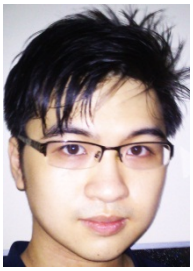
My Hot Secretary Project Manual

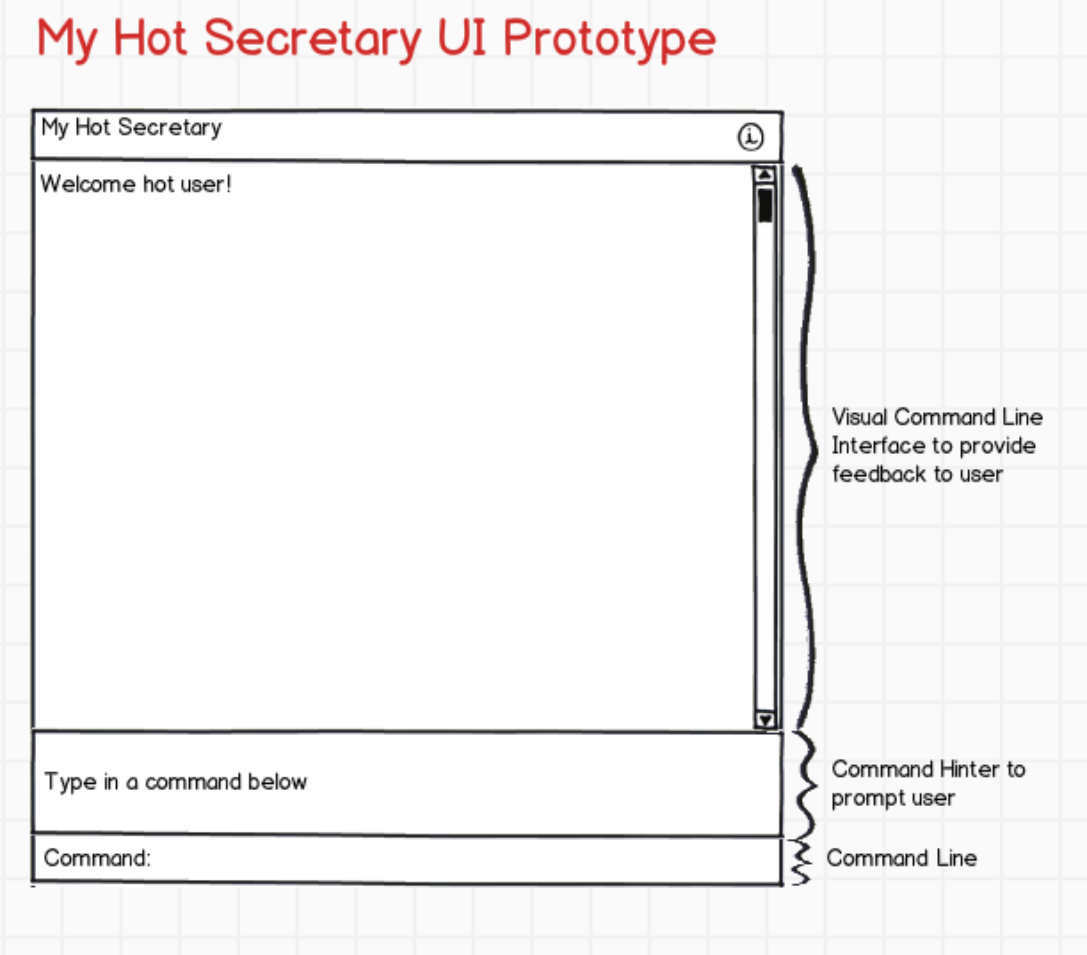
			
Timothy Lim Team leader Database	Shekhar Test master Processor	John Integration Ui	Kahou Cheong Documentation Command Parser

Contents

My Hot Secretary 4
How do I install it? 4
What does My Hot Secretary do? 4
Core Features 5
Quick Visual Guide 6
The Command Guide 7
Add 7
Supported Datetime Formats 7
Timed Task 8
Deadline Task 8
Floating Task 8
Display / Search 8
By Task 8
Update 8
Mark task as done <mark> <task name> 9
Delete - <delete> <task name> 9
Undo - <undo> 9
Redo 9
Help 9
Sync (Google Calendar) - [sync] 9
Program Architecture 11
User Interface 12
Class members: 14
Class methods: 14
Logic 14
Command Parser 15
Command Extractor 15
Date Extractor 16

Time Extractor 17
Name Extractor 17
Command 18
Processsor 18
Processor Logic 19
Storage 20
Database 21
Syncronize 21
TaskRecordFile 22
ConfigFile 22
GoogleCalendar 22
Database Policies 22
Testing 25
Appendices 26
Credits 26

User Guide



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. It is a directly executable file that runs on all major platforms.

