Bilkent University Computer Science Department

Object Oriented Software Engineering

CS-319

Summer Term Project Design Report

Dear Diary

- Abdurrezak Efe 21301883
- Ayşegül Sümeyye Kütük 20900538
- Enes Kavak 21302618

1. Introduction

1.1 Purpose of the system

Dear Diary is a neat desktop application that offers its users a diary experience via an elegant interface allowing to keep the archive of writings, albums, special days/dates through the different functions of the application software structure. Comparing to the other diary applications in the market, Dear Diary looks a lot fancier, works with a more effective environment and offers a more practical usage to the candidate diary writers. Within the project improvement, we aim to encourage people to share their experiences in a reliable environment and to make their own personal journal of daily events, appointments, secrets and feelings. Through the practical usage of the application system, Dear Diary serves as a tool for its users to acquire the habit of writing daily diary.

1.2 Design Goals

To determine the design goals is a must in terms of building the details of the system in order to declare the application qualities. Hence, we made quite an effort to build our design goals on the functional and non-functional requirements of the system that we summed elaborately in the analysis report. Our specific design goals are presented below.

1.2.1 End User Criteria:

Ease of Use: Since it is one of our goals to offer a practical usage and encourage people to write more, we tried to build an application software with a neat and simple design. In terms of usage, our application interface provide its users a distinct menu by which the users will easily navigate themselves through the desired operations they wish to fulfill. Since all the operation transitions are depending on the user mouse input, the usage of the system is quite explicit.

Ease of Learning: Since we present a brand new application, the operation flow of the system will be unknown for the user point of view. The buttons and the menu details are assigned with reasonable tags. With that, we try to help the users to understand which button will generate which operation. Since the user interface have become quite clear to follow by this way, we thought it would be trivial to add an instructive user guide.

1.2.2 Maintenance Criteria:

Extendibility: Extensions of a software product is certainly essential in terms of renewing itself and improvement. New components, features or the updates upon the built in functions will be generated in order to get more interest and attention from the user's point of view. Also, as the software technology of the machinery changes through time, the adaptability of the application will be ensured through new extensions and updates.

Portability: Portability criteria must be taken into consideration since to keep the application compatible and enduring is crucial in order to reach a wide range of user. Our decision for the implementation of the project is marked to be in Java since it is most likely to provide platform independency by its JVM.

Modifiability: To modify the attributes and functionalities of a product must always be possible in terms of eliminating and easily dissolving any probable performance/maintenance problems through its lifetime. For the Dear Diary software system, we aimed to minimize the coupling of the subsystems as much as possible, to avoid great impacts on system components by a probable change. In this way, any modification at any time will be an easy task to achieve, we think.

Reliability: We aim to finalize our product with a bug free and homegenous system that allows the user have a proper and successful experience every time they attempt to use the application. It is a must for us to provide a strong system that will not crash under any circumstances and whatever input is given by the user. In order to deal with this issue, we are planning to generate regular tests on the application within each stage of development and also during its lifetime in the market.

1.2.3 Performance Criteria:

Response Time: In terms of performance, the user must be able to interact in a fast manner with the application in order to keep the attention and interest stable. In order not to give users a troubled experience and cause the distraction of their interest, we tried to build our system in such a way that it will respond immediately by the user's choice of input. Every action through the usage of the application will be instantaneously reacted by the system.

1.2.4 Trade Offs:

Performance vs. Memory:

The user satisfaction is the biggest concern for us as the project development team. By this reason, we are thrilled to build the application in such a way that the flow of operations from the user's choice of input will be smooth enough to keep the user active and devoted to what they do. We diminished the qualities concerning the memory criteria in order to make the transitions properly working.

Efficiency vs Reusability:

Since we do not have any determined plans to integrate our application system to a different environment or any other product with a whole new idea, we did not concerned about the reusability of our system. By this reason, we designed all the objects and object related classes according to the our start idea in order not to end up with a source code that is too complex to manage, modify and adapt to any probable extension. All we think that matters most is the criteria of efficiency of the system performance. Hence we mostly focused on how we can make the application more efficient in terms of design stages by us, engineers and usage by the user perspective.

1.3 Definitions, acronyms, and abbreviations

Abbreviations:

JVM: [1] Java Virtual Machine

1.4. References

[1] http://en.wikipedia.org/wiki/Java_(programming_language)

2. Software Architecture

2.1. Overview

In this part, we will try to explain how we decomposed the software we develop into subsystems in order to make it robust and efficient. As there is no multilayer structure in our project that needs a special kind of design pattern, our design is neat and straightforward.

2.2. Architectural Styles

2.2.1 Layers

The architectural structure of Dear Diary possesses three layers. Namely:

- User Interface
- Diary Manager
- Diary Entities

Our system is user dependent which implies that the first layer that interacts with lower layers of the program is the user interface. We intended to provide our users with a fancy and easy-to-use interface.

The second layer is Diary Manager which controls almost everything. Although is not coded in a new class, Diary Manager controls every delete, save, update operation there is. However, it needs another sublayer which is Diary Entities layer.

Diary entities layer is the lowest level of abstraction in our program and makes every operation possible by providing corresponding entities and objects.

2.2.2 Model View Controller

By using Model View Controller model, we made it easier for the program to perform its tasks. As MVC proposes a system that has three basic subsystems responsible for the data, its processing and its displayment in a desired way.

In our system, the model has the raw data of users, days, entries, albums and everyting Dear Diary stores and processes. Meanwhile, view displays the data that is being modified or updated in runtime. Finally, controller manipulates the model data to update the system everytime the user interacts with the system.

2.3. Hardware / Software Mapping

Our program has been being implemented in Java programming languase so it uses the latest JDK. Meanwhile, Dear Diary interacts with users by a keyboard to write entries, notes, album descriptions etc. and mouse to click on buttons.

The system's requirements are minimal as it is a small sized desktop application with data stored in .txt and .ser files.

As Java provides its developers with Serializable interface for objects to be stored in .ser files, we had no hardship in storing objects in .ser files with security.

Our system can easily be launched in any operating system as long as the OS has an up-to-date Java compiler. Finally, our program does not need any internet connection to work properly.

2.4. Persistent Data Management

As Dear Diary is not a very advanced based complex project, we needed no well designed databases or any advanced data structure to store the data of the users and its sub files. Thus, Dear Diary stores all the data it has in .txt files which keep track of logins, signups, dates of signups and logins; meanwhile, .ser files store the updated data of any user and its components.

