[T14J3][V0.2]

Quick ToDo

			
Vaarnan Drolia	Poornima Muthukumar	Harsha Narayan	Ankit Ganla
The One(Leader), DB God, Mercurial Man	Design Guru, Ms. Social(Team Comm)	Stunt Girl(Testing), Grim Reaper(Deadlines)	Handyman(Features), Author(Documentation)

CREDITS

JIntellitype
Tulisky
Apache Commons Http Client
Oracle Java Documentation and API
Numerous tips from www.stackoverflow.com
Component Mover class by Rob Camick

GETTING STARTED

QuickToDo is a to-do manager targeted to power users, with the aim of making their workflow simple, fast and efficient. The features of this product are built for convenience. The entire functionality is accessible entirely through the Graphical User Interface (GUI). The backend of the product has been intentionally built to ease maintenance and continuous upgrade. Functionality can be added, removed and modified relatively easily. We have tried to ensure readable and high quality code with effective cohesion between the various components. In developing this product further, we ask that you uphold and/or enhance these standards and keep the user in mind when tailoring features.

Getting the code -

QuickToDo is an open source software and the entire source code can be found at this repository - http://code.google.com/p/quicktodo
We hope you enjoy developing this product further as much as we did when creating it.

Building code -

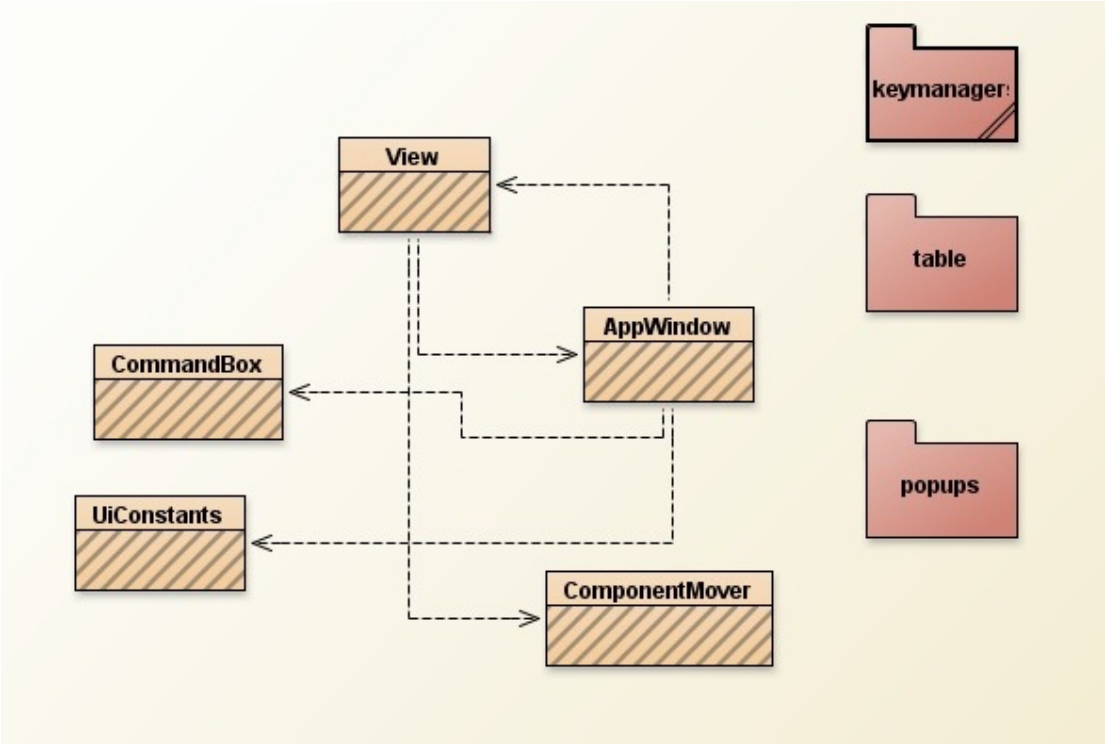
The code includes the ant build file build.xml and build.properties which can be used for packaging the final product. The default directory it builds to is “target”.

