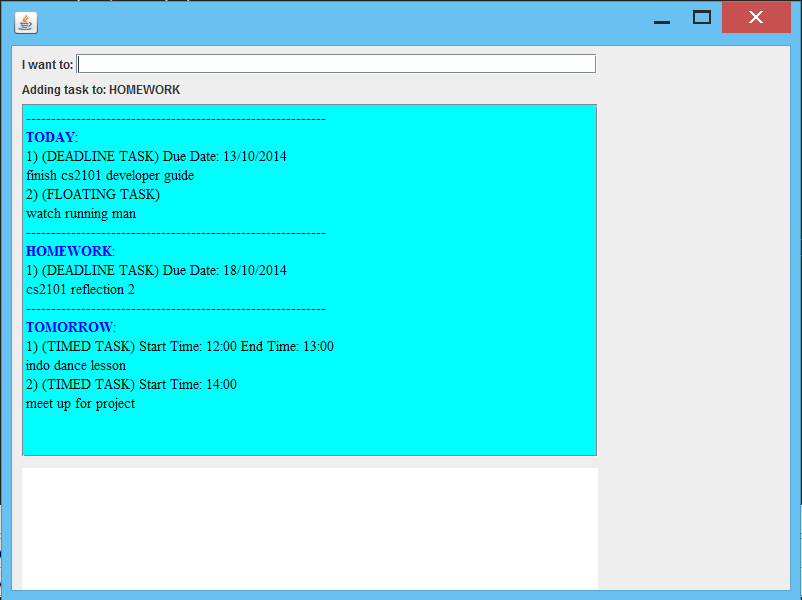
**To-Do-This (Great Day, Any Day)**

Supervisor: Hanrui  
Extra Feature: Flexi Command pro

|  |  |  |  |
| --- | --- | --- | --- |
| Jermyn Tanu  Team leader  Code quality | Justin Law  Documentation Team Manager | Victor Tan  Testing Integration | Choong Hui Min  Deliverables and deadlines  Scheduling and tracking |

## User guide

|  |  |  |
| --- | --- | --- |
| **Command Name** | **Format** | **Description** |
| Add | <Event Name> <Time> <Colour> <Label> <!> | Add an event to the specific  label. Command is flexible. |
| Delete | Delete <Label><TaskNum> | Delete an event under a  specific label. |
| DeleteLabel | Delete<Label> | Delete a specific label. |
| Display | Display <Label> | Display all the tasks under a  specific label. |
| Edit | Edit <Label> <TaskNum> <Details> | Edit an already added task  under a specific label. |
| Hide | Hide <Label> | Hide all tasks under a specific  label. |
| Search | Search <Keywords> | Search all task containing the  keywords. |
| Sort | Sort <Label> | Sort all task under a specific label by date. |
| Undo | Undo | Undo user previous command. |
| Help | Help | List all available commands |
| Exit | Exit | Exit the program. |
| Label | Label <Label> | To create a new label. |
| Done | Done <Label><TaskNum> | To mark a specific task as done |

**User guide (changes):**

ADD

[taskDetails] [Date] [Time] [Priority]

Normal task / Timed task depends on whether the time and date field are entered.

EDIT

[Edit] [LabelName] [taskID] [taskDetails] [Date] [Time] [Priority]

Can edit tasks with greater details.

## Developer Guide

The system is able to add and manage tasks simply and efficiently. Commands are passed from the user to ToDoThis.java which diverts it to TDTParser.java. The following components of the command are separated to be accessed again by the Logic within TDTParser.java. Command class is created and all the components are stored within. In the logic component, the Command class and DateAndTime class helps to create the Task class which will be stored in Storage.

The new developer has to understand the system flow from the user to the logic and then back to the user.