When Dear Diary is launched, no file is on use as long as no sign up or login operation is performed. When a login/signup is done, .txt files make sure that the new user has a valid username with a sufficient password or if the user entered his password correctly to enter the system.

After a successfull login operation, a particular .ser file is in use. However, as there is no way for Java forms to get information from another, the correspondng .ser file is processed in every Java form once they are opened.

Whenever an addition, save, deletion or update operation is performed; the corresponding .ser file is updated accordingly in runtime.

2.5. Access Control and Security

Dear Diary application, as the name proposes, needs a strong password protection and data privacy. Therefore, the users will not be able to see each others' entries, notes, talks, albums and not even days. To provide users with privacy, our system dictates that the user has a minimum seven characters long password with a unique username. Actually, the mentioned signup, login structure was the hardest part to implement and provide the users with. Nevertheless, a diary with no protection of the data is as useles as a bank with no gate.

Additionally, as our system does not have any kind of third party connection such as internet, bluetooth etc. it is not vulnerable to any attack.

2.6. Boundary Conditions

Initialization

The program needs no installation, it can easily be launched from .java file using a Java compiler. When launched, the program will handle the rest by itself.

Termination

Dear Diary can be terminated by x icon on top right/left corner(depending on the OS it is run on). Since the data is updated in runtime, no data loss is of concern. However, as mentioned before the data that was not saved will not be of use in future if the form is disposed by the user.

Additionally, there is also a logout button in every frame of Dear Diary which simply return to the login page.

Error

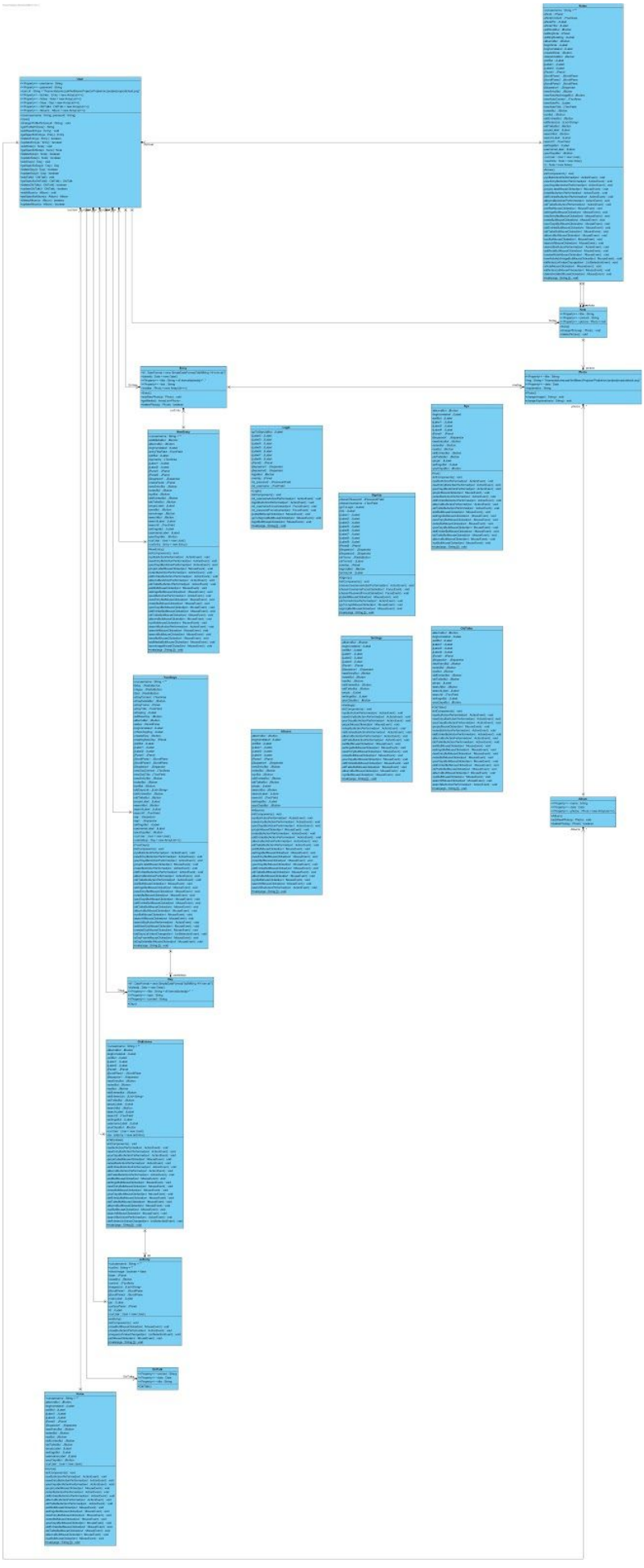
As the program is coded with Java, it handles exceptions very well. Whenever, a file is corrupted the system simply do not load the data stored in that file. On the other hand, although the program is robust enough to be run many times consecutively, cannot save data in some situations like non-responding frames due to RAM's situation.