NAVIGATING THE CODE

The overall QuickToDo architecture is simple. The User Interface is removed from the backend and functions independently, dealing with the user. QuickToDo has a highly intuitive and convenient graphical user interface (GUI). The logic component is the core of the code which handles major functionality. The storage is again separate and deals only with maintaining the users data.

The following guide will help you get information on the section of the code you wish to focus on. We have tried to make it as complete as possible. Major functions and critical code sections are explained in greater detail. The entire list of API's, classes, functions and relevant java documents for the code can be found enclosed.
This guide describes each package in the code and classes associated with it. Each section has a description of the classes followed by a brief note (if required) on maintaining that section of code.

PACKAGE- USER INTERFACE



OVERVIEW

The UI for QuickToDo consists of a Command Line Interface (CLI) embedded withing a Grapgical User Interface (GUI). QuickToDo is run entirely through the keyboard. The GUI has features such as autocomplete, dynamic search display and boxes that appear only when required to show feedback or results. Shortcuts are provided for almost every function to aid the power user. The section below describes the UI classes and critical functions that run the UI. It has several sub- packages that hold relevant classes. For a full detail of the UI classes, check the source code

INDEPENDENT CLASSES

AppWindow

- Initializes the frame, command box, icon, table, scroll pane and all other UI components.
- Tracks input mapped keys.
- Decides the current view that the user sees (box, label, both etc).

CommandBox

- Receives all the input from the user.
- Manipulates it and sends it either to the app window or the view
- Also listens to key press and key release.

UiConstants - Holds a list of all the UI constants to make code more maintainable

ComponentMover - Allows you to move a component by using a mouse.

View

- Acts as a middle man between the AppWindow and the controller.
- It initializes the AppWindow
- Receives the command from the AppWindow and sends it to the controller.
- It thus links the UI to logic.

SUB PACKAGE - KEYMANAGERS

Classes – GlobalHotKeyListener, GlobalKeyboardGrabber, ProgramKeyManager

For details on these classes refer to the appendix

KeyManagers has classes that handle our global hotkey functions. We need to choose from one of 2 external libraries (Jintellitype or Tulisky) depending on the operating system and architechture (32 or 64 bit).

SUB PACKAGE - POPUPS

Classes – EventDetailPanel, HelpScreen , PasswordField, ResultBox

For details on these classes refer to the appendix

This class manages the popups that we show the user. Boxes that appear to display feedback and search results and then disappear dynamically are controlled from here