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

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):

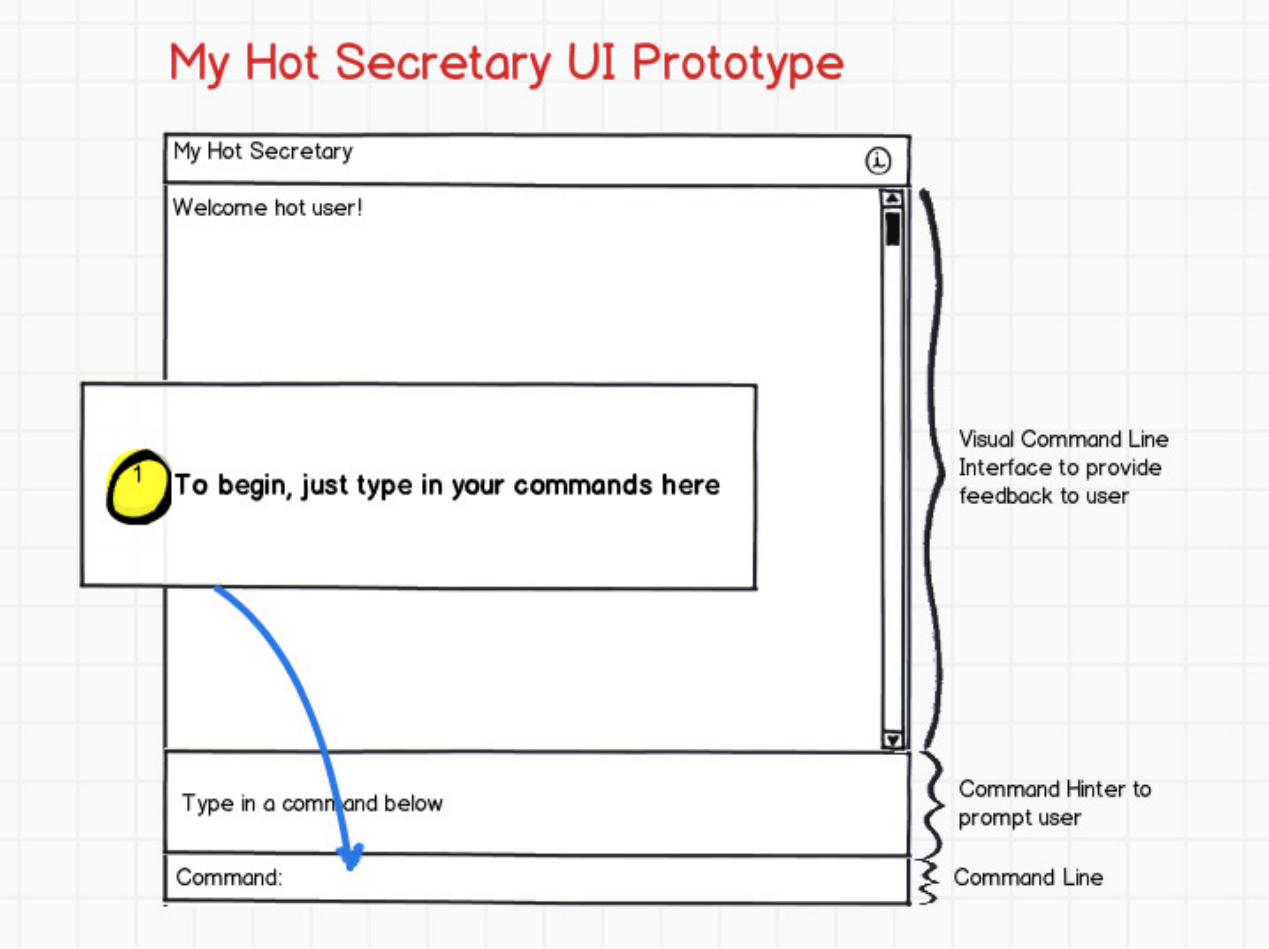
- Automated synchronization of tasks with Google Calendar.
- Automatic pushing of tasks after every operation.
- Automatic pulling of tasks from Google calendar every 5 minutes.
- Manual pulling of events from Google calendar.