3. Subsystems

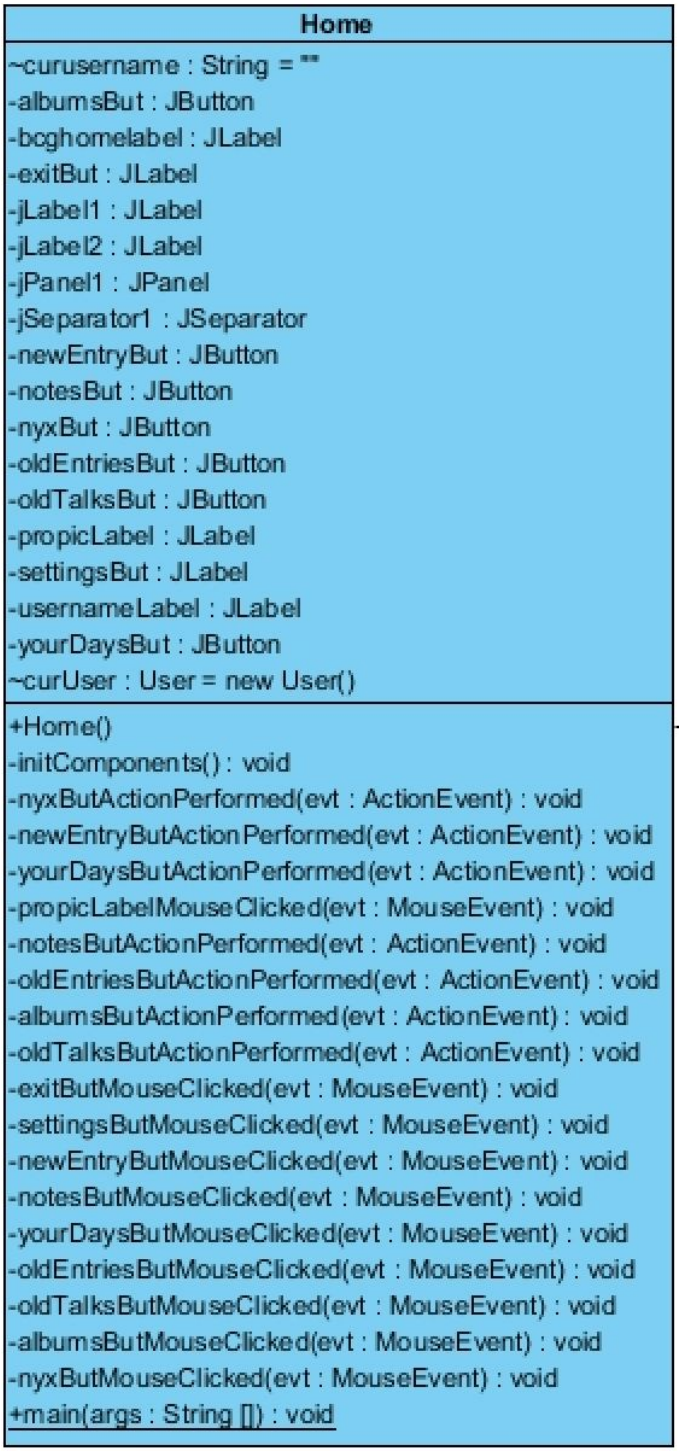
In this section we will provide the detailed information about the interfaces of our subsystems.

User Interface Subsystem Interface

User interface provides users with a fancy and easy to use layout. Besides, it helps to send data between frames, crucial classes and so on. Here is a full view of the program with its interface forms.



Home Class


Home
~curusername : String = ""
-albumsBut : JButton
-bcghomelabel : JLabel
-exitBut : JLabel
-jLabel1 : JLabel
-jLabel2 : JLabel
-jPanel1 : JPanel
-jSeparator1 : JSeparator
-newEntryBut : JButton
-notesBut : JButton
-nyxBut : JButton
-oldEntriesBut : JButton
-oldTalksBut : JButton
-propicLabel : JLabel
-settingsBut : JLabel
-usernameLabel : JLabel
-yourDaysBut : JButton
~curUser : User = new User()
+Home()
-initComponents() : void
-nyxButActionPerformed(evt : ActionEvent) : void
-newEntryButActionPerformed(evt : ActionEvent) : void
-yourDaysButActionPerformed(evt : ActionEvent) : void
-propicLabelMouseClicked(evt : MouseEvent) : void
-notesButActionPerformed(evt : ActionEvent) : void
-oldEntriesButActionPerformed(evt : ActionEvent) : void
-albumsButActionPerformed(evt : ActionEvent) : void
-oldTalksButActionPerformed(evt : ActionEvent) : void
-exitButMouseClicked(evt : MouseEvent) : void
-settingsButMouseClicked(evt : MouseEvent) : void
-newEntryButMouseClicked(evt : MouseEvent) : void
-notesButMouseClicked(evt : MouseEvent) : void
-yourDaysButMouseClicked(evt : MouseEvent) : void
-oldEntriesButMouseClicked(evt : MouseEvent) : void
-oldTalksButMouseClicked(evt : MouseEvent) : void
-albumsButMouseClicked(evt : MouseEvent) : void
-nyxButMouseClicked(evt : MouseEvent) : void
+main(args : String []) : void

Attributes:

public String curusernane: This string holds the name of the user that logged in.

public JLabel exitBut: This label works as a button for logging out from the system and common for all frames. That is, all frames has the label as a button.

public JButton newEntryBut: This button works for opening up the corresponding frame implied by the button name, in this case “New Entry”. When the user wants to write a new entry, s/he must use this button to be able to open the frame s/he can write it.

public JButton notesBut: This button is to open up the corresponding frame implied by the button name, in this case “Notes”.

public JButton nyxBut: This button is to open up the corresponding frame implied by the button name, in this case, our chatbot “Talk to Nyx”.

public JButton oldEntriesBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Entries”.

public JButton oldTalksBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Talks”.

public JButton yourDaysBut: This button is to open up the corresponding frame implied by the button name, in this case “Days”.

private JLabel propicLabel: This label holds the profile picture as the name proposes.

private User curUser: This User object collects all the data from methods about the user that logged in.

Constructors:

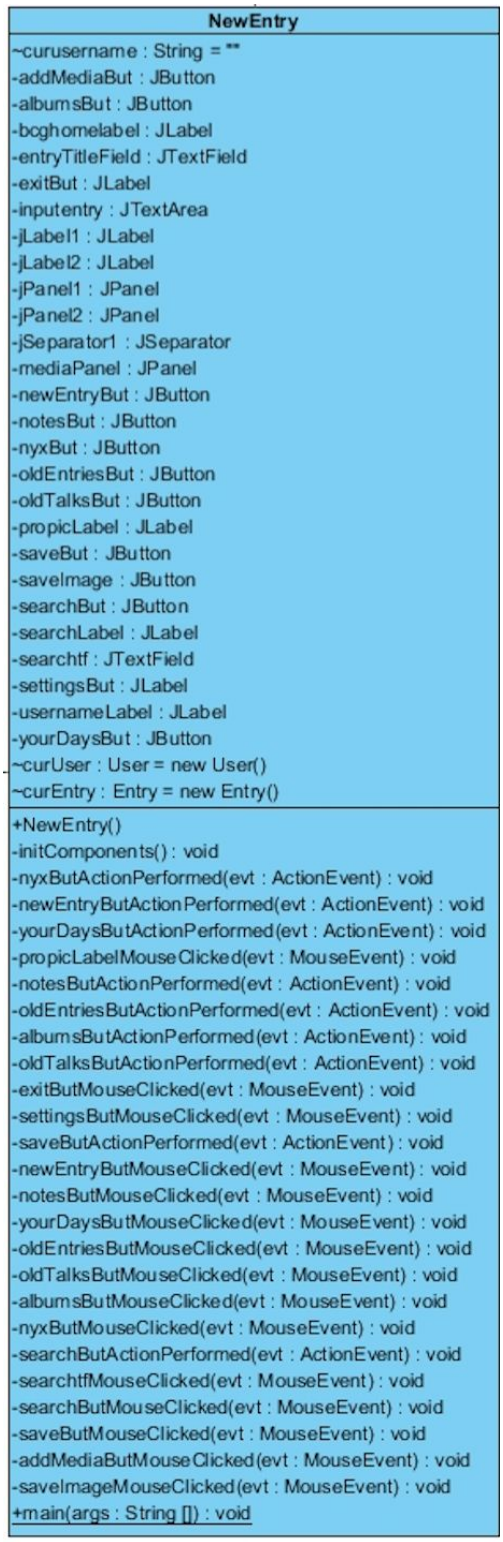
public Home: Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