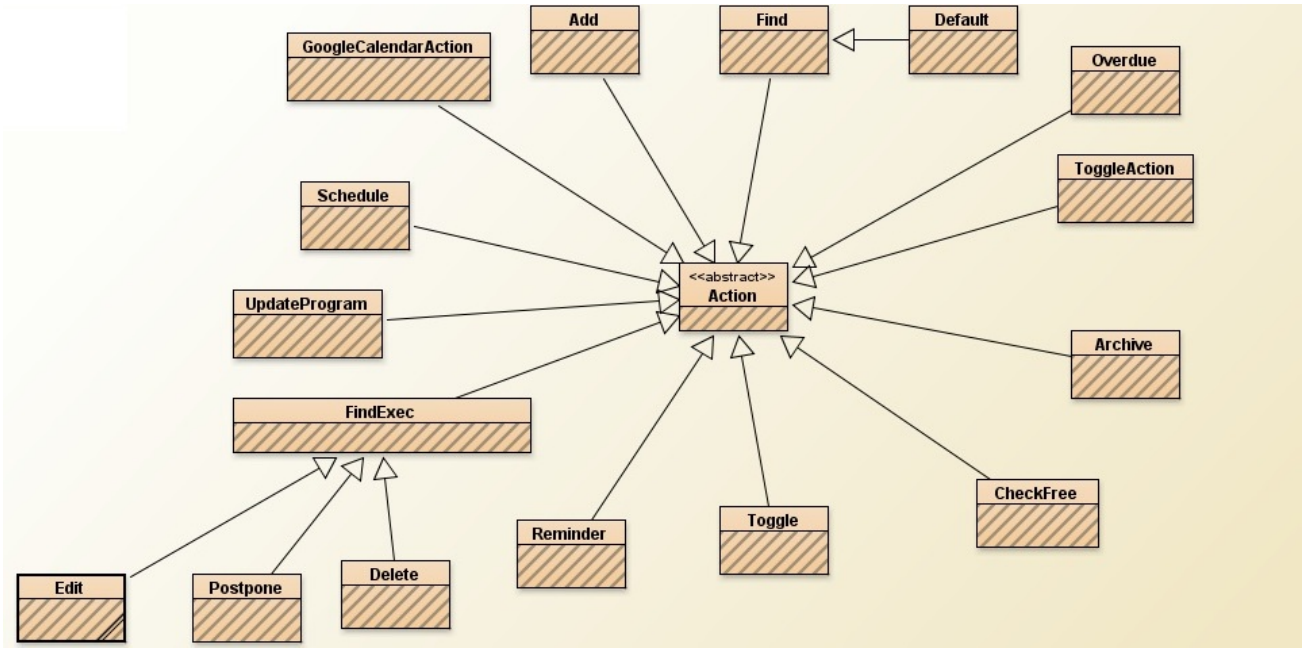
SUB PACKAGE - TABLE

Classes – EventButton, EventCellRenderer, EventTable, EventTableModel, TableButtonListener

For more details on these classes refer to the appendix

This package has classes which support the table view that displays events to the user.

PACKAGE - ACTION



OVERVIEW

The action package has an abstract class called Action. All other classes in this package directly or indirectly extend this class. Each action that the user performs is an object – for example, a user wanting to add a task will be creating an Add object. This object will then add the event. The same applies to all other actions. Which action to choose is decided by the keyword entered by the user. The default action if the user does not specify a keyword is passed to a default class. This executes find functionality. Find is highly effective as a default option as the program then provides functionality to perform most other functions with keyboard shortcuts.

Each action object also stores the result of the action it carried out. For example, the event which is added to the storage by an add object can be stored within that object itself. Hence each class in this package stores the event it affected. (e.g. delete stores the event it deleted).

UNDO FUNCTIONALITY

This makes it much easier to implement functionality like undo. We simply place the action object on an undo stack. If the user needs to undo, the last object is popped from the stack and can be manipulated to undo its action. For example, add can be undone by just removing the event it added. The event it added is, as mentioned before, stored in the add object.

To make sure only relevant objects are added to the undo stack, we use a boolean isUndoable() function, which keeps track of a boolean undoableCheck variable. If undoableCheck is true, only then is an object placed on the undo stack. This is because not all actions require an undo. For actions that may need undo, this value is set to true ONLY when the change is finally effected and the action has been completed. This is to prevent bogus entries in the stack.

The Action abstract class also has a function named execute(). This function takes in the command string entered by the user. It is overwritten in each subclass and dynamic binding ensures the correct instance of this function is executed depending on which action object is being run. Each object then parses the user input in a unique way to extract information required (see parser for more details).

FindExec Class

Another important class in this package is the FindExec class. This class extends from action, and a few other classes extend from FindExec. We found that there are several functionalities that require the use of the find function before executing another action. For example, to edit an event, it must first be found. But if an event is found, it may not necessarily need to be edited.
When the user selects the task that he actually wants to act on, the UI passes in the event's ID as a string. Only if an id is picked up does the FindExec allow the action to execute. Until then it functions as a find.

The execute functions of all classes in this package return an event array to the calling function.

OTHER CLASSES

Here are the classes. References to other classes here include DataManager (see DataManager) and Parser(see Parser)

Add - Adds an event to the database

Archive - Returns an array of events that have been ticked (marked done).

Schedule - checks for all free slots matching a particular duration in a particular day

CheckFree - Checks if a given time is free to schedule a task.

Default - Executes a search by creating a find object. Used if no keyword is entered

Delete - Removes an event from the storage using the DataManager remove() function

Edit - Modifies an event

Find- Searches for all events with a particular string and returns them all in an array

GoogleCalendarAction – Sync with Google Calendar (also see Google Manager and DataManager)

Overdue - execute() in this class returns an array of events that are Overdue.