**Architecture Diagram**

Logic

ToDoThis

Storage

Parser

**OOP Component Design Diagram**

**To-Do-This**

**Logic**

TDTLogicc

TDTGUI

Task

**Parser**

**Storage**

TDTParser

TDTStorage

Command

TDTDateAndTime

**Sequence diagram For To-Do-This**

TDT Storage

TDT Logic

TDT Parser

TDT GUI

User

Show updated list of tasks

Show message “Task Deleted”

removeTask(Task)

executeCommand(command)

Parse(DELETE)

Delete task

TDT Storage

TDT Logic

TDT Parser

TDT GUI

User

Show message “Task added”

Show updated list of tasks

addTask(Task)

executeCommand(command)

Parse(ADD)

Add new task

**Design and Implementation**

**Use Case**

Software System: To-Do-This

User Case: To-Do-This – Add Command

Actors: Users of To-Do-This

MSS:

1. User runs To-Do-This.
2. To-Do-This loads current database (To-Do-This.txt) of previously stored tasks.
3. To-Do-This displays all the labels together with the stored tasks.
4. User chooses to add task under existing TOMORROW.
5. User changes current label to TOMORROW.
6. User enters new task under label TOMORROW.
7. To-Do-This displays updated label TOMORROW with newly added task.
8. To-Do-This displays message “Add success”.
9. User exits.

Use Case ends

Extensions:

2a. To-Do-This is unable to locate existing database.

2a1. To-Do-This creates a new database file (To-Do-This.txt) for storage.

4a. To-Do-This does not contain the label TOMORROW.

4a1. To-Do-This creates a new label TOMORROW.

5a. User incurs a typing error while trying to change current label to TOMORROW.

5a1. To-Do-This creates a new label with the incorrectly spelled TOMORROW.

5a2. User wishes to undo previous command.

6a. User inputs an invalid task format

6a1. To-Do-This categorises task under FLOATING TASK by default.

Software System: To-Do-This

User Case: To-Do-This – Delete Command

Actors: Users of To-Do-This

MSS:

1. User runs To-Do-This.
2. To-Do-This loads current database (To-Do-This.txt) of previously stored tasks.
3. To-Do-This displays all the labels together with the stored tasks.
4. User chooses to delete task under existing TOMORROW.
5. User deletes task under label TOMORROW.
6. To-Do-This displays updated label TOMORROW after deleting task.
7. To-Do-This displays message “Task deleted”.
8. User exits.

Use Case ends

Extensions:

2a. To-Do-This is unable to locate existing database.

2a1. To-Do-This creates a new database file (To-Do-This.txt) for storage.

4a. To-Do-This does not contain the label TOMORROW.

4a1. To-Do-This displays message “Error, label does not exist”.

5a. User incurs a spelling error while typing the label TOMORROW.

5a1. To-Do-This displays message “Error, label does not exist”.

5b. User inputs an invalid task ID for deletion.

5b1. To-Do-This displays message “Error, invalid task number”.

**Code Examples**

**Graphical User Interface**

public static void main**(**String**[]** args**)** **{**

EventQueue**.**invokeLater**(new** Runnable**()** **{**

public void run**()** **{**

**try** **{**

TDTGUI frame **=** **new** TDTGUI**();**

frame**.**feedbackArea**.**setText**(**frame**.**doInit**());**

frame**.**setVisible**(true);**

frame**.**taskPane**.**setText**(**frame**.**displayTask**());**

frame**.**taskLabel**.**setText**(**"Adding task to: "

**+** frame**.**storage**.**getCurrLabel**());**

**}** **catch** **(**Exception e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**});**

**}**

The main method invokes the GUI which starts up the java applet.

private String doInit**()** **{**

storage **=** **new** TDTStorage**(**FILENAME**);**

parser **=** **new** TDTParser**();**

logic **=** **new** TDTLogic**(**storage**);**

**try** **{**

storage**.**readInitialise**();**

**}** **catch** **(**Exception e**)** **{**

**return** "Unable to create todothis.txt"**;**

**}**

**return** "Todo-This ready!"**;**

**}**

doInit() creates a storage text file which records the task database.