private void propicLabelMouseClicked: This method works for opening the window that is specific to the corresponding operation, in this case updating the profile picture.

private void notesButMouseClicked, oldEntriesButMouseClicked, newEntryButMouseClicked, yourDaysButMouseClicked, albumsButMouseClicked, oldTalksButMouseClicked, nyxButMouseClicked, settingsButMouseClicked, exitButMouseClicked: These functions simply change the frame to the corresponding frames proposed by the names. These functions are common for all of the frames except Login and Sign Up frames.

New Entry Class



Attributes:

public String curusernane: This string holds the name of the user that logged in.

public JLabel bcghomeLabel: This label holds the background image of the current frame.

public JButton addMediaBut: This button is to open up the “Add Media” window to let the user add images to their new entry content.

public JLabel exitBut: This label works as a button for logging out from the system and common for all frames. That is, all frames has the label as a button.

public JButton albumsBut: This button is to open up the corresponding frame implied by the button name, in this case “Albums”.

public JTextField entryTitleField: This text field holds the information of the title of the entry which is taken from the user during the new entry creation.

public JTextArea inputentry: This text area holds the content of the new entry that is written by the user.

public JButton saveBut: This button is to save the content created by the user in the text area and text field.

public JButton saveImage: This button is to save the media that users choose to add to their new entry.

public JLabel settingsBut: This label is to open up the settings section and let users do changes as they wish.

public JLabel usernameLabel: This labels holds the username information on the left part of the screen.

public JButton newEntryBut: This button works for opening up the corresponding frame implied by the button name, in this case “New Entry”. When the user wants to write a new entry, s/he must use this button to be able to open the frame s/he can write it.

public JButton notesBut: This button is to open up the corresponding frame implied by the button name, in this case “Notes”.

public JButton nyxBut: This button is to open up the corresponding frame implied by the button name, in this case, our chatbot “Talk to Nyx”.

public JButton oldEntriesBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Entries”.

public JButton oldTalksBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Talks”.

public JButton yourDaysBut: This button is to open up the corresponding frame implied by the button name, in this case “Days”.

private JLabel propicLabel: This label holds the profile picture as the name proposes.

private User curUser: This User object collects all the data from methods about the user that logged in.

Constructor:

public newEntry: Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

private void propicLabelMouseClicked: This method works for opening the window that is specific to the corresponding operation, in this case updating the profile picture.

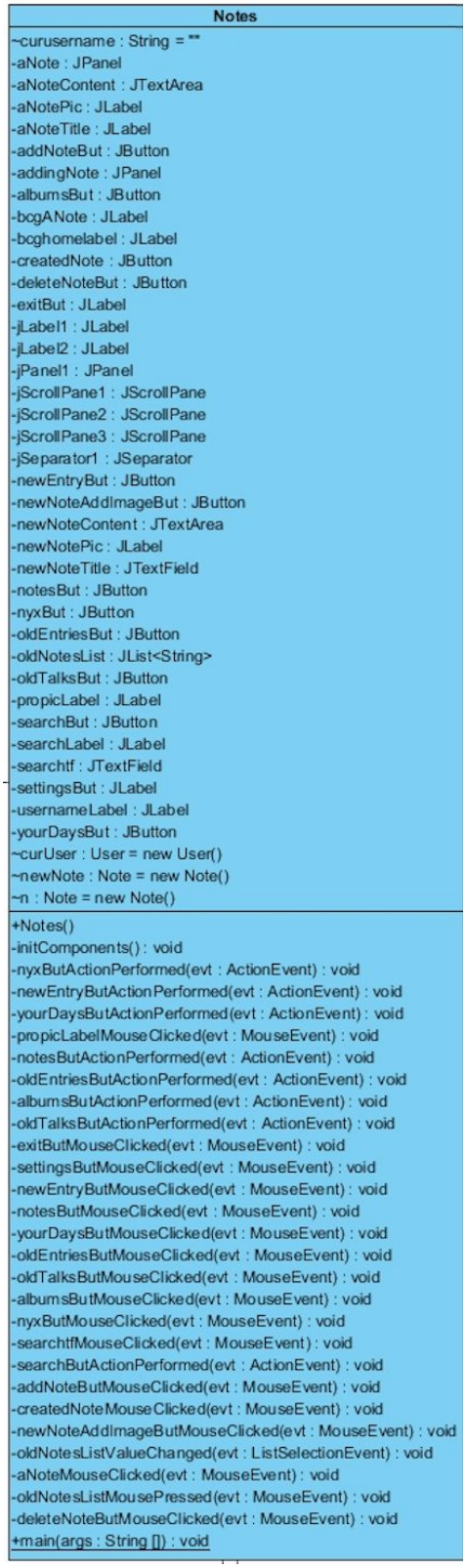
private void notesButMouseClicked, oldEntriesButMouseClicked, newEntryButMouseClicked, yourDaysButMouseClicked, albumsButMouseClicked, oldTalksButMouseClicked, nyxButMouseClicked, settingsButMouseClicked, exitButMouseClicked: These functions simply change the frame to the corresponding frames proposed by the names. These functions are common for all of the frames except Login and Sign Up frames.

private void saveButMouseClicked: This function simply saves the new entry that is created by the user via the entry interface.

private void saveImageMouseClicked: This function simply saves the new media that is added by the user to the new entry content.

private void addMediaButMouseClicked: This function simply adds media that is chosen by the user to the new entry content.