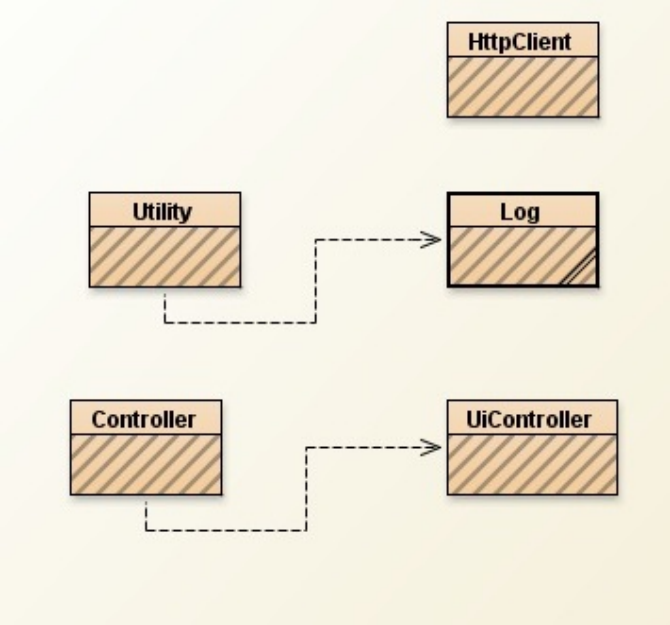
Reminder- execute() in this class returns a sorted array of all events in the storage.

Toggle – Chooses a toggle action and passes it to ToggleAction

ToggleAction - Actually implements the toggle (tick and star or vice versa)

UpdateProgram- Allows automatic update of QuickToDo

PACKAGE - CORE



OVERVIEW

The core package contains critical classes that access both User Interface as well as Storage classes. They also work with Logic classes to implement user functionality which is required by the user. The core is the heart of the code, where it all comes together

Controller

- Implements Runnable
- Actually runs the program
- Loads permanent storage into live storage using DataManager
- Maintains undo stack
- Chooses action objects to implement functionality
- Communicates with the UiController

HttpClient - Connects to the program website, checks for an update and downloads the latest file

UiController

- Controls what the user sees
- All result arrays from the action execute() functions are passed to this class
- It manipulates the result and passes it to the user via display
- Displays feedback to the user and error messages if needed

Log

- Uses an instance of the Logger class provided by java and the handlers available with it
- Initializes the logger which is used by all classes.
- Functions to log data-
    - DEBUG is an INFO level message
    - WARN is a WARNING level message
    - ERROR is a SEVERE level message

Utility

- This class just contains some functions used by many other classes
- They are grouped here for abstraction purposes

Developer Notes

UiController

Important functions to note
showResult()- This is a critical function that actually decides the display and hence what the user sees. Hiding and showing of display boxes, feedback boxes and such is driven from here. Please read through the code carefully to understand this function.

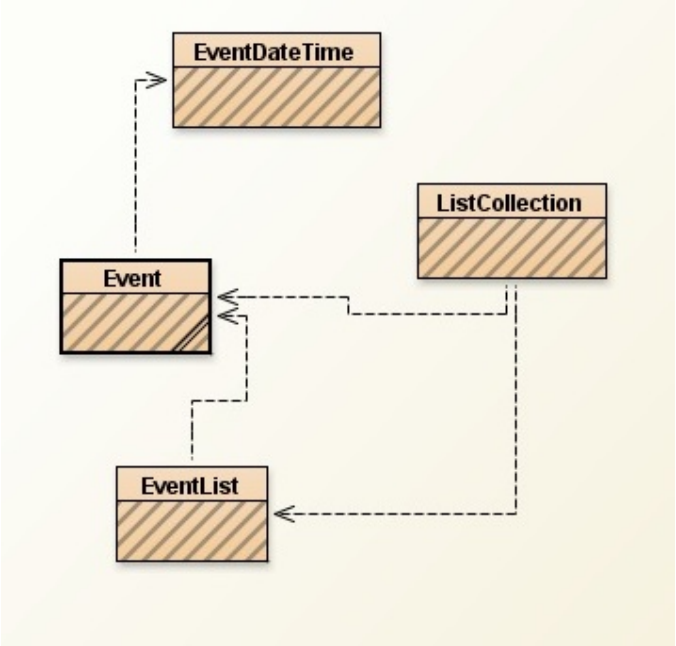
HttpClient

If changes need to be made to server links and/or update systems, this class will have to be modified

Log

Changes may be made to this class to create a different logging system

PACKAGE - DATA



OVERVIEW

The Data package consists of classes that define the data types in this code. These are elemental and used all over the code. An understanding of these data types is essential to the developer. Please read this section carefully and examine the package irrespective of which section you are working on.