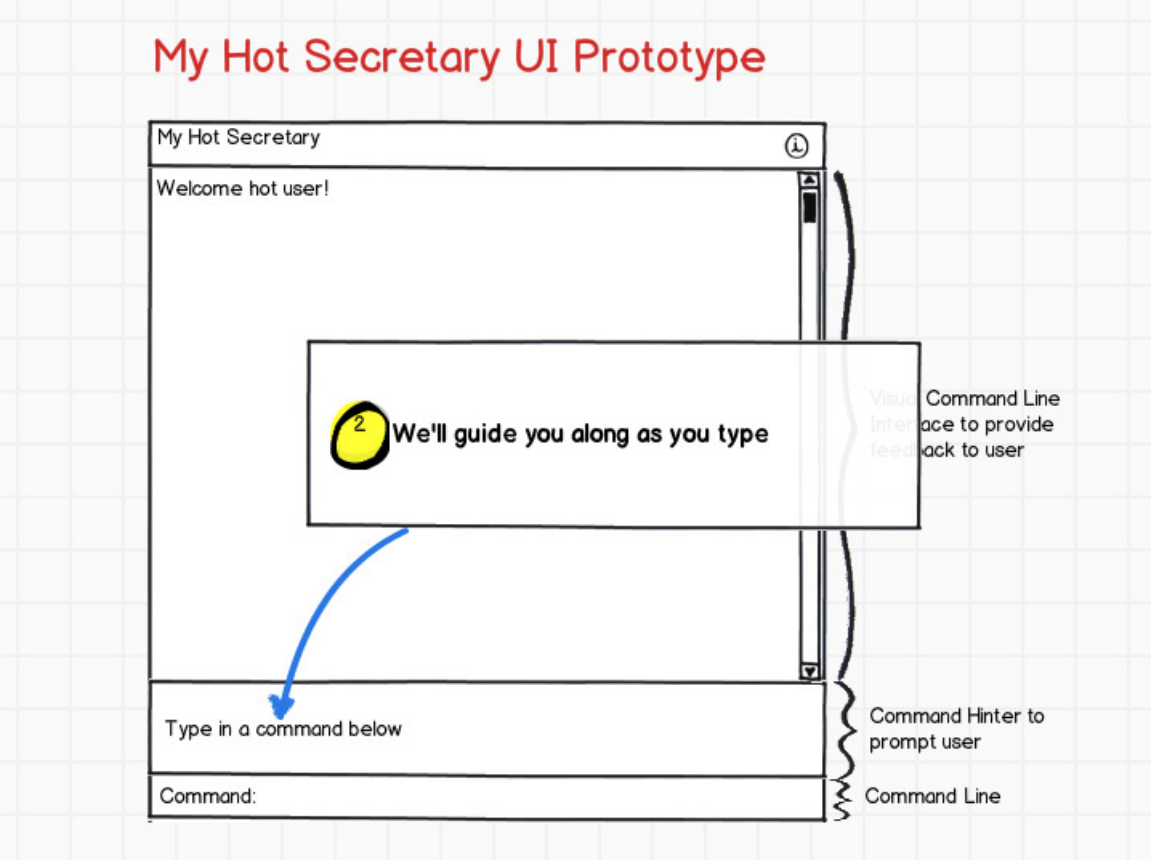
Flexi-commands

- Natural language recognition
- Recognize multiple command formats

Support for undo of previous actions

Quick Visual Guide





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

Display / Search

Display uses the 'show'/ 'search' commands to display all matching results.

By Day/Date - <show> <datetime>

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

By Task - ***<show> <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)

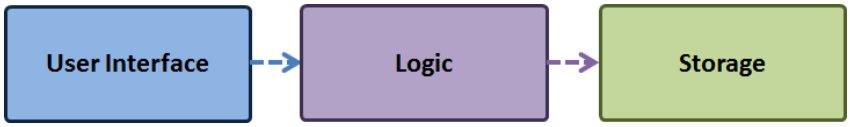
My Hot Secretary is a desktop application that helps a user keeps track of tasks.

