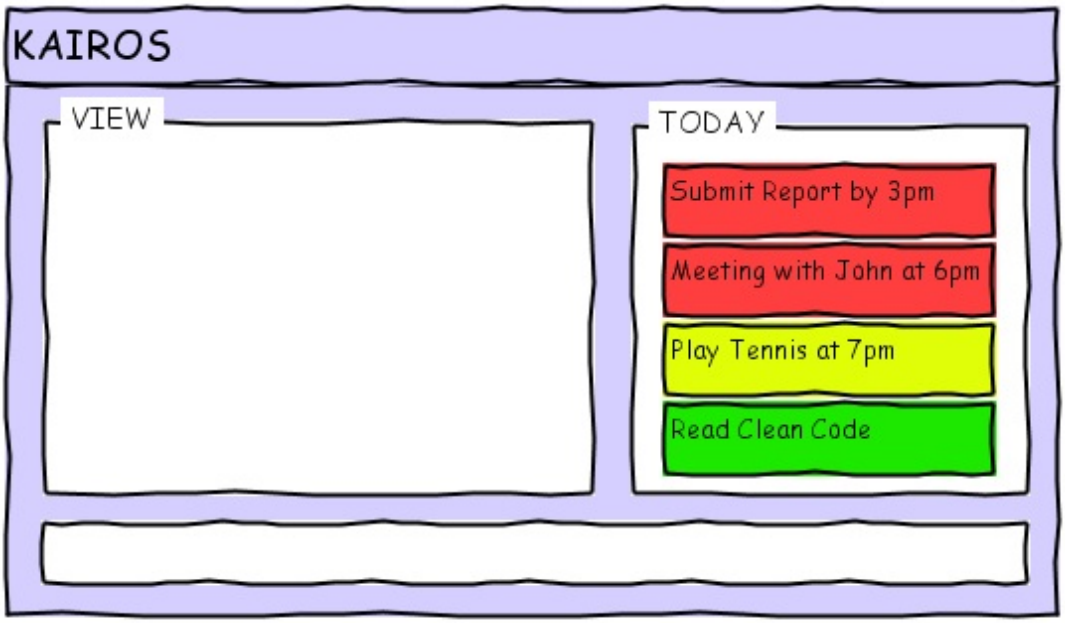
[F12-J3][V0.1]

Kairos

USER GUIDE



 Nguyen Hoang Thanh Duc, David	 Hari Krishna Vetharenian	 Karthik Rajasekaran	 Bathini Yatish
Team Leader	Repo Manager	Architect	Tester, Debugger

Hello World !

Kairos is a open-source tool that helps you keep track and stay ahead of all of life's happenings all in one interactive app that is both easy-to-use and packed with features. It helps you meet deadlines, finish work and lets you achieve greater productivity. Kairos is designed to put user experience at the forefront during task handling through the ease of Text Command Line Interface. So let Kairos help you make every second count.
Let's get started.

Autostart - Boot up your system and Kairos is there waiting for you to start the tasks for the day.
Shortcut – Hit **Ctrl+Alt+T** at any point to open the task menu at any point in time to view the Kairos console.

FEATURES

1. SimpleAdd:
Just type in what you would normally write down and voila! Kairos understands and updates your task list.kairos have an exhaustive add command which is capable of handling all events possible with a few short keywords. It's as intuitive as it can get.
Intelligent - Auto complete helps you save typing a few more keystrokes. The text also is colored based on the priority of the task, highlighting to you various parts of the input.

Task with time happening on the same day
_add meeting with John at 5pm

Task with time happening on tomorrow
_add submit report by 10am tomorrow

Task with start and end time
_add play tennis from 2pm to 6pm this Saturday

Recurrent task
_add visit Anny every sun and tues at 6pm

Task on specific date with time
_add attend ABC conference on 29 Oct 2012 10am

Task with priority
_add attend CS2103 lecture on 14 Sep 2012 2pm -h (high priority)
_add attend ECE sharing session 10 Sep 2012 (medium priority date/time specified)
_add re-organize room (floating tasks low priority)

Task with start and end date, and location
_add camping at Nation Park from 10 Oct 2012 10am to 11 Oct 2012 5pm
_add camping at Nation Park from 10/10 10am to 11/10 5pm
_add camping at Nation Park from 10/10 1000 to 11/10 1700

Floating task
_add reading The Mythical Man Month

2. SimpleEdit:
Need to change some details of a task? The edit command has got you covered. It's able to quickly find the task you need to modify and enable you to change it's description.