**Parser**

public Command parse**(**String userCommand**)** **{**

COMMANDTYPE commandType **=** COMMANDTYPE**.**INVALID**;**

String labelName **=** ""**;**

boolean isHighPriority **=** **false;**

boolean completeDetails **=** **false;**

boolean isValidEdit **=** **false;**

String commandDetails **=** ""**;**

int taskID **=** **-**1**;**

TDTDateAndTime dateAndTime **=** **new** TDTDateAndTime**();**

String dateAndTimeParts **=** ""**;**

ArrayList**<**String**>** prepositionWords **=** **new** ArrayList**<**String**>();**

prepositionWords**.**add**(**"on"**);**

prepositionWords**.**add**(**"at"**);**

prepositionWords**.**add**(**"by"**);**

prepositionWords**.**add**(**"from"**);**

prepositionWords**.**add**(**"about"**);**

prepositionWords**.**add**(**"to"**);**

prepositionWords**.**add**(**"-"**);**

prepositionWords**.**add**(**"until"**);**

prepositionWords**.**add**(**"till"**);**

prepositionWords**.**add**(**"next"**);**

prepositionWords**.**add**(**"following"**);**

commandType **=** determineCommandType**(**getFirstWord**(**userCommand**));**

String remainingWords **=** removeFirstWord**(**userCommand**);**

**switch(**commandType**)** **{**

**case** ADD **:**

remainingWords **=** userCommand**;**

isHighPriority **=** isPriority**(**remainingWords**);**

**if** **(**isHighPriority**)** **{**

remainingWords **=** remainingWords**.**replace**(**"!"**,** ""**);**

**}**

parts **=** remainingWords**.**split**(**" "**);**

**for** **(**int i **=** 0**;** i **<** parts**.**length**;** i**++)** **{**

String checkWord **=** parts**[**i**];**

**if** **(**TDTDateAndTime**.**checkDate**(**checkWord**))** **{**

completeDetails **=** **true;**

**}** **else** **if** **(**TDTDateAndTime**.**checkTime**(**checkWord**))** **{**

completeDetails **=** **true;**

**}** **else** **if** **(**TDTDateAndTime**.**checkDay**(**checkWord**)!=**0**)** **{**

completeDetails **=** **true;**

**}** **else** **if** **(**TDTDateAndTime**.**checkMonth**(**checkWord**)!=**0**)** **{**

**if** **((**i**>**0**)** **&&** parts**[**i**-**1**].**contains**(**"\\d+"**))** **{**

**if** **(** **(**i**+**1 **<** parts**.**length**)** **&&**

**(**parts**[**i**+**1**].**contains**(**"\\d+"**)))** **{**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,** parts**[**i**-**1**]);**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,** parts**[**i**]);**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,** parts**[**i**+**1**]);**

commandDetails **=** removeDetails**(**commandDetails**,** i**);**

checkWord **=** ""**;**

i**++;**

**}**

**}**

**}**

**if** **(**completeDetails**)** **{**

**if** **(**i**>**0**)** **{**

**if** **(**prepositionWords**.**contains**(**parts**[**i**-**1**]))** **{**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,**

parts**[**i**-**1**]);**

commandDetails **=** removeDetails**(**commandDetails**,** i**);**

**}**

**}**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,**

checkWord**);**

completeDetails **=** **false;**

**}** **else** **{**

commandDetails **=** addDetails**(**commandDetails**,** checkWord**);**

**}**

**}**

dateAndTime **=** **new** TDTDateAndTime**(**dateAndTimeParts**);**

**break;**

**case** DELETE **:**

parts **=** remainingWords**.**split**(**" "**);**

**if** **(!**isValidPartsLength**())** **{**

**break;**

**}**

// delete [label] / delete [taskID]

**if** **(**parts**.**length **==** 1**)** **{**

**if** **(**parts**[**0**].**matches**(**"\\d+"**))** **{**

taskID **=** getTaskID**(**0**);**

**}** **else** **{**

labelName **=** getLabelName**(**0**);**

**}**

**}**

// delete [label][taskID] or delete [taskID][label] (assumes label to be one word)

**if** **(**parts**.**length **==** 2**)** **{**

**if** **(**parts**[**1**].**matches**(**"\\d+"**))** **{**

taskID **=** getTaskID**(**1**);**

labelName **=** getLabelName**(**0**);**

**}** **else** **{**

taskID **=** getTaskID**(**0**);**

labelName **=** getLabelName**(**1**);**

**}**

**}**

**break;**

**case** EDIT **:**

parts **=** remainingWords**.**split**(**" "**);**

**if** **(!**isValidPartsLength**())** **{**

**break;**

**}**

// [taskID][commandDetails]