Functionality summary:

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

Program Architecture

MHS utilizes an n-tier architecture:

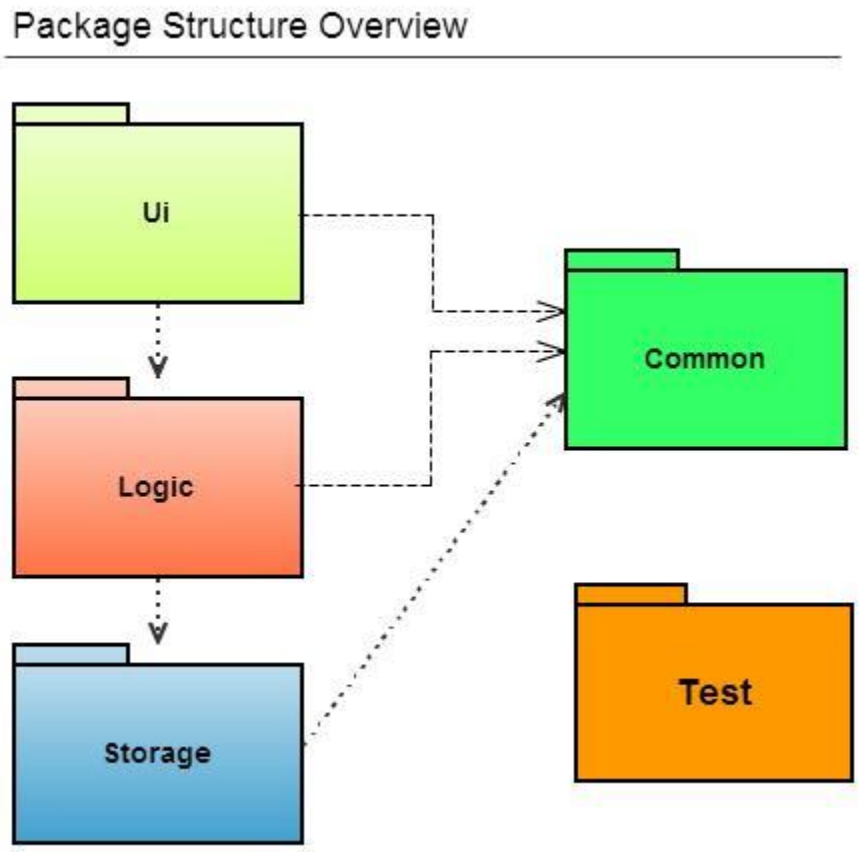


Main components are :

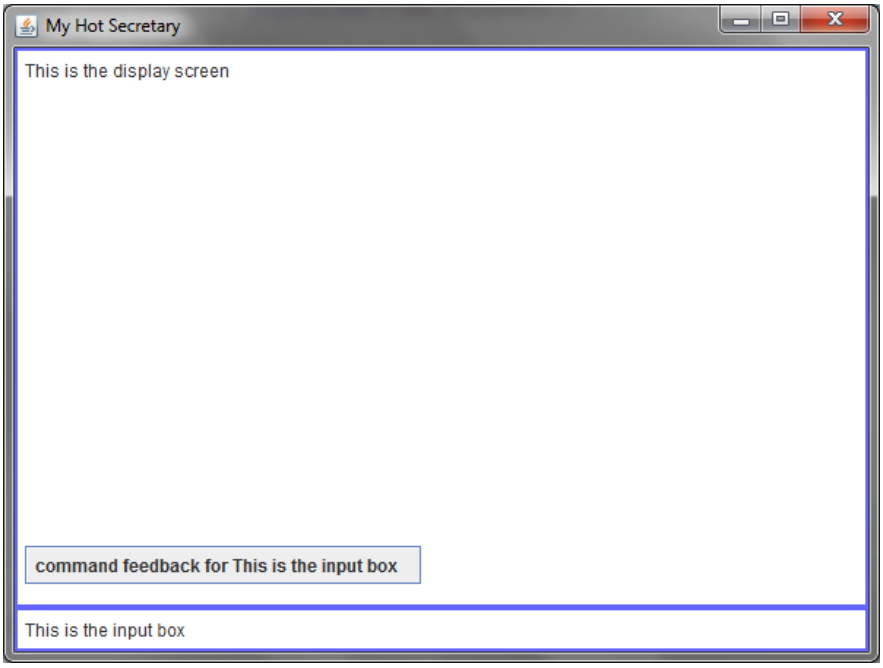
- **UI:** User interface consists of a JTextField to get user input, a JTextArea to display program output and a JPopup to provide feedback as the user types.

- **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.

