**UI CLASSES**

1. UI Class

|  |
| --- |
| **UI** |
| -Signal status  - HANDLE hConsole |
| - void displayBoard(Signal statusSignal)  - void setStatus(Signal statusSignal)  - void setInput(string textInput);  - void drawBanner()  - void writeWords(string words, int startH, int startW)  - void displayEntryList( vector<string>\* calendarEntryList, vector<string> \*generalEntryList )  + void coloredDisplayFormattedString(int,string)  + void setNormal()  + void drawBox()  + void didUKnowBox()  + Signal getStatus()  + void setScreenSize()  + void clearStatus()  + void gotoxy(int x,int y)  + void startingScreenDisplay()  + void mainScreenDisplay(vector<string>\* calendarEntryList, vector<string>\* generalEntryList) |

* 1. Signal getStatus()
     + get the status signal of UI displaying process
  2. void clearStatus()
     + clear the status signal of UI process to default CLEAR signal
  3. void startingScreenDisplay()
     + show the starting screen to user at the beginning of the program
  4. void mainScreenDisplay(vector<string>\* calendarEntryList, vector<string>\* generalEntryList)
     + show the main screen to interact with the user
  5. void gotoxy(int x, int y)
     + moves the cursor to specific coordinate of the screen
  6. void setNormal()
     + Sets control attribute
  7. Void drawBox()
     + Draws a box for text background
  8. void didUKnowBox()
     + Draws the “Did you know box”

1. IO Class

|  |
| --- |
| **IO** |
| - Signal status  - string input |
| - void setStatus(Signal statusSignal)  - void setInput(string textInput);  + Signal getStatus()  + void clearStatus()  + void getRawInput()  + string getText()  + void displayMessage(string output) |

* 1. void getRawInput()
     + get user input string through command line
  2. void displayMessage(string output)
     + display feedback message string of the system to user
  3. Signal getStatus()
     + get the status signal of IO displaying process
  4. void clearStatus()
     + clear the status signal of IO process to default CLEAR signal

**EXECUTOR CLASSES**

1. Executor Class (Super Class)

|  |
| --- |
| **Executor** |
| # Signal status  # int findBlockIndex(string details, int blockLocation)  # string extractField(string details, int startLocation)  # int extractIndex(string details)  # string extractDescription(string details)  # string extractLocation(string details)  # string extractTime(string details)  # string extractDate(string details)  # int extractPriority(string details) |
| + void execute() (virtual)  + void undo() (virtual)  + Signal getStatus() |

* 1. void execute()
     + executes the predefined function of the executor
  2. void undo()
     + Undo the last changes made by the executor
  3. void getStatus
     + Returns the status of the executor

1. AddExecutor Class

|  |
| --- |
| **AddExecutor** |
| # Signal status  # int findBlockIndex(string details, int blockLocation)  # string extractField(string details, int startLocation)  # int extractIndex(string details)  # string extractDescription(string details)  # string extractLocation(string details)  # string extractTime(string details)  # string extractDate(string details)  # int extractPriority(string details)  - vector<string>\* \_calendarEntryList  - vector<string>\* \_generalEntryList;  - vector<string> \_undoGeneralEntryList  - vector<string> \_undoCalendarEntryList  - string \_details |
| + void execute()  + void undo()  + Signal getStatus() |

* 1. void execute()
     + executes the add entry functionality to storage
     + the entry can be saved either to the Calendar Entry List or the General Entry List depending on the format of the entry
  2. void undo()
     + Undo the last changes made by the executor
  3. void getStatus
     + Returns the status of the executor

1. DeleteExecutor Class

|  |
| --- |
| **DeleteExecutor** |
| # Signal status  # int findBlockIndex(string details, int blockLocation)  # string extractField(string details, int startLocation)  # int extractIndex(string details)  # string extractDescription(string details)  # string extractLocation(string details)  # string extractTime(string details)  # string extractDate(string details)  # int extractPriority(string details)  - vector<string>\* \_calendarEntryList  - vector<string>\* \_generalEntryList;  - vector<string> \_undoGeneralEntryList  - vector<string> \_undoCalendarEntryList  - string \_details |
| + void execute()  + void undo()  + Signal getStatus() |