**if** **(**parts**[**0**].**matches**(**"\\d+"**))** **{**

taskID **=** getTaskID**(**0**);**

**if** **(**parts**.**length **>** 1**)** **{**

remainingWords **=** remainingWords**.**substring**(**parts**[**0**].**length**()).**trim**();**

isValidEdit **=** **true;**

**}**

// [labelname][taskID][commandDetails]

**}** **else** **if** **(**parts**.**length **>** 1**)** **{**

**if** **(**parts**[**1**].**matches**(**"\\d+"**))** **{**

taskID **=** getTaskID**(**1**);**

labelName **=** getLabelName**(**0**);**

remainingWords **=**

remainingWords**.**substring**(**parts**[**0**].**length**()).**trim**();**

remainingWords **=**

remainingWords**.**substring**(**parts**[**1**].**length**()).**trim**();**

isValidEdit **=** **true;**

**}**

**}**

**if** **(**isValidEdit**)** **{**

// same as ADD

remainingWords **=** remainingWords**.**trim**();**

isHighPriority **=** isPriority**(**remainingWords**);**

**if** **(**isHighPriority**)** **{**

remainingWords**.**replace**(**"!"**,** ""**);**

**}**

parts **=** remainingWords**.**split**(**" "**);**

**for** **(**int i **=** 0**;** i **<** parts**.**length**;** i**++)** **{**

String checkWord **=** parts**[**i**];**

**if** **(**TDTDateAndTime**.**checkDate**(**checkWord**))** **{**

completeDetails **=** **true;**

**}** **else** **if** **(**TDTDateAndTime**.**checkTime**(**checkWord**))** **{**

completeDetails **=** **true;**

**}** **else** **if** **(**TDTDateAndTime**.**checkDay**(**checkWord**)!=**0**)** **{**

completeDetails **=** **true;**

**}** **else** **if** **(**TDTDateAndTime**.**checkMonth**(**checkWord**)!=**0**)** **{**

**if** **((**i**>**0**)** **&&** parts**[**i**-**1**].**contains**(**"\\d+"**))** **{**

**if** **(** **(**i**+**1 **<** parts**.**length**)** **&**  **(**parts**[**i**+**1**].**contains**(**"\\d+"**)))**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,** parts**[**i**-**1**]);**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,** parts**[**i**]);**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,** parts**[**i**+**1**]);**

commandDetails **=** removeDetails**(**commandDetails**,** i**);**

checkWord **=** ""**;**

i**++;**

**}**

**}**

**}**

**if** **(**completeDetails**)** **{**

**if** **(**i**>**0**)** **{**

**if** **(**prepositionWords**.**contains**(**parts**[**i**-**1**]))** **{**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,**