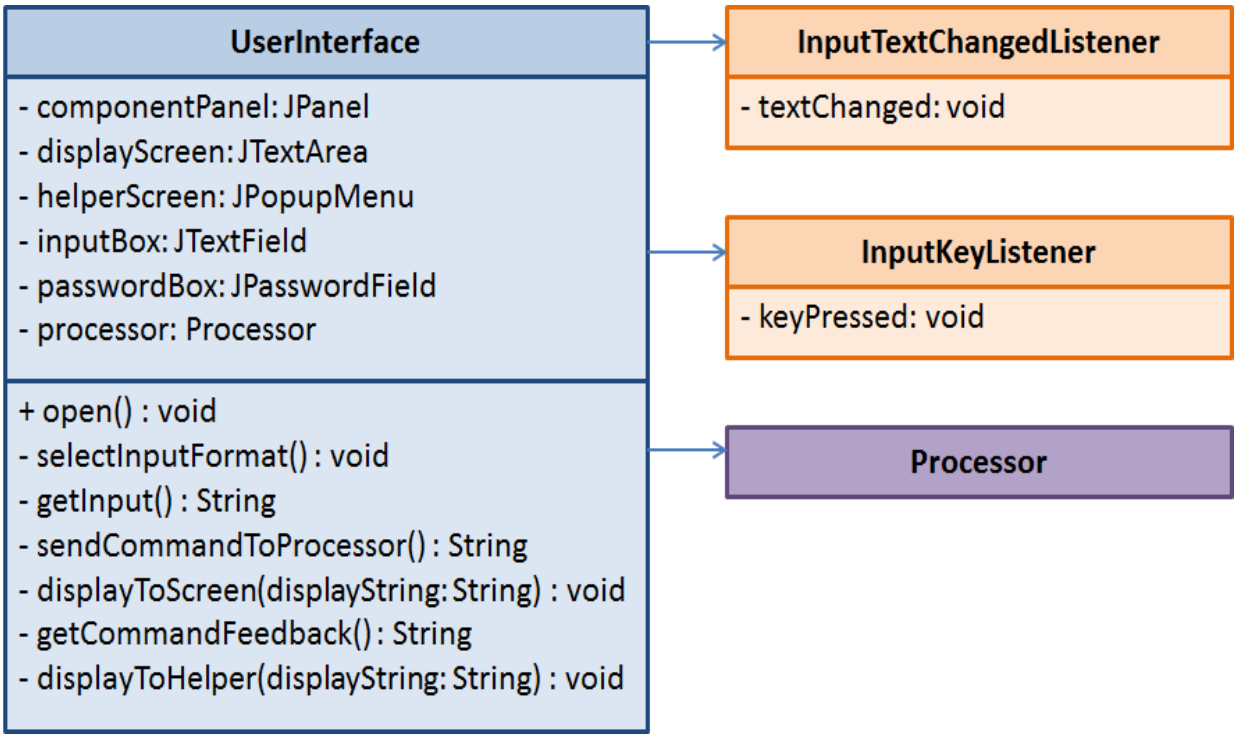
- **Storage:** Storage Tier interfaces persistent data storage mechanism, on local disk and remote (Google Calendar Service) and handles task queries and operations.



User Interface



The interface is implemented in the UserInterface class using javax.swing components:



Class members:

1. componentPanel: used to arrange all components of interface
2. displayScreen: used to display output to user
3. helperScreen: provide feedback to user as they type a command
4. inputBox: used to get plain text user input
5. passwordBox: used to get user password