Notes Class


Notes
~curusername : String = ""
-aNote : JPanel
-aNoteContent : JTextArea
-aNotePic : JLabel
-aNoteTitle : JLabel
-addNoteBut : JButton
-addingNote : JPanel
-albumsBut : JButton
-bcgANote : JLabel
-bcghomelabel : JLabel
-createdNote : JButton
-deleteNoteBut : JButton
-exitBut : JLabel
-jLabel1 : JLabel
-jLabel2 : JLabel
-jPanel1 : JPanel
-jScrollPane1 : JScrollPane
-jScrollPane2 : JScrollPane
-jScrollPane3 : JScrollPane
-jSeparator1 : JSeparator
-newEntryBut : JButton
-newNoteAddImageBut : JButton
-newNoteContent : JTextArea
-newNotePic : JLabel
-newNoteTitle : JTextField
-notesBut : JButton
-nyxBut : JButton
-oldEntriesBut : JButton
-oldNotesList : JList<String>
-oldTalksBut : JButton
-propicLabel : JLabel
-searchBut : JButton
-searchLabel : JLabel
-searchtf : JTextField
-settingsBut : JLabel
-usernameLabel : JLabel
-yourDaysBut : JButton
~curUser : User = new User()
~newNote : Note = new Note()
~n : Note = new Note()
+Notes()
-initComponents() : void
-nyxButActionPerformed(evt : ActionEvent) : void
-newEntryButActionPerformed(evt : ActionEvent) : void
-yourDaysButActionPerformed(evt : ActionEvent) : void
-propicLabelMouseClicked(evt : MouseEvent) : void
-notesButActionPerformed(evt : ActionEvent) : void
-oldEntriesButActionPerformed(evt : ActionEvent) : void
-albumsButActionPerformed(evt : ActionEvent) : void
-oldTalksButActionPerformed(evt : ActionEvent) : void
-exitButMouseClicked(evt : MouseEvent) : void
-settingsButMouseClicked(evt : MouseEvent) : void
-newEntryButMouseClicked(evt : MouseEvent) : void
-notesButMouseClicked(evt : MouseEvent) : void
-yourDaysButMouseClicked(evt : MouseEvent) : void
-oldEntriesButMouseClicked(evt : MouseEvent) : void
-oldTalksButMouseClicked(evt : MouseEvent) : void
-albumsButMouseClicked(evt : MouseEvent) : void
-nyxButMouseClicked(evt : MouseEvent) : void
-searchtfMouseClicked(evt : MouseEvent) : void
-searchButActionPerformed(evt : ActionEvent) : void
-addNoteButMouseClicked(evt : MouseEvent) : void
-createdNoteMouseClicked(evt : MouseEvent) : void
-newNoteAddImageButMouseClicked(evt : MouseEvent) : void
-oldNotesListValueChanged(evt : ListSelectionEvent) : void
-aNoteMouseClicked(evt : MouseEvent) : void
-oldNotesListMousePressed(evt : MouseEvent) : void
-deleteNoteButMouseClicked(evt : MouseEvent) : void
+main(args : String []) : void

Attributes:

public String curusernane: This string holds the name of the user that logged in.

public JLabel bcghomeLabel: This label holds the background image of the current frame.

public JButton addMediaBut: This button is to open up the “Add Media” window to let the user add images to their new entry content.

public JLabel exitBut: This label works as a button for logging out from the system and common for all frames. That is, all frames has the label as a button.

public JButton albumsBut: This button is to open up the corresponding frame implied by the button name, in this case “Albums”.

public JLabel settingsBut: This label is to open up the settings section and let users do changes as they wish.

public JLabel usernameLabel: This labels holds the username information on the left part of the screen.

public JButton newEntryBut: This button works for opening up the corresponding frame implied by the button name, in this case “New Entry”. When the user wants to write a new entry, s/he must use this button to be able to open the frame s/he can write it.

public JButton notesBut: This button is to open up the corresponding frame implied by the button name, in this case “Notes”.

public JButton nyxBut: This button is to open up the corresponding frame implied by the button name, in this case, our chatbot “Talk to Nyx”.

public JButton oldEntriesBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Entries”.

public JButton oldTalksBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Talks”.

public JButton yourDaysBut: This button is to open up the corresponding frame implied by the button name, in this case “Days”.

private JLabel propicLabel: This label holds the profile picture as the name proposes.

private User curUser: This User object collects all the data from methods about the user that logged in.

public JList<string> oldNotesList: This list holds the information of all the old notes recorded by the user.

private JButton addNoteBut: This button is to open a frame for the user to create a new note.

private JLabel aNoteTitle: This label is to hold the information of the title of the chosen one of the old notes.

private JTextArea aNoteContent: This text area is to hold the content of the new note that is created by the user.

private Jlabel aNotePic: This label is to show the image that is added by the user to the old note.

private JButton deleteNoteBut: This button is to delete the content of the chosen note by the user.

private JTextField newNoteTitle: This text field is to hold the title information of the note that is being created.

private JTextArea newNoteContent: This text area is to hold the content of the new note that is being created by the user.

private JButton newNoteAddImageBut: This button is to open up a window to add media to the note that is being created by the user.

private JButton createdNote: This button is to save the prepared note by the user and to add to the old notes list.

Constructor:

public Notes(): Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

private void propicLabelMouseClicked: This method works for opening the window that is specific to the corresponding operation, in this case updating the profile picture.

private void notesButMouseClicked, oldEntriesButMouseClicked, newEntryButMouseClicked, yourDaysButMouseClicked, albumsButMouseClicked, oldTalksButMouseClicked, nyxButMouseClicked, settingsButMouseClicked, exitButMouseClicked: These functions simply change the frame to the corresponding frames proposed by the names. These functions are common for all of the frames except Login and Sign Up frames.

private void newNoteAddImageButMouseClicked: This function is to execute the add media operation to the note that is being created by the user.

private void addNoteButMouseClicked: This function is to open a frame for the user to create a new note.

private void deleteNoteButMouseClicked: This function is to delete the content of the chosen note by the user.

private void newNoteAddImageButMouseClicked: This function is to open up a window to add media to the note that is being created by the user.

private void createdNoteMouseClicked: This function is to save the prepared note by the user and to add to the old notes list.