parts**[**i**-**1**]);**

commandDetails **=** removeDetails**(**commandDetails**,**

i**);**

**}**

**}**

dateAndTimeParts **=** addDetails**(**dateAndTimeParts**,**

checkWord**);**

completeDetails **=** **false;**

**}** **else** **{**

commandDetails **=** addDetails**(**commandDetails**,**

checkWord**);**

**}**

**}**

dateAndTime **=** **new** TDTDateAndTime**(**dateAndTimeParts**);**

**}**

**break;**

**case** LABEL **:**

commandType **=** COMMANDTYPE**.**LABEL**;**

labelName **=** remainingWords**;**

**break;**

**case** SORT **:**

commandType **=** COMMANDTYPE**.**SORT**;**

**break;**

**case** UNDO **:**

commandType **=** COMMANDTYPE**.**UNDO**;**

**break;**

**case** SEARCH **:**

commandType **=** COMMANDTYPE**.**SEARCH**;**

commandDetails **=** remainingWords**;**

**break;**

**case** DISPLAY **:**

String checkDisplay**[]** **=** remainingWords**.**split**(**" "**);**

commandType **=** COMMANDTYPE**.**DISPLAY**;**

labelName **=** checkDisplay**[**0**];**

**break;**

**case** HIDE **:**

String checkHide**[]** **=** remainingWords**.**split**(**" "**);**

commandType **=** COMMANDTYPE**.**HIDE**;**

labelName **=** checkHide**[**0**];**

**break;**

**case** DONE **:**

commandType **=** COMMANDTYPE**.**DONE**;**

parts **=** remainingWords**.**split**(**" "**);**

**if** **(!**isValidPartsLength**())** **{**

**break;**

**}**

**if** **(**parts**.**length **==** 1**)** **{**

**if** **(**parts**[**0**].**matches**(**"\\d+"**))** **{**

taskID **=** getTaskID**(**0**);**

**}** **else** **{**

labelName **=** getLabelName**(**0**);**

**}**

**}**

**if** **(**parts**.**length **==** 2**)** **{**

**if** **(**parts**[**1**].**matches**(**"\\d+"**))** **{**

taskID **=** getTaskID**(**1**);**

labelName **=** getLabelName**(**0**);**

**}** **else** **{**

taskID **=** getTaskID**(**0**);**

labelName **=** getLabelName**(**1**);**

**}**

**}**

**break;**

**case** INVALID **:**

commandType **=** COMMANDTYPE**.**INVALID**;**

**break;**

**default:**

//Will not reach here

**break;**

**}**

**return** **new** Command**(**commandType**,** labelName**,** taskID**,** commandDetails**,** dateAndTime**,** isHighPriority**);**

**}**

Parser reads in user input and creates Command object containing task details.

**Logic**

public String executeCommand**(**Command command**)** **{**

String feedback **=** ""**;**

storage**.**getLabelPointerStack**().**push**(**storage**.**getCurrLabel**());**

storage**.**getUndoStack**().**push**(**storage**.**copyLabelMap**());**

**switch(**command**.**getCommandType**())** **{**

**case** ADD **:**

**return** doADD**(**command**);**

**case** DELETE **:**

feedback **=** doDelete**(**command**);**

storage**.**write**();**

**return** feedback**;**

**case** EDIT **:**

**return** doEdit**(**command**);**

**case** LABEL **:**

**return** doLabel**(**command**);**

**case** SEARCH **:**

**return** ""**;**

**case** HIDE **:**

**return** doHide**(**command**);**

**case** UNDO **:**

storage**.**getLabelPointerStack**().**pop**();**

storage**.**getUndoStack**().**pop**();**

feedback **=** doUndo**(**command**);**

storage**.**write**();**

**return** feedback**;**

**case** DISPLAY **:**

**return** doDisplay**(**command**);**

**case** DONE **:**

feedback **=** doDone**(**command**);**

storage**.**write**();**

**return** feedback**;**

**default:**

**return** ""**;**

**}**

**}**

**Storage**

public void readInitialise**()** **throws** IOException **{**

**try** **{**

BufferedReader br **=** **new** BufferedReader**(new** FileReader**(**fileName**));**

int totalLabel **=** Integer**.**parseInt**(**br**.**readLine**());**

**for(**int i **=** 0**;** i **<** totalLabel**;** i **++)** **{**

String label **=** br**.**readLine**();**

labelMap**.**put**(**label**,** **new** ArrayList**<**Task**>());**

**}**

**while(**br**.**ready**())** **{**

String line **=** br**.**readLine**();**

String**[]** params **=** line**.**split**(**"\t"**);**

TDTDateAndTime date **=** **new** TDTDateAndTime**(**params**[**4**],** params**[**5**],**

params**[**6**],** params**[**7**]** **);**

Task task **=** **new** Task**(**0**,** params**[**0**],** params**[**1**]** **,** date**,false,**