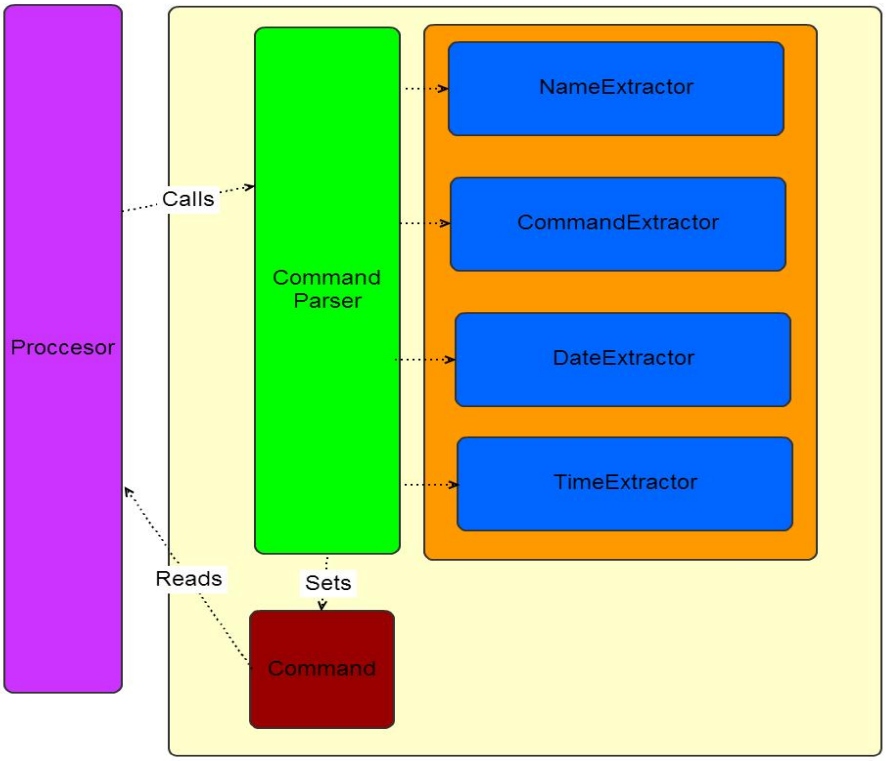
Class methods:

1. open: used to make the user interface visible to the user
2. displayToScreen: used to show output to the displayScreen.
3. selectInputFormat: used to switch between plain text input and password input, this selection is based on the processor parameter: isPasswordExpected().
4. getInput: gets the corresponding input from inputBox or passwordBox based on current input format.
5. sendCommandToProcessor: used to execute user command
6. getCommandFeedback: used to get feedback while a user is typing a command
7. UserInterface has an object of InputTextChangedListener so that getCommandFeedback is called after every keystroke
8. UserInterface has an object of InputKeyListener so that executeCommand is called when user hits the enter key

Logic

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

Command Parser



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.

Command

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.

Processsor

The processor class is the main logic of the class. It receives a command string from the UI, parses it using the Command Parser, processes it and then passes it to the database to perform the given operation on it. It contains an object of Command Parser and Database. The processor also provides real time command feedback to the user.

Processor Logic

1. UI calls the executeCommand() method with the command string.
2. Processor sends the command string to Command Parser which parses the command and returns an object of Command.

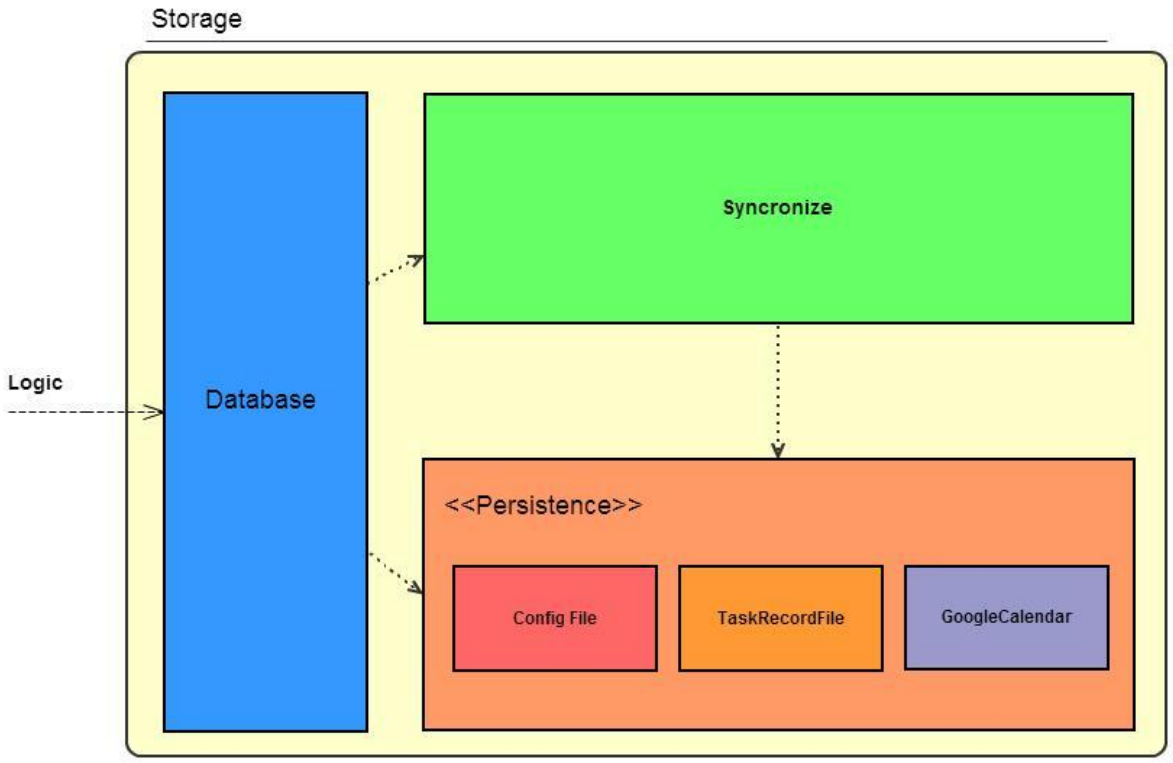
3 The processor class understands the kind of command and then manipulates tasks depending on it.