Use Case:
Scenario - User wants to modify a meeting with John, postpone it from 5 to 6pm
Actor - Jim
Pre-Condition : User is on the CLI of Kairos
1. User enters the following command:
_edit John.
This shows a list of all his tasks involving "John".
2. User adds another keyword "meeting" to refine his search:
_edit John meeting
This lists down all his meetings scheduled with John. Since there is only one meeting, he can select the task by pressing **TAB**.
3. The task details appear on the command line and the user can edit it now.
4. User can now postpone his meeting with John by changing the time of the meeting to 6pm in the task details. The user presses **ENTER** to confirm this change.

Extensions:
2. (a) There are no tasks with John and meeting. Displays <none found> and waits for user to change command.
3. (a) Incorrect values: If user enters incorrect time, inform user about time error and go to 3.

3. SimpleView
Zip through your tasks with view. It can find tasks based on any parameter, listing them in an easy to read format. It even allows you to create lists for different priorities, places and people, making it incredibly friendly.

All tasks with John. Works similarly for place, date
_view John

All high priority tasks of the day.
_view -h

All the done tasks of the day
_view done

Also each of the view statements can be appended by an **all** keyword which will display all the tasks matching with it in the entire database of tasks to be done/already done.
eg: **_view -h all**
_view done all

4.Undo
Made a mistake? No need to fret. With undo, you can remove the changes you may have previously edited or added.
_undo
or use the hotkey as follows
Ctrl + Z

5.SimpleComplete
When you are all done with the tasks for the day, just complete them. Strike them off the task list or if you'd like a back up, archive them to a log file.
Complete the task that is present at the index for the day's task
_done [index]

Complete with all the tasks for the day
_done all

6. SimpleRemainders
You might in the rush of things forget about certain events and tasks. Which is why Kairos has SimpleRemainders. It allows to set notifications to pop up in the System Tray and in the task menu. By default alerts are sent just before the task starts. Kairos also have other alert features like an alert 30mins before the task.
Alerts are not sent for low priority tasks as there isn't a to-do by time for them.

7. Help!?
Stuck? Forgot syntax? Worry not because Kairos has an inbuilt help feature.
_help
This will enlist all the user commands available

_help [command_name]
This will display all options for the [command_name] command

8. GCal Integration:
Combine the power of Google Services with Kairos.
_sync your_name_here@gmail.com *****

Sync your tasks with your Google Calendar and take it anywhere you go.

Thank you for choosing Kairos to be your daily task manager!
Make every second count.

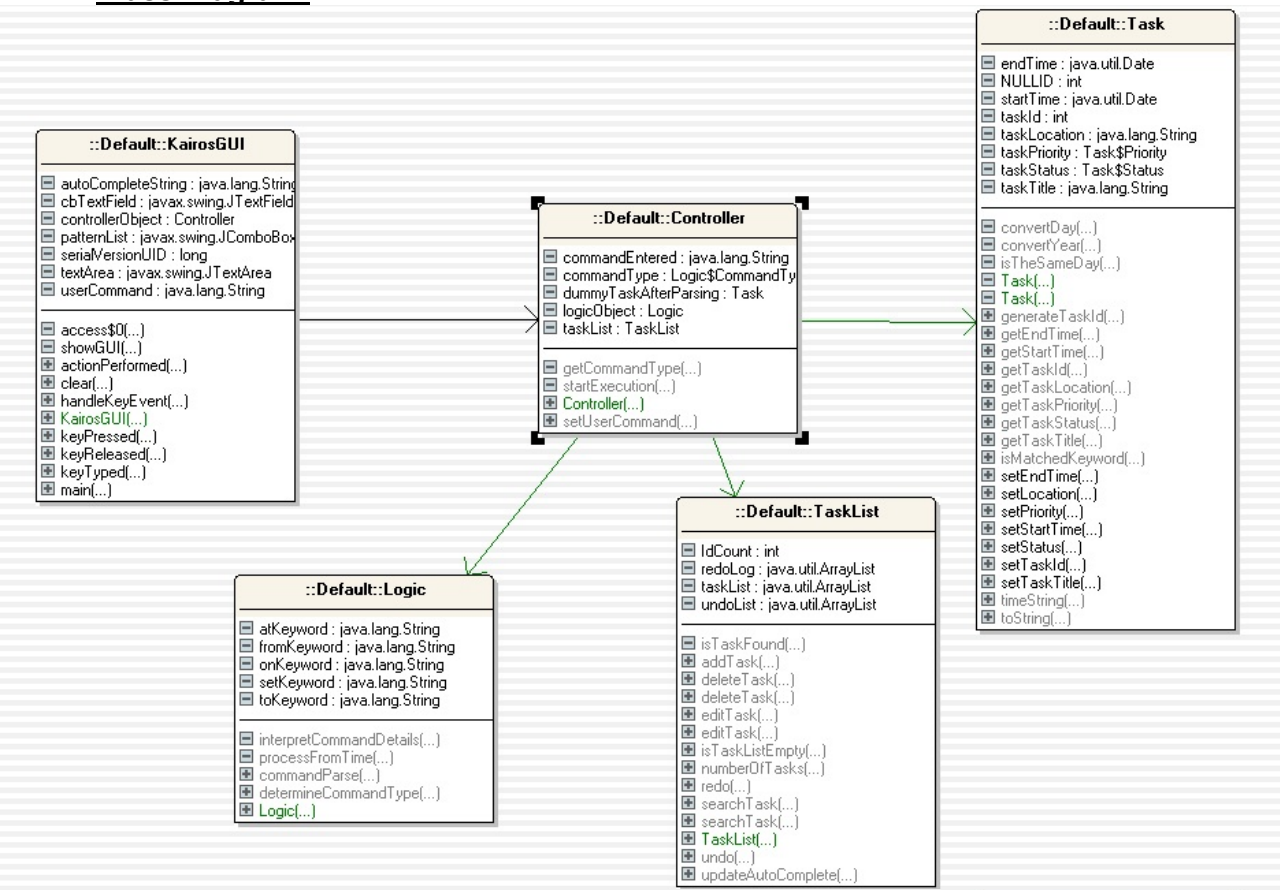
[F12-J3][V0.1]