Event

- The fundamental element of the program. Each task the entered is stored as an event object
- For a complete list of attributes of each event object, please examine the code.

EventDateTime

- Object that manipulates time, date and calendar objects in this code.
- The EventDateTime type is a consolidated data type that brings date and time together to define a start or end time of an event.

EventList

- Array implementation of lists that stores events.
- Events must belong to a list and can belong only to one list
- A list may have multiple events of the same name, because the two instances will have different Ids

ListCollection

- All lists are stored in a hash map
- This is the live storage. Access to the events for the logic ends at this point. Only DataManager (see DataManager) can link this to permanent storage

Developer Notes

See appendix for important functions that may be important in handling data.

PACKAGE - GOOGLE CALENDAR

This package only has the **GoogleCalendar** class has functions to log into Google Calendar. It can add, edit and delete events in the Google Calendar if the user is logged in.
Please read the Google Calendar API to understand the code more easily.

NOTE- Only events with user specified duration are pushed to Google Calendar. If the events have only a date, it will be an all day event in Google Calendar. Otherwise it will have a start and end time as usual.

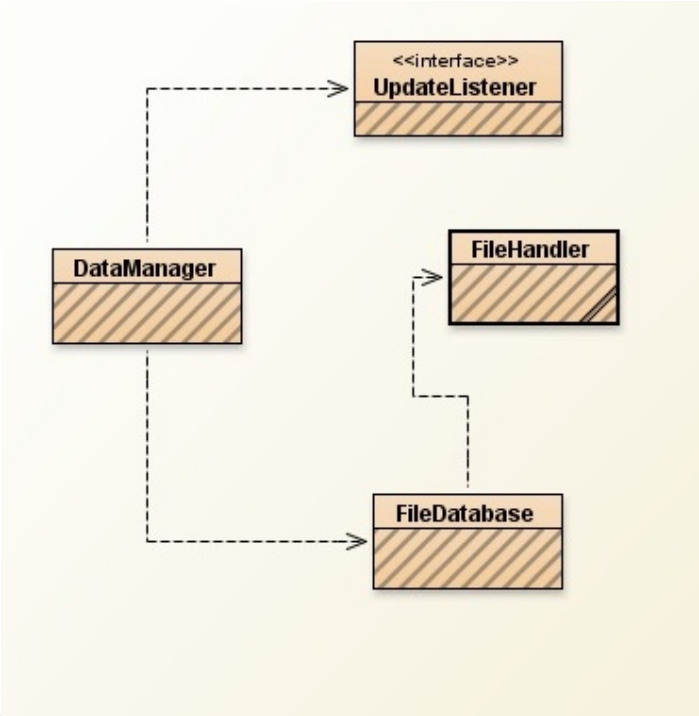
PACKAGE – PARSER

OVERVIEW

QuickToDo uses a parser that allows the user to enter commands in a variety of different ways, without restricting him/her to any particular format. We have tried, as far as possible, to allow commands to be as natural as possible. While we are proud of the current parser, we believe it can go a long way towards natural language processing and being even more intuitive than it already is. Unless you know exactly what you are doing, the regular expressions are best left untouched!

Developer Notes

The parser uses regular expressions to parse a string. We have functions that can create an event from a string, or we can extract only certain attributes such as start time, end time, duration etc. The parser is used extensively throughout the code.
Functions provided by the parser are listed in the java documentation. If you intend to work with the parser, please make sure you study this code well.



PACKAGE - STORAGE

OVERVIEW

The application has a live storage system, in the form of a hash map, as well as a permanent storage system with files. The two storage systems are continuously synced. The live storage allows quick searching and sorting while the permanent storage maintains a hard record.