**false,** **false);**

**if(!**labelMap**.**containsKey**(**params**[**0**]))** **{**

labelMap**.**put**(**params**[**0**],** **new** ArrayList**<**Task**>());**

**}**

task**.**setTaskID**(this.**getLabelSize**(**params**[**0**])** **+** 1**);**

**if(**params**[**2**].**equals**(**"true"**))** **{**

task**.**setHighPriority**(true);**

**}**

**if(**params**[**3**].**equals**(**"true"**))** **{**

task**.**setDone**(true);**

**}**

**this.**addTask**(**task**);**

**}**

br**.**close**();**

**}** **catch(**Exception e**)** **{**

bw **=** **new** BufferedWriter**(new** FileWriter**(**fileName**));**

**}**

**}**

public void write**(){**

**try** **{**

bw **=** **new** BufferedWriter**(new** FileWriter**(**fileName**));**

bw**.**write**(this.**getLabelMap**().**size**()+** ""**);**

bw**.**newLine**();**

Set**<**String**>** labels **=** **this.**getLabelMap**().**keySet**();**

Iterator**<**String**>** labelIter **=** labels**.**iterator**();**

**while(**labelIter**.**hasNext**())** **{**

bw**.**write**(**labelIter**.**next**());**

bw**.**newLine**();**

**}**

Iterator**<**Task**>** iter **=** **this.**getTaskIterator**();**

**while(**iter**.**hasNext**())** **{**

Task task **=** iter**.**next**();**

bw**.**write**(**task**.**getLabelName**()** **+** "\t" **+** task**.**getDetails**()** **+** "\t"

**+**

task**.**isHighPriority**()** **+** "\t" **+** task**.**isDone**()** **+** "\t" **+**

task**.**getDateAndTime**().**getStartDate**()** **+** "\t" **+**

task**.**getDateAndTime**().**getEndDate**()** **+** "\t" **+**

task**.**getDateAndTime**().**getStartTime**()** **+** "\t" **+**

task**.**getDateAndTime**().**getEndTime**());**

bw**.**newLine**();**

**}**

bw**.**close**();**

**}** **catch** **(**IOException e**)** **{**

System**.**out**.**println**(**"Error. Unable to initialise write file."**);**

**}**

**}**

**INSTRUCTIONS FOR TESTING**

Run TDTParserTest.java or TDTStorageTest.java to facilitate with the testing of the methods.

**API FOR METHODS (IMPORTANT)**

**Class Command**

The Command class is used to filter the details that the user keys in and separate them into the different components.

Field Summary

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private int | taskID  Integer value assigned to the task for labelling. |
| private String | labelName  Category that the task is added under. |
| private String | commandDetails  Contains task details that user wants to input. |
| private TDTDateAndTime | dateAndTime  Contains date and time of task. |
| private boolean | isHighPriority  True is task is of high priority and false if otherwise. |

Constructor Summary

|  |  |
| --- | --- |
| Modifier | Constructor and Description |
| Public | Command(COMMANDTYPE commandType, String labelName, **int** taskID,  String commandDetails, TDTDateAndTime dateAndTime,  **boolean** isHighPriority)  Creates a Command object for the specified user input. |

Method Summary

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| COMMANDTYPE | getCommandType()  Return CommandType such as add or delete and so on. |
| void | setCommandType(COMMANDTYPE commandType)  this.commandType = commandType |
| String | getLabelName()  return labelName |
| void | setLabelName(String labelName)  this.labelName = labelName |
| String | getCommandDetails()  return commandDetails |
| void | setCommandDetails(String commandDetails)  this.commandDetails = commandDetails |
| boolean | isHighPriority()  return isHighPriority |
| void | setHighPriority(boolean isHighPriority)  this.isHighPriority = isHighPriority |
| int | getTaskID()  return taskID |
| void | setTaskID(int taskID)  this.taskID = taskID |
| TDTDateAndTime | getDateAndTime()  return dateAndTime |
| void | setDateAndTime(TDTDateAndTime dateAndTime)  this.dateAndTime = dateAndTime |

**Class Task**

public class Task implements Comparable<Task>

The Task class contains the vital information regarding the task that the user wants to add.