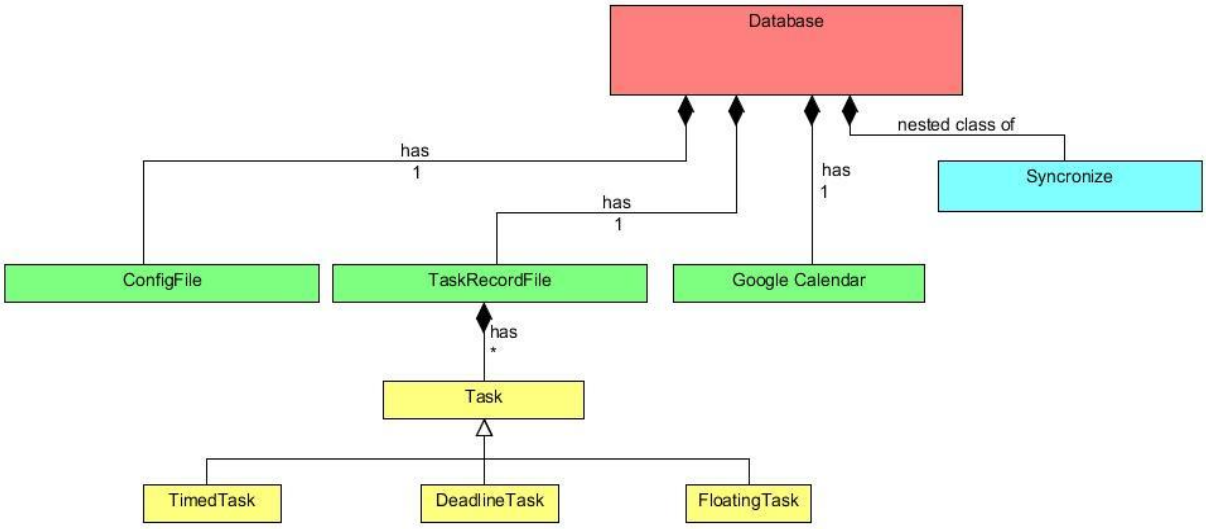
4 It then calls the Database class to perform CRUD.

5 The confirmation message is sent back to UI

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.





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 identifer (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.

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

GoogleCalendar interfaces Google Calendar CRUD operations and bridges task to CalendarEventEntry operations.

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

Exception thrown when task specified is invalid

TODO: create custom exception invalidTaskFormat for tasks that do 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 syncronization

Exception thrown when task does not exist

Exception thrown when task specified is invalid

TODO: create custom exception taskNotFound for operations on tasks that do not exist

Deleting Task

Deletes existing task and updates created time and updated time.

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

Exception thrown when task does not exist
Exception thrown when task specified is invalid