Data is stored in the form of lists. Currently, the system provides a default list which contains all the events. This is an ArrayList implementation that has all our event objects. This is to facilitate multiple lists in the future, where the user can specify which list he needs to access.

DataManager
This class is the link between the Logic and the Storage system.

- Access to events in permanent and live storage must pass through this class
- Updates permanent storage based on changes in live storage every time a change is made
- loads up the love storage the first time the application is opened

FileDatabase

- Handles the permanent storage
- Links DataManager to the permanent storage with reading, writing and access functions

FileHandler

- Lowest level functions
- Stores data passed to it in xml files. Can convert these files to text files for ease of reading if required
- extracts files during automatic updating of application

Developer Notes

DataManager
Important functions to note-

savelists() - this function updates permanent storage. It overwrites all relevant data in permanent storage with new data.
loadlists()- this function loads all data in permanent storage into the live storage.
Provides **add(), remove() and replace()** which are required by action objects to modify storage.

FileHandler and FileDatabase are quite self - explanatory. Some notes are provided in the appendix.

Major Changes from V0.1

User Interface

1. Removed the tree structure from the UI
2. Merged into one view with single command box and dropdown tables with dialogs
3. Several shortcut keys added and made keyboard user friendly
4. Table view added and UI heavily revamped
5. User feedback, help screen added

Action

1. Added schedule, check free, star, tick, overdue, archive/done and reminder functionality
2. Added find and execute class
3. Added Google Calendar sync and automatic program update ability
4. Made undo more extensive
5. Global shortcut keys for Windows/Linux/Mac
6. Added more test cases
7. Improved significantly on the parser
8. Multiple event ticking/starring/deleting added

GoogleCalendar
STORAGE

Additional Programming Development

KNOWN ISSUES and BUGS TO KILL

If you are interested in making this product better by fixing bugs, here are some things you can work on

1. Multiple delete undo
2. Testing coverage can be made more extensive (never ending)

FUTURE GOALS

1. Multiple lists to be brought back
2. Shared lists
3. (Much) more testing
4. Log4J
5. Even better parsing

6. Support for repeating events
7. Multiple Star colours
8. Actual synchronization of all events with Google Calendar

CONCLUSION

We hope this developer guide helps you understand the source code better and make worthwhile contributions to making this product more user friendly and efficient. We only ask that you always keep the end user in mind while writing new features and try to uphold and enhance the quality of code. Have fun building and creating something you can be proud of!

APPENDIX

Additional Documentation for Action Package

Add -

- Adds an event to the database. Stores an array of length 1 with the event it has added
- Stores this array in each object as well

Archive

- Returns an array of events that have been ticked (marked done).

Schedule

- checks for all free slots matching a particular duration in a particular day
- Uses a minute by minute search. If the number of consecutive free minutes is equal to the duration needed, it stores that time period as a free slot.

CheckFree

- Checks if a given time is free to schedule a task.
- Stores an array of events clashing with that time
- If the array is empty, the slot is free. Otherwise clashing events are returned.

Default
- Executes a search by creating a find object
- Executed if user enters no keyword into the command box

Delete
- Removes an event from the storage using the DataManager remove() function

Edit
- Modifies an event
- Uses the DataManager replace function to swap the two events

Find
- Searches for all events with a particular string and returns them all in an array

GoogleCalendarAction
- Links the the GoogleCalendar class in the googlecal package (see GoogleCalendar)
- Uses DataManager functions setGoogleCalendar() and getGoogleCalendar().

Overdue
- execute() in this class returns an array of events that are Overdue.
- The overdue() function in this class finds these overdue events by comparing the current date and time to the date and time fields in the event object in question.

Reminder
- execute() in this class returns a sorted array of all events in the storage.
- Order of sorting is- overdue events, starred events (see Star in Toggle Class) and then all other events.

Toggle
- There are two functionalities that can be toggled- Star and Tick.
- This class decides which of these two toggle actions is required and then implements them by called the ToggleAction class.
- All objects of this type are undoable. The undoableCheck value is always true. The action is reversed by toggling tick or star in a similar manner as the first change.

