**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” **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
       - Parameter = void
       - Return =void
  2. void undo()
     + Undo the last changes made by the executor
       - Parameter = void
       - Return = void
  3. Signal getStatus()
     + Returns the status of the executor
       - Parameter= void
       - Return = Signal status of the executor

1. AddExecutor Class

|  |
| --- |
| **AddExecutor** |
| # Signal status  - vector<string>\* \_calendarEntryList  - vector<string>\* \_generalEntryList;  - vector<string> \_undoGeneralEntryList  - vector<string> \_undoCalendarEntryList  - string \_details |
| # 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()  + 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
       - Parameter = void
       - Return =void
  2. void undo()
     + Undo the last changes made by the executor
       - Parameter = void
       - Return =void
  3. Signal getStatus
     + Returns the status of the executor
       - Parameter = void
       - Return = Signal status of the executor

1. DeleteExecutor Class

|  |
| --- |
| **DeleteExecutor** |
| # Signal status  - vector<string>\* \_calendarEntryList  - vector<string>\* \_generalEntryList;  - vector<string> \_undoGeneralEntryList  - vector<string> \_undoCalendarEntryList  - string \_details |
| # 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()  + 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
       - Parameter = void
       - Return =void
  2. void undo()
     + Undo the last changes made by the executor
       - Parameter = void
       - Return =void
  3. void getStatus
     + Returns the status of the executor
       - Parameter = void
       - Return = Current signal status of the executor

1. ExitExecutor Class

|  |
| --- |
| **ExitExecutor** |
| # Signal status  -vector<string>\* \_generalEntryList;  - StorageHandler\* \_store;  - bool\* \_quit; vector<string>\* \_calendarEntryList |
| # 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()  + Signal getStatus() |

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

1. SearchExector Class

|  |
| --- |
| **SearchExecutor** |
| # Signal status  - vector<string>\* \_calendarEntryList;  - vector<string>\* \_generalEntryList;  - vector<string>\* \_matchedEntryList;  - vector<string> \_undoEntryList;  - vector<string> \_undoMatchedEntryList;  - string \_details; |
| # 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()  + 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;
       - Parameter = void
       - Return =void
  2. void undo()
     + Undo the last changes made by the executor
       - Parameter = void
       - Return =void
  3. void getStatus
     + Returns the status of the executor
       - Parameter = void
       - Return = current signal status of the executor

1. UpdateExecutor Class

|  |
| --- |
| **UpdateExecutor** |
| # Signal status  - vector<string>\* \_calendarEntryList  - vector<string>\* \_generalEntryList;  - vector<string> \_undoGeneralEntryList  - vector<string> \_undoCalendarEntryList  - string \_details |
| # 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()  + 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
       - Parameter = void
       - Return =void
  2. void undo()
     + Undo the last changes made by the executor
       - Parameter = void
       - Return =void
  3. void getStatus
     + Returns the status of the executor
       - Parameter = void
       - Return = current signal status of the executor

**HANDLER CLASSES**

1. Function Handler

|  |
| --- |
| **FunctionHandler** |
| - StatusHandler sh  - 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
       - Parameter = void
       - Return =void
2. Signal getStatus()
   * + Retrieves the current status of the class
       - Parameter = void
       - Return = current signal status of the function handler
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.
       - Parameter
         1. string input = passing the input string to the function
         2. bool\* quit = pass the pointer for Boolean function that contains whether the user wants to quit the program
         3. vector<string>\* generalEntryList is the vector contains general entries
         4. vector<string>\* calendarEntryList is the vector that contains calendar entries
         5. vector<string>\* diduknowBoxList is the vector that contains hints and general feedbacks in using the programs.
       - Return =void
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
       - Parameter = void
       - Return =void
2. void separate(string userInput)
   * + Separates user input's string into 2 parts, the input and the string to be processed
       - Parameter = string that contains user’s input
       - Return =void
3. Executor\* pack(bool\* quit, vector<string>\* calendarEntryList, vector<string>\* generalEntryList,vector<string>\* diduknowBoxList, StorageHandler\* store)
   * + Creates an appropriate executor based on the commands Parameter
       - Parameter
         1. string input = passing the input string to the function
         2. bool\* quit = pass the pointer for Boolean function that contains whether the user wants to quit the program
         3. vector<string>\* generalEntryList is the vector contains general entries
         4. vector<string>\* calendarEntryList is the vector that contains calendar entries
         5. vector<string>\* diduknowBoxList is the vector that contains hints and general feedbacks in using the programs.
       - Return = void
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 storage after input processing
       - Parameter = void
       - Return =void
2. void readData(vector<string> \*ram)
   * + Read the data stored in the file, and put it in a vector of string.
       - Parameter = vector that contains all of the string being used on the fly by the program
       - Return =void
3. void writeData(vector<string> \*ram)
   * + Write the data stored to the file, the data is stored in a vector of string.
       - Parameter = vector that contains all of the string being used on the fly by the program
       - Return =void
4. 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
      * Parameter = signal that is going to be checked
      * Return = the validity of the signal to the specification

b. bool error (Signal signal)

* + - Check if the signal is an error.
      * Parameter = signal that is going to be checked
      * Return = the validity of the signal to the specification