Field Summary

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private int | taskID  Integer value assigned to the task for labelling. |
| private String | labelName  Category that the task is added under. |
| private TDTDateAndTime | dateAndTime  Contains date and time of task. |
| private String | details  Contains task details that user wants to add. |
| private boolean | isHighPriority  True is task is of high priority and false if otherwise. |
| private boolean | hide  False if user wants to display otherwise true. |
| private boolean | isDone  True if task has been completed and false if otherwise. |

Constructor Summary

|  |  |
| --- | --- |
| Modifier | Constructor and Description |
| public | Task(**int** taskID,  String labelName,  String details,  TDTDateAndTime dateAndTime, **boolean** p)  Creates a Task object for the specified user input. |
| public | Task(**int** taskID,  String labelName,  String details,  TDTDateAndTime dateAndTime, **boolean** p,  **boolean** done,  **boolean** hide)  Creates a Task object for the specified user input. |

Method Summary

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| int | getTaskID()  return taskID |
| void | setTaskID(int taskID)  this.taskID = taskID |
| String | getLabelName()  return labelName |
| void | setLabelName(String labelName)  this.labelName = labelName |
| String | getDetails()  return details |
| void | setDetails(String details)  this.details = details |
| boolean | isHighPriority()  return isHighPriority |
| void | setHighPriority(boolean isHighPriority)  this.isHighPriority = isHighPriority |
| boolean | isHide()  return hide |
| void | setHide(boolean hide)  this.hide = hide |
| boolean | isDone()  return isDone |
| void | setDone(boolean isDone)  this.isDone = isDone |
| TDTDateAndTime | getDateAndTime()  return dateAndTime |
| void | setDateAndTime(TDTDateAndTime dateAndTime)  this.dateAndTime = dateAndTime |

**Class TDTDateAndTime**

The TDTDateAndTime class stores converted date and time in the format DD/MM/YYYY, XX:XX 24hours format.

Field Summary

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private String | startDate  starting date of task. |
| private String | endDate  ending date of task. |
| private String | startTime  starting time of task. |
| private String | endTime  starting time of task. |
| private String | details  date and time combined. |

Constructor Summary

|  |  |
| --- | --- |
| Modifier | Constructor and Description |
| public | TDTDateAndTime(String dateAndTime\_details)  Creates a TDTDateAndTime object for the specified user date and time. |
| public | TDTDateAndTime(String startDate, String endDate, String startTime,  String endTime)  Creates a TDTDateAndTime object for the specified user date and time. |
| public | TDTDateAndTime()  Creates a TDTDateAndTime object for the specified user date and time. |

Method Summary

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| void | decodeDetails(String details)  Takes in user input date and time and converts it to the desired format. |
| String | getStartDate()  return startDate |
| String | getEndDate()  return endDate |
| String | getStartTime()  return startTime |
| String | getEndTime()  return endTime |
| String | getDetails()  return details |
| void | displayDateTime(boolean deadline)  Displays date and time. |
| void | display()  Displays date and time and type of task. |
| static boolean | isValidTimeRange(String time)  To validate if the time input is valid. Returns true if valid otherwise false. |
| static boolean | isValidTimeCompare(String startTime, String endTime)  To validate if startTime is earlier than endTime such that task is valid. Returns true if valid otherwise false. |
| static boolean | isValidDateRange(String date)  To validate if the date input is valid. Returns true if valid otherwise false. |
| static boolean | isValidDateCompare(String startDate, String endDate)  To validate if the startDate is earlier or equal to the endDate such that task is valid. Returns true if valid otherwise false. |
| static int | getNumOfDaysFromMonth(int month, int year)  Return number of days for a given month in a given year. |
| static boolean | checkTime(String nextWord)  Checks for all possible formats of time input. Examples include 2am, 11pm, 2359, 2.00, 2.00am, 12.15pm.  Return true if String contains a time format otherwise returns false. |
| static boolean | checkDate(String nextWord)  Checks for all possible formats of date input. Examples include 2-2-1992, 05/05/1995, 6.1.1944, 03041992. Return true if String contains a date format otherwise returns false. |
| static int | checkDay(String day)  return int value corresponding to a certain day of the week. Examples include 2 for Monday, 3 for Tuesday and so on. |

**Class TDTParser**

public class TDTParser implements ITDTParser

The TDTParser class takes in user input and returns a Command object. ITDTParser is an interface which contains enum COMMANDTYPE.

Method Summary

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| Command | parse(String userCommand)  returns a Command object with user input separated into the different components. |
| private static COMMANDTYPE | determineCommandType(String commandTypeString)  returns a COMMANDTYPE object. Default is add. |

**Class TDTLogic**

public class TDTLogic implements ITDTLogic

The TDTLogic class takes in a Command object and executes the command accordingly. ITDTLogic is an interface which contains all possible commands that TDTLogic is able to execute.