ToggleAction
- Actually implements the toggle.
- Modifies the ticked or starred attribute of an event (see Event)
- Uses the DataManager replace() function to replace the previous event with the new toggled event.
- All objects of this type are undoable. The undoableCheck value is always true. The action is reversed by toggling tick or star in a similar manner as the first change.

UpdateProgram
- Connects to the server, checks for an update.
- If an update is available, it downloads the new update file, verifies it and then updates the current file. It then deletes the file it had downloaded.

Action Class Postpone

CLASS NOT SHIPPED IN CURRENT CODE
This class appears in the current source code, but implementation is still under development.
The objective is to allow the user to postpone a task just by specifying the changed time or duration, rather than going via edit.

UI PACKAGE - KEYMANAGERS

GlobalHotKeyListener

- onHotKey() tells the program what to do on hot key. This needs to be implemented correctly corresponding to the hot key used

GlobalKeyboardGrabber

- depending on the operating system and architechture (32 or 64 bit), class will choose either one of 2 external libraries (Jintellitype or Tulisky)

NOTE- Classes that enable the global hotkey are also included in the code. Please refer to java documentation on Jintellitype and Tulisky

ProgramKeyManager

- Subclass of KeyEventDispatcher which overrides some of the default keys to enable us to extend the functionality of some special keys.

UI PACKAGE – POPUPS

EventDetailPanel

- Displays a detailed view of the event
- If the user selects an event to see more details this box pops up

HelpScreen

- Displays the help screen HTML page when the user asks for help

PasswordField

- Displays a box for input of “username” and “password” for login access to Google Calendar

ResultBox

- This is a live feedback box that pops up each time an action is performed

UI PACKAGE – TABLE

EventButton

- Controls the toggle actions- helps to star and unstar events. Whenever an event is displayed, it decides whether to show a tick or a star or not

EventCellRenederer

- Implements table cell rendereder. Returns either a button or a label to be displayed in the table

EventTable

- Class which displays the table of events, allows the user to scroll through events, select evnts. Has a

default table model

EventTableModel
- Defines and sets the model for the event table

TableButtonListener
- Responds to mouse clicks on the star and tick buttons in the table

Important aspects of package DATA

Event
Important attribute - deadlineEvent. If the user does not specify a particular time or duration, but specifies by when a task should be done (deadline), this attribute is set to true.

Event has get() and set() functions for every attribute which are important to access and change the attributes of an object. In addition an equals() function is provided that can compare to event objects and is true if the objects are identical.

EventDateTime
This class uses the java calendar API. Calendar objects are of the Gregorian Calendar type. The constructor for this class can be default (if the user does not specify a time). More constructors are provided for a variety of inputs. All these are geared to creating start and end time attributes for an event.
Additionally get() and set() functions are provided to access and set Calendar attributes, date and time. An equals() function is provided to check if two EventDateTime objects are identical.
A compareTo() function is also provided which returns the difference in time between two EventDateTime objects.

EventList
Functionality to the user is provided primarily using the add(), remove() and replace() functions which modify the live storage. Usually, this class will be accessed by DataManager (see DataManager).
Additional get() and set() functions are provided to access and set names of lists, and all the events in the list, either individually or as a whole. A contains() function allows us to check if an event is present in a particular list.

ListCollection
Default list called “default” is created by the program
This has add(), remove() and replace() functions. These functions actually modify the events in live storage and the chain of events ends here. From here, permanent storage just needs to be updated.

Maintaining STORAGE

Datamanager

savelists() - this function updates permanent storage. It overwrites all relevant data in permanent storage with new data. It is called each time a change is made to any event or a list of events. Most functions that deal with adding, deleting or modifying data will end with this function. We update the permanent storage as soon as a change is made (each time live storage is changed). This function is critical.

loadlists()- this function loads all data in permanent storage into the live storage. It is called each time the application starts up and repopulates the hash map with the data in permanent storage to bring the application to life. This function is critical.

FileHandler

Important functions to note -

getSha1Checksum() - This function returns the SHA-1 encryption checksum for the supplied file which can be used to verify the integrity of the file. It returns a string which is the SHA-1 Checksum.

unzipFile() - unzips the file specified by the string to to the path specified by the second parameter.