YourDays Class:

Attributes:

public String curusernane: This string holds the name of the user that logged in.

public JLabel bcghomeLabel: This label holds the background image of the current frame.

public JButton addMediaBut: This button is to open up the “Add Media” window to let the user add images to their new entry content.

public JLabel exitBut: This label works as a button for logging out from the system and common for all frames. That is, all frames has the label as a button.

public JButton albumsBut: This button is to open up the corresponding frame implied by the button name, in this case “Albums”.

public JLabel settingsBut: This label is to open up the settings section and let users do changes as they wish.

public JLabel usernameLabel: This labels holds the username information on the left part of the screen.

public JButton newEntryBut: This button works for opening up the corresponding frame implied by the button name, in this case “New Entry”. When the user wants to write a new entry, s/he must use this button to be able to open the frame s/he can write it.

public JButton notesBut: This button is to open up the corresponding frame implied by the button name, in this case “Notes”.

public JButton nyxBut: This button is to open up the corresponding frame implied by the button name, in this case, our chatbot “Talk to Nyx”.

public JButton oldEntriesBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Entries”.

public JButton oldTalksBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Talks”.

public JButton yourDaysBut: This button is to open up the corresponding frame implied by the button name, in this case “Days”.

private JLabel propicLabel: This label holds the profile picture as the name proposes.

private User curUser: This User object collects all the data from methods about the user that logged in.

public JList oldDaysList: This list shows all the days created before on a label. The list items are clickable.

public JButton addNewDay: This buttons opens up a frame and allows the user to determine credentials of the day being created.

public JTextArea newDayTitle: This text area gets the information about the title from the user. If it is left empty, the date is chosen as the title.

public JTextArea newDayContent: This area, as the name proposes, gets the content from the user as a string.

public JRadioButton BDay, Happy,Sad: This buttons let the user to specify the day’s type. “Bday” stands for birthdays, “Happy” stands for happy days and “Sad” stands for sad days.

public JButton createdDay: This button finalizes the creation process of a new day and saves the changes on the day. Then, it updates the user’s information with some I/O operations in runtime.

public JTextField aDayTitle: This textfield shows the title of the day that is being shown.

public JTextArea aDayContent: This textarea shows the content of the day being shown in the frame.

public JButton aDayDelete: This button allows the user to delete the day being shown in the frame.

Constructor:

public YourDays: Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

private void oldDaysListValueChanged: This function gets the selected item from the list and via an I/O operation gets the content of the selected day from current user's file. Then, opens a frame to display the content of the day selected.

P.S: There are MouseClickListeners for all of the components above; however, as the attributes are explained clearly, we decided not to repeat what they do.

Albums Class:

Attributes:

public JLabel exitBut: This label works as a button for logging out from the system and common for all frames. That is, all frames has the label as a button.

public JLabel bcghomeLabel: This label holds the background image of the current frame.

public JButton albumsBut: This button is to open up the corresponding frame implied by the button name, in this case “Albums”.

public JLabel settingsBut: This label is to open up the settings section and let users do changes as they wish.

public JButton newEntryBut: This button works for opening up the corresponding frame implied by the button name, in this case “New Entry”. When the user wants to write a new entry, s/he must use this button to be able to open the frame s/he can write it.

public JButton notesBut: This button is to open up the corresponding frame implied by the button name, in this case “Notes”.

public JButton nyxBut: This button is to open up the corresponding frame implied by the button name, in this case, our chatbot “Talk to Nyx”.

public JButton oldEntriesBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Entries”.

public JButton oldTalksBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Talks”.

public JButton yourDaysBut: This button is to open up the corresponding frame implied by the button name, in this case “Days”.

private JLabel propic: This label holds the profile picture as the name proposes.

Constructor:

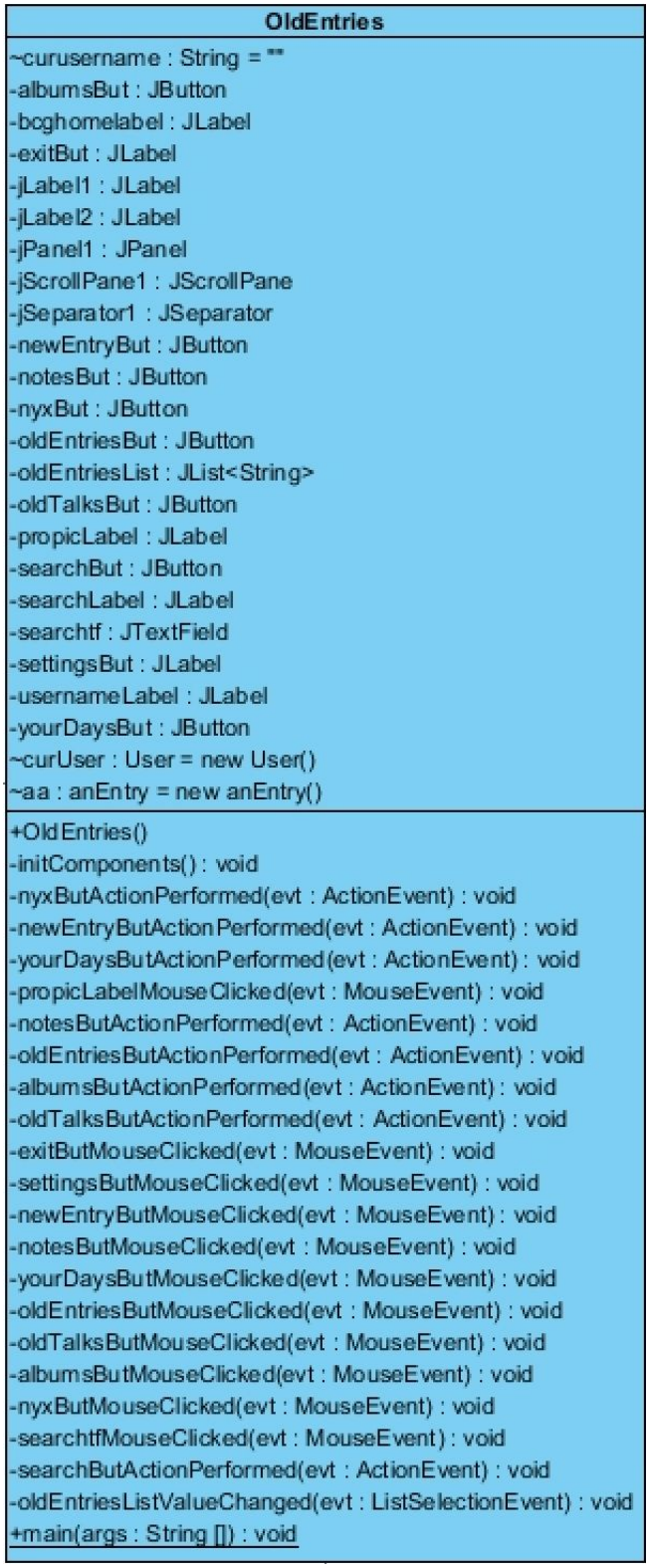
public Albums(): Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

private void propicMouseClicked: This method works for opening the window that is specific to the corresponding operation, in this case updating the profile picture.

private void notesButMouseClicked, oldEntriesButMouseClicked, newEntryButMouseClicked, yourDaysButMouseClicked, albumsButMouseClicked, oldTalksButMouseClicked, nyxButMouseClicked, settingsButMouseClicked, exitButMouseClicked: These functions simply change the frame to the corresponding frames proposed by the names. These functions are common for all of the frames except Login and Sign Up frames.

