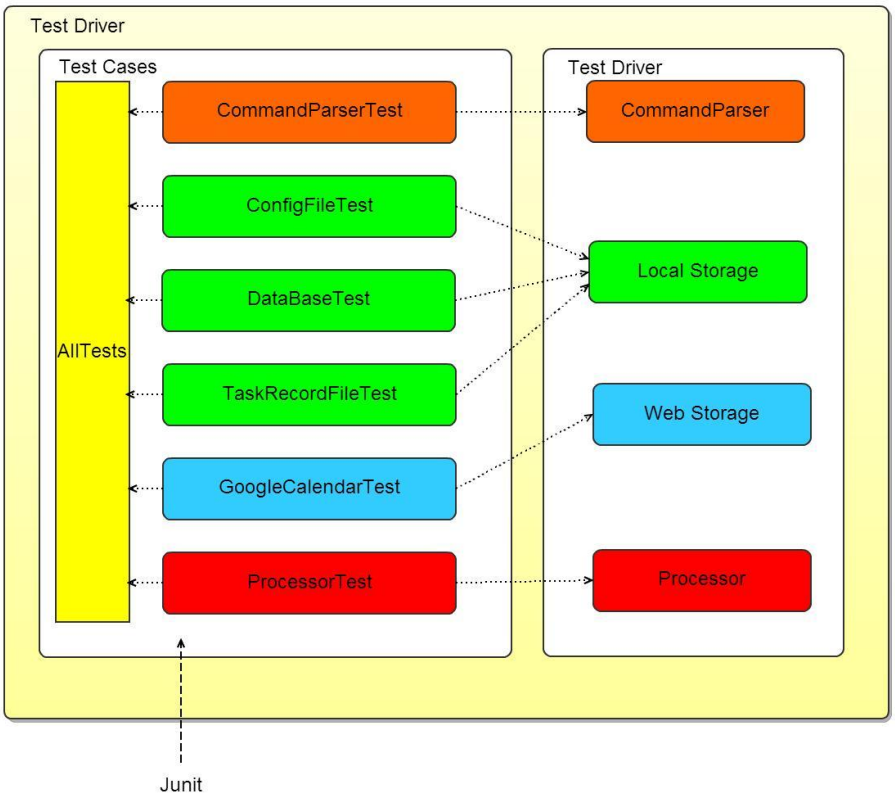


Figure 11 : Testing Package Diagram

Tests are contained with the Test Package - mhs.test.

1. **Manual Testing**
2. **System Testing**
    a. Logging – All important events in the system are logged. The entire flow of the program can be traced using the MHS.log.
3. **Unit Testing**
    - Test Suite - All Tests
        - Individual jUnit tests

Note: Database Test may have failed test cases if anti-virus is enabled since it is dependent on I/O operations, disable anti-virus before running tests.

Intergration

New features should be implemented in a separate branch. The new branch should be merged with the trunk only after it is certain that the system with the new feature is stable. Assessment of stability is done through the testing process documented below.

Build Automation

Build automation is achieved using Apache Ant.

The following are built automatically:

- Jar distributable
- Javadocs
- Combined dependencies into a single jar file

Appendices

- MHS Javadoc - http://cs2103aug12-t14-2j.googlecode.com/hg/doc/index.html

Credits

1. Google Calendar API(GData) and java client library: http://code.google.com/p/gdata-java-client/downloads/list
2. Date Time library : http://joda-time.sourceforge.net/
3. GSON: http://code.google.com/p/google-gson/