* 1. void execute()
     + executes the delete entry functionality from storage
     + the entry can be deleted either from the Calendar Entry List or the General Entry List depending on the index chosen
  2. void undo()
     + Undo the last changes made by the executor
  3. void getStatus
     + Returns the status of the executor

1. ExitExecutor Class

|  |
| --- |
| **ExitExecutor** |
| # Signal status  # int findBlockIndex(string details, int blockLocation)  # string extractField(string details, int startLocation)  # int extractIndex(string details)  # string extractDescription(string details)  # string extractLocation(string details)  # string extractTime(string details)  # string extractDate(string details)  # int extractPriority(string details)  -vector<string>\* \_generalEntryList;  StorageHandler\* \_store;  bool\* \_quit; vector<string>\* \_calendarEntryList |
| + void execute()  + Signal getStatus() |

* 1. void execute()
     + signals the storage handler to save the data and exits the program
  2. void getStatus
     + Returns the status of the executor

1. SearchExector Class

|  |
| --- |
| **SearchExecutor** |
| # Signal status  # int findBlockIndex(string details, int blockLocation)  # string extractField(string details, int startLocation)  # int extractIndex(string details)  # string extractDescription(string details)  # string extractLocation(string details)  # string extractTime(string details)  # string extractDate(string details)  # int extractPriority(string details)  - vector<string>\* \_entryList;  - vector<string>\* \_matchedEntryList;  - vector<string> \_undoEntryList;  - vector<string> \_undoMatchedEntryList;  - string \_details; |
| + void execute()  + void undo()  + Signal getStatus() |

* 1. void execute()
     + executes the search entry functionality from storage by using the user’s input keyword
     + the entry can be either from the Calendar Entry List or the General Entry List and it will be stored inside the matchedEntryList;
  2. void undo()
     + Undo the last changes made by the executor
  3. void getStatus
     + Returns the status of the executor

1. UpdateExecutor Class

|  |
| --- |
| **UpdateExecutor** |
| # Signal status  # int findBlockIndex(string details, int blockLocation)  # string extractField(string details, int startLocation)  # int extractIndex(string details)  # string extractDescription(string details)  # string extractLocation(string details)  # string extractTime(string details)  # string extractDate(string details)  # int extractPriority(string details)  - vector<string>\* \_calendarEntryList  - vector<string>\* \_generalEntryList;  - vector<string> \_undoGeneralEntryList  - vector<string> \_undoCalendarEntryList  - string \_details |
| + void execute()  + void undo()  + Signal getStatus() |

* 1. void execute()
     + executes the update entry functionality from storage
     + the updated entry can be either from the Calendar Entry List or the General Entry List depending on the index chosen
  2. void undo()
     + Undo the last changes made by the executor
  3. void getStatus
     + Returns the status of the executor

**HANDLER CLASSES**

1. UI Handler

|  |
| --- |
| **UIHandler** |
| - StatusHandler sh  - void setUIStatus(Signal statusSignal)  - string interpreteSignal(Signal outSignal)  - IO io;  - UI ui;  - Signal UIstatus |
| + Signal getStatus()  + void setStatus()  + void getInput()  + void displayMessage(Signal outSignal)  + void displayMessage(vector<string>\* result)  + void displayMessage(string result)  + bool void startingScreenDisplay()  + void mainScreenDisplay(vector<string>\* calendarEntryList, vector<string>\* generalEntryList)  + string retrieveInput()  + Signal getStatus()  + void clearStatus() |

* 1. void setStatus()
     + Sets the status of the UI handler after printing UI
  2. void getInput()
     + This operation is used to get input message from the user. It calls IO to get user input through command line and store it in the private string named input
  3. void displayMessage(Signal outSignal)
     + This operation is used to display output message to the user. It takes in the feedback signal passed by main(), call private interpretSignal(Signal) to intepret to a string and calls IO to output to the user
  4. void displayMessage(vector<string>\* result)
     + Displays messages or strings that have been stored in the string. We use this to output the search result.
  5. void displayMessage(string result)
     + Displays a string containing the operation result.
  6. void startingScreenDisplay()
     + This operation is used to display the startup screen to the user at the beginning of the program
  7. void mainScreenDisplay(vector<string>\* calendarEntryList, vector<string>\* generalEntryList)
     + This operation is used to display the main screen through UI class which interfaces the interaction with user. It will call the UI to display the main screen which splits the screen into 3 part, the Calendar List, the General List, and the input screen.
  8. string retrieveInput()
     + This method retrieves user’s input after processed in the IO class
  9. Signal getStatus()
     + This method returns the status of UI handler
  10. void clearStatus();
      + This method clears the UI handler’s status