Old Entries class:



Attributes:

public String curusername: This string holds the name of the user that logged in.

public JLabel bcghomeLabel: This label holds the background image of the current frame.

public JLabel exitBut: This label works as a button for logging out from the system and common for all frames. That is, all frames has the label as a button.

public JButton newEntryBut: This button works for opening up the corresponding frame implied by the button name, in this case “New Entry”. When the user wants to write a new entry, s/he must use this button to be able to open the frame s/he can write it.

public JButton notesBut: This button is to open up the corresponding frame implied by the button name, in this case “Notes”.

public JButton nyxBut: This button is to open up the corresponding frame implied by the button name, in this case, our chatbot “Talk to Nyx”.

public JButton oldEntriesBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Entries”.

public JButton oldTalksBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Talks”.

public JButton yourDaysBut: This button is to open up the corresponding frame implied by the button name, in this case “Days”.

private JLabel propicLabel: This label holds the profile picture as the name proposes.

private User curUser: This User object collects all the data from methods about the user that logged in.

public JList oldEntriesList: This list shows all the entries created before on a label. The list items are clickable.

Constructor:

public OldEntries(): Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

oldEntriesListValueChanged: This function opens a frame which shows the selected entriy's information.

private void notesButMouseClicked, oldEntriesButMouseClicked, newEntryButMouseClicked, yourDaysButMouseClicked, albumsButMouseClicked, oldTalksButMouseClicked, nyxButMouseClicked, settingsButMouseClicked, exitButMouseClicked: These functions simply change the frame to the corresponding frames proposed by the names. These functions are common for all of the frames except Login and Sign Up frames.

Settings Class:

Attributes:

public JButton albumsBut: This button is to open up the corresponding frame implied by the button name, in this case “Albums”.

public JLabel bcghomeLabel: This label holds the background image of the current frame.

public JLabel exitBut: This label works as a button for logging out from the system and common for all frames. That is, all frames has the label as a button.

public JLabel settingsBut: This label is to open up the settings section and let users do changes as they wish.

public JButton newEntryBut: This button works for opening up the corresponding frame implied by the button name, in this case “New Entry”. When the user wants to write a new entry, s/he must use this button to be able to open the frame s/he can write it.

public JButton notesBut: This button is to open up the corresponding frame implied by the button name, in this case “Notes”.

public JButton nyxBut: This button is to open up the corresponding frame implied by the button name, in this case, our chatbot “Talk to Nyx”.

public JButton oldEntriesBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Entries”.

public JButton oldTalksBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Talks”.

public JButton yourDaysBut: This button is to open up the corresponding frame implied by the button name, in this case “Days”.

private JLabel propic: This label holds the profile picture as the name proposes.

Constructor:

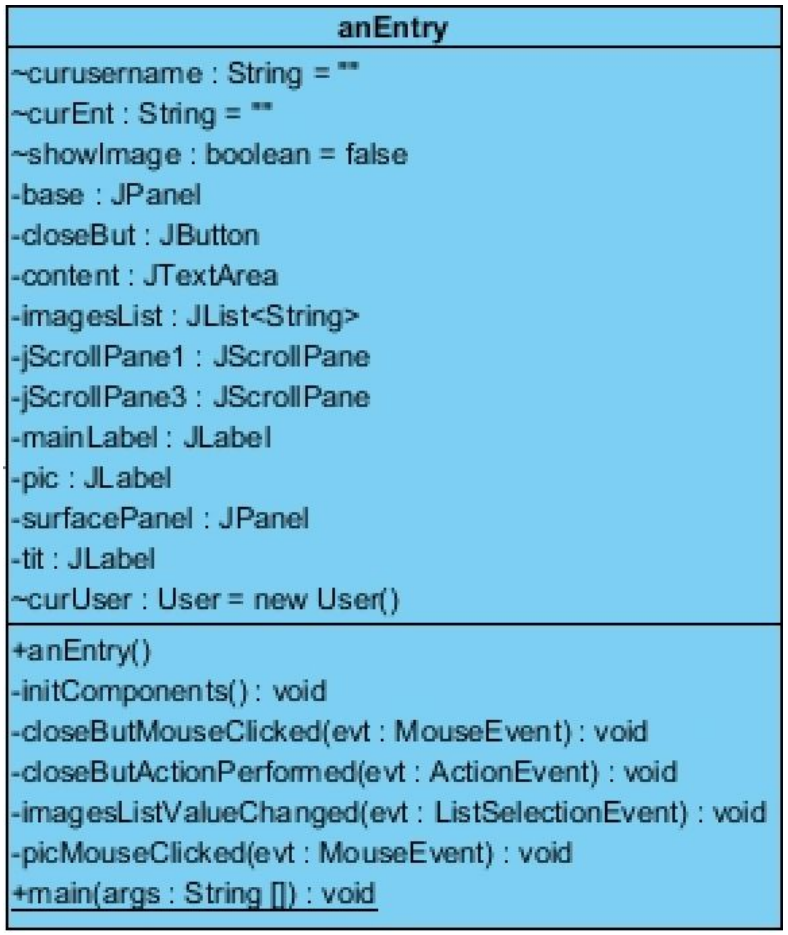
public Settings(): Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

private void propicMouseClicked: This method works for opening the window that is specific to the corresponding operation, in this case updating the profile picture.

private void notesButMouseClicked, oldEntriesButMouseClicked, newEntryButMouseClicked, yourDaysButMouseClicked, albumsButMouseClicked, oldTalksButMouseClicked, nyxButMouseClicked, settingsButMouseClicked, exitButMouseClicked: These functions simply change the frame to the corresponding frames proposed by the names. These functions are common for all of the frames except Login and Sign Up frames.

anEntry Class:



Attributes:

public String curusernane: This string holds the name of the user that logged in.

private User curUser: This User object collects all the data from methods about the user that logged in.

public boolean showImage: This boolean keeps track of an image's being shown or not. That is, when the user chooses an image to see in a bigger frame, showImage becomes true; otherwise stays false.

public JList imageList: This list shows the images that was added by the user when s/he created the entry.

public JTextArea content: This text area shows the content of the entry as a string in a fancy window.

public JLabel tit: This label shows the title of the entry chosen.

public JLabel pic: This label shows the chosen image by clicking on the corresponding image on imageList bigger.

public Entry curEnt: This Entry object holds the information of the entry currently being shown.

public JButton closeBut: This button simply stans to close the window of the displayed entry.