Field Summary

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private TDTStorage | storage  Stores local memory, input and output details. |

Constructor Summary

|  |  |
| --- | --- |
| Modifier | Constructor and Description |
| public | TDTLogic(TDTStorage storage)  Creates a TDTStorage object and retrieves latest storage. |

Method Summary

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| String | executeCommand(Command command)  Return a String object containing feedback to inform user of command execution progress. |
| String | doADD(Command command)  Return a String object to inform user that task has been added successfully.  Default method if command type is not specified. |
| String | doDelete(Command command)  Method is able to delete all entries and labels (delete), delete just a label (delete labelName), delete an entry in current label (delete 3), or delete an entry from a certain label (delete labelName 2).  Return a String object to inform user of delete execution progress. |
| ArrayList<Task> | doSearch(Command command)  Return an ArrayList of Task objects that contain the search keywords. |
| String | doEdit(Command command)  Method is able to edit a task from current label or from a certain label.  Return a String object to inform user of edit execution progress. |
| String | doUndo(Command command)  Method is able to undo last command.  Return a String object to inform user of undo execution progress. |
| String | doDisplay(Command command)  Method is able to display all labels and tasks.  Return a String object to inform user of display execution progress. |
| String | doHide(Command command)  Method is able to alter the visibility of labels.  (If user does not wish to view them, he can choose to hide them from view)  Return a String object if label name does not exist. |
| String | doDone(Command command)  Method is able to mark tasks that have been completed.  Return a String object to inform user of done execution progress. |
| String | doLabel(Command command)  Method is able to create a new label. |

**Class TDTStorage**

public class TDTStorage implements ITDTStorage

The TDTStorage class takes in an ArrayList of Task objects and stores them in a text file. HashMap of Task objects also created to allow easy referencing. A stack is used, and a simple pop() operation will revert back to the last change (undo).

Field Summary

|  |  |
| --- | --- |
| Modifier and Type | Field and Description |
| private String | fileName  Contains name of textfile to be created. |
| private HashMap<String, ArrayList<Task>> | labelMap  Contains HashMap linking ArrayList<Task> to labels. |
| private String | currLabel  Label which user is currently located at. |
| private Stack<HashMap<String, ArrayList<Task>>> | undoStack  Stack containing the different progresses of the task manager. |
| private Stack<String> | labelPointerStack  Contains last accessed label. |
| private BufferedWriter | bw  Object used to write input into textfile. |

Constructor Summary

|  |  |
| --- | --- |
| Modifier | Constructor and Description |
| public | TDTStorage(String fileName)  Creates a TDTStorage object with the necessary HashMap and Stacks. |

Method Summary

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| void | readInitialise()  Access textfile and creates HashMap and Stacks to store the tasks and labels written in textfile. |
| void | write()  Write new tasks and labels into existing textfile. |
| Iterator<Task> | getTaskIterator()  Returns TaskIterator of HashMap of current label. |
| int | getLabelSize(String labelName)  Return number of tasks stored under the label labelName. |
| void | addTask(Task task)  Add task under current label. |
| HashMap<String, ArrayList<Task>> | copyLabelMap()  Returns a copy of a HashMap linking tasks to labels. |
| String | getFileName()  Returns fileName |
| void | setFileName(String fileName)  this.fileName = fileName |
| HashMap<String, ArrayList<Task>> | getLabelMap()  return labelMap |
| void | setLabelMap(HashMap<String, ArrayList<Task>> labelMap)  this.labelMap = labelMap |
| String | getCurrLabel()  Return currLabel |
| void | setCurrLabel(String currLabel)  this.currLabel = currLabel |
| Stack<HashMap<String, ArrayList<Task>>> | getUndoStack()  return undoStack |
| void | setUndoStack(Stack<HashMap<String, ArrayList<Task>>> undoStack)  this.undoStack = undoStack |
| Stack<String> | getLabelPointerStack()  return labelPointerStack |
| void | setLabelPointerStack(Stack<String> labelPointerStack)  this.labelPointerStack = labelPointerStack |