Kairos

DEVELOPER GUIDE

1. **Design and implementation:**
   There are 7 main class which are:
   - Controller: plays a role as a container which will create objects of other classes and passing parameter between functions and objects.
   - UI: interacts with User such as taking User command and display message to User.
   - Logic: interprets the command from User. It also analyzes and provides necessary information to Kairos.
   - TaskList: acts as a buffer which contains a list of Task. It also has 2 Log stack for Undo and Redo feature.
   - Task: is the fundamental object of Kairos. It contains all the information of a task which User is trying to manage.
   - Autocomplete: (to be implemented) is inherited from TaskList. This will work concurrently with UI to provide a user-friendly feature.
   - Storage: (to be implemented) is the main database which will store all the task even when the application is closed or terminated.

2. **Class Diagram:**



3. **Important APIs:**
    a. *Controller:*

boolean setUserCommand(String userCommand)	Function that interacts with the GUI and calls the corresponding functions to perform the operations relevant to the commands entered by the user
String[] getTodayTask()	Returns an array containing the tasks pertaining to that day to be displayed on a separate frame by the GUI

    b. *GUI:*

void invokeGUI()	Sets up the GUI frame and starts it.

    c. *Logic:*

Task parseCommand(String command)	splits up the command into respective fields then combines them into one Task object which it then returns to the Controller
CommandType determineCommandType(String command)	returns the CommandType of the command entered.
Date processFromTime(String fromTime){	returns the parsed date to be added to the Task generated by the parseCommand function

    d. *Task*

bool setTaskId(int id);	assign ID of a task
bool setTaskTitle(String title);	assign Title of a task
bool setTaskStartTime(Date startTime);	assign start time of a task
bool setTaskEndTime(Date endTime);	assign end time of a task
bool setTaskLocation(String location);	assign location details of a task
bool setTaskStatus(int status);	update status of a task

bool setTaskPriority(int priority);	assign priority of a task
int getTaskId();	return ID of a task
String getTaskTitle();	return the title of a task
Date getTaskStartTime();	return the start time of a task
Date getTaskEndTime();	return the end time of a task
String getTaskLocation();	return the location of a task
int getTaskStatus();	return the status of a task
int getTaskPriority();	return the priority of a task
int generateTaskId();	generate a unique id based on the date and time of creation of the task (**to be implemented**)
String toString();	return a string with all data of a task collated together
boolean isMatchedKeyword(String keyword)	check if a task contains the input keyword

e. *TaskList:*

bool addTask(Task task);	add a task into the data structure
bool deleteTask(Task task);	delete a task from the data structure
bool deleteTask(int taskId); (**overloading**)	delete a task with the input ID from the data structure
bool editTask(int taskId, Task updatedTask);	edit a task with the input ID by the updatedTask from the data structure
Task searchTask(task Id);	search a task by ID (return a unique task)
List<Task> searchTask(String keyword); (**overloading**)	search tasks by their info, return a list of results
int totalNumberOfTasks();	return total number of tasks
bool isTaskListEmpty();	return TRUE if empty, else return FALSE
bool updateAutoComplete();	**to be implemented in future**

f. *Storage: (to be implemeted)*

void writeToFile(ArrayList<Task>)	writes the entire stored list of tasks onto an xml file
ArrayList<Task> readFromFile()	returns an arraylist of task objects that are stored in the file

[paths]
default = https://nuskarthik:****@code.google.com/p/cs2103aug12-f12-3j/