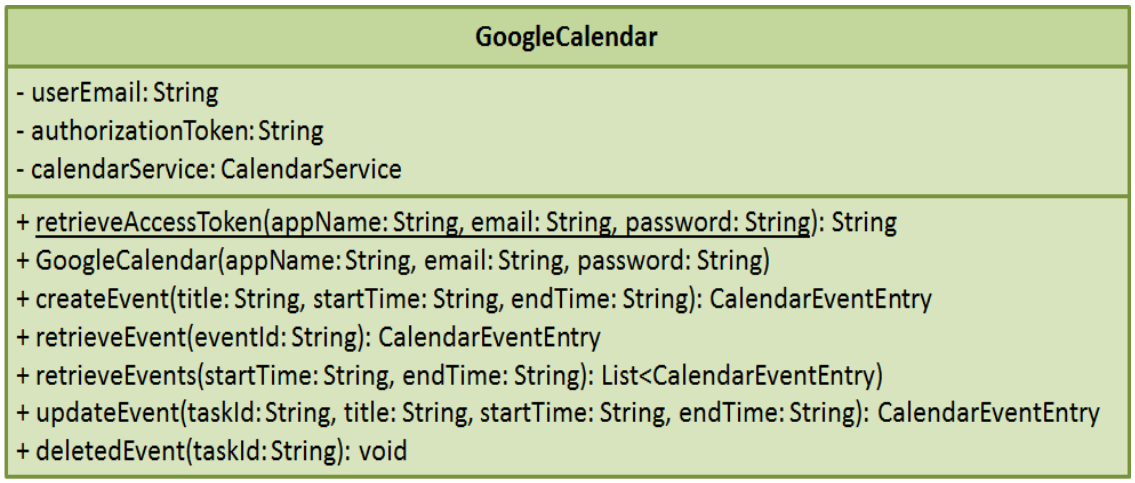
Querying Task(s)

Exception thrown when task specified is invalid

Google Calendar

This class utilizes the Google Calendar Data API and java code provided from:

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



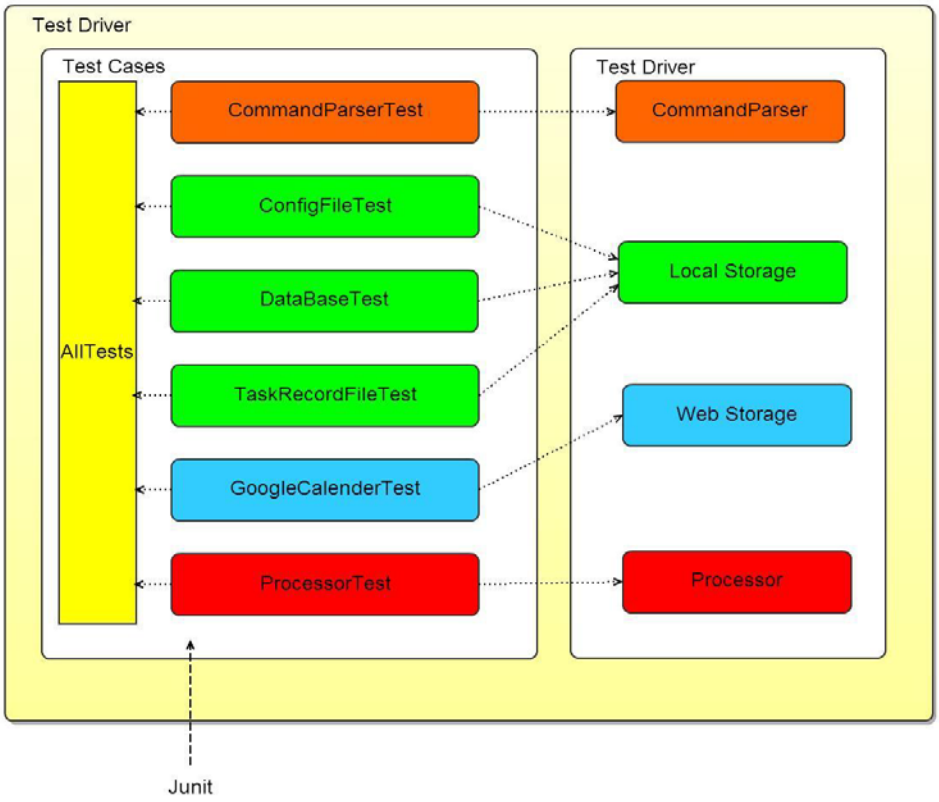
Class members:

1. userEmail: used to create the URL to access user's Google Calendar
2. authorizationToken: used to authorize access to user's Google Calendar
3. calendarService: from Google Calendar java library

Class methods:

1. retrieveAccessToken: used to get the authorization token for a user's Google Calendar
2. Using the retrieved access token, an instance of GoogleCalendar can be created
3. createEvent: adds an event to user's Google Calendar and returns the added entry
4. retrieveEvent: gets an event from user's Google Calendar
5. retrieveEvents: gets all events within the specified time range
6. updateEvent: updates an event in user's Google Calendar based on specified parameters, it then returns the updated entry
7. deleteEvent: deletes an event from users' Google Calendar

Testing



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

1. **Manual Testing**
2. **System Testing**
3. **Unit Testing**
    - Test Suite - All Tests
        - Individual jUnit tests

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

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/