1. Function Handler

|  |
| --- |
| **FunctionHandler** |
| - StatusHandler sh  - vector<string> ram  - Signal fxStatus  - StorageHandler store  - stack<Executor\*> undoStk |
| + Signal getStatus()  + void setStatus()  + void execute(string input, bool\* quit,vector<string>\* generalEntryList, vector<string>\* calendarEntryList, vector<string>\* diduknowBoxList) |

1. void setStatus()
   * + sets the status attribute of the Function Handler class
2. Signal getStatus()
   * + Retrieves the current status of the class
3. void execute(string input, bool\* quit,vector<string>\* generalEntryList, vector<string>\* calendarEntryList, vector<string>\* diduknowBoxList)
   * + The operation is periodically called in main(). It will handle the flow of the logic component.
4. Language Handler

|  |
| --- |
| **LanguageHandler** |
| - StatusHandler sh;  - Signal command;  - Signal langStatus;  - string details; |
| - bool leap(int year)  - bool isDate(string date)  - bool isTime(string time)  - bool isInt(string inx)  - bool isLogicDate(string date)  - bool isLogicTime(string time)  - bool isLogicPriority(string priority)  - void encoder(string input, Signal command)  - void setCommand(string userCommand)  + Signal getStatus()  + void separate(string userInput) throw (string)  + Executor\* pack(bool\* quit, vector<string>\* calendarEntryList, vector<string>\* generalEntryList,vector<string>\* diduknowBoxList, StorageHandler\* store) |

1. void getStatus()
   * + Retrieves the status of Language Handler
2. void separate(string userInput)
   * + Separates user input's string into 2 parts, the input and the string to be processed
3. Executor\* pack(bool\* quit, vector<string>\* calendarEntryList, vector<string>\* generalEntryList,vector<string>\* diduknowBoxList, StorageHandler\* store)
   * + Creates an appropriate executor based on the commands
4. Storage Handler

|  |
| --- |
| **StorageHandler** |
| - char buffer[MAXMIUM\_WORDS]  - static string DataBaseFile  - static string DataBaseTempFile |
| + Signal getStatus()  + void setStatus()  + void readData(vector<string> \*ram)  + void writeData(vector<string> \*ram)  + bool checkFileExistence(string filePath, string fileName)  + void disassociateFile(fstream & file)  + void associateFile(string filePath, string fileName, fstream & file, OPEN\_TYPE mode)  + void deleteFile(string filePath, string fileName)  + void renameFile(string filePath, string oriName, string newName)  + void replaceFile(string oriPath, string oriName, string repName) |

1. void setStatus()
   * + Sets the status of the language handler after input processing
2. void readData(vector<string> \*ram)
   * + Read the data stored in the file, and put it in a vector of string.
3. void writeData(vector<string> \*ram)
   * + Write the data stored to the file, the data is stored in a vector of string.
4. bool checkFileExistence(string filePath, string fileName)
   * + Checks whether the file located in filePath and fileName do exist
5. void disassociateFile(fstream & file)
   * + Removes the file association within the program
6. void associateFile(string filePath, string filename, fstream & file, OPEN\_TYPE mode)
   * + Associate files to the program
7. void deleteFile(string filePath, string fileName)
   * + Deletes text file with the same path and name as filePath and fileName
8. void renameFile(string filePath, string oriName, string newName)
   * + Renames the file
9. void replaceFile(string oriPath, string oriName, string repName)
   * + Replace an old file with the newly updated files
10. StatusHandler

|  |
| --- |
| **StatusHandler** |
|  |
| + bool success(Signal signal)  + bool error(Signal signal) |

a. bool success(Signal signal)

* + - Check the signal whether it is a success or fail

b. bool error (Signal signal)

* + - Check if the signal is an error.