Constructor:

public anEntry: Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

private void imageListValueChanged: This function gets the value of the selected image and opens it up on a frame called pic.

OldTalks Class:

Attributes:

public JButton albumsBut: This button is to open up the corresponding frame implied by the button name, in this case “Albums”.

public JLabel bcghomeLabel: This label holds the background image of the current frame.

public JLabel exitBut: This label works as a button for logging out from the system and common for all frames. That is, all frames has the label as a button.

public JLabel settingsBut: This label is to open up the settings section and let users do changes as they wish.

public JButton newEntryBut: This button works for opening up the corresponding frame implied by the button name, in this case “New Entry”. When the user wants to write a new entry, s/he must use this button to be able to open the frame s/he can write it.

public JButton notesBut: This button is to open up the corresponding frame implied by the button name, in this case “Notes”.

public JButton nyxBut: This button is to open up the corresponding frame implied by the button name, in this case, our chatbot “Talk to Nyx”.

public JButton oldEntriesBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Entries”.

public JButton oldTalksBut: This button is to open up the corresponding frame implied by the button name, in this case “Old Talks”.

public JButton yourDaysBut: This button is to open up the corresponding frame implied by the button name, in this case “Days”.

private JLabel propic: This label holds the profile picture as the name proposes.

Constructor:

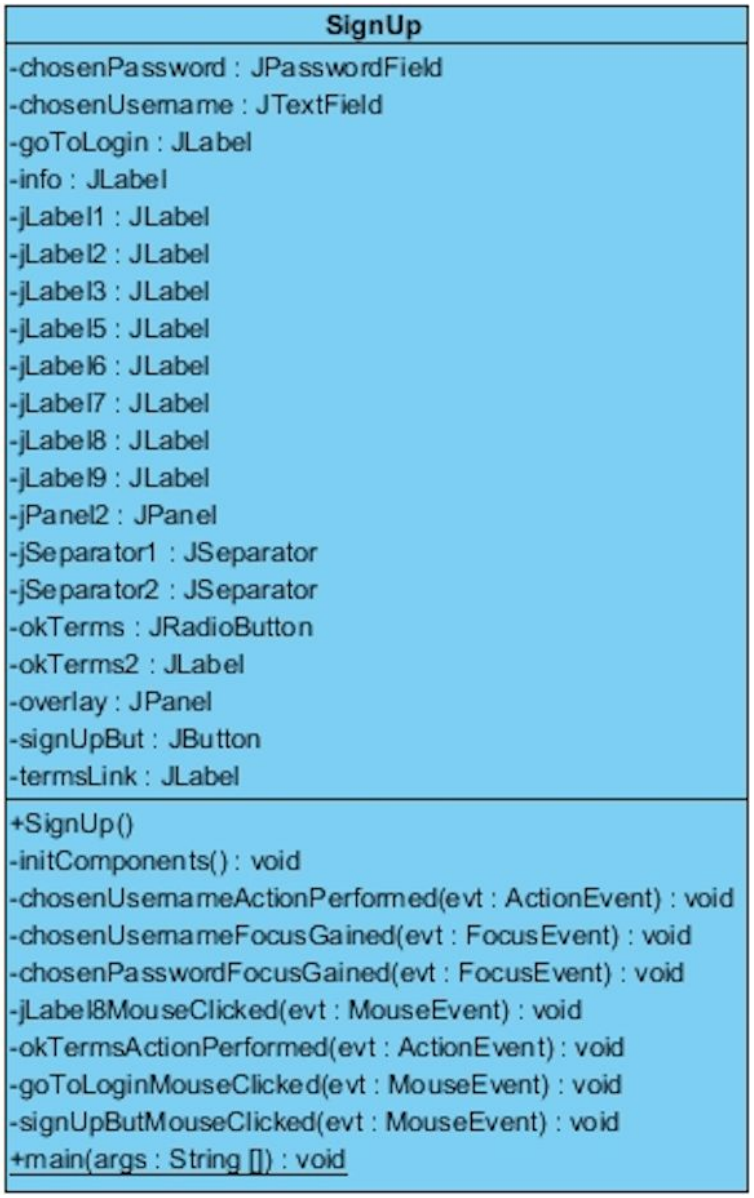
public OldTalks: Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

private void propicMouseClicked: This method works for opening the window that is specific to the corresponding operation, in this case updating the profile picture.

private void notesButMouseClicked, oldEntriesButMouseClicked, newEntryButMouseClicked, yourDaysButMouseClicked, albumsButMouseClicked, oldTalksButMouseClicked, nyxButMouseClicked, settingsButMouseClicked, exitButMouseClicked: These functions simply change the frame to the corresponding frames proposed by the names. These functions are common for all of the frames except Login and Sign Up frames.

SignUp Class:



Attributes:

private JPasswordField chosenPassword: This password field is for the user to enter the password s/he choose to use later when s/he wants to sign in the system.

private JTextField chosenUsername: This text field is for the users to enter the username they wish to use for the later log in operations to the system.

private JLabel goToLogin: This label is to open up the log in frame.

private JLabel info: This label is to inform the users about how safe the application is.

private JRadioButton okTerms: This radio button is used for the validation to accept the terms of service by the user.

private JLabel termsLink: This label is for the extended information about the terms of service that is prepared for the users to read.

Constructor:

public SignUp: Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

There are MouseClickListeners for all of the components above; however, as the attributes are explained clearly, we decided not to repeat what they do.

Login:

Attributes:

public JTextField txt_username: This area gets the user's username for login operation.

public JPasswordField txt_password: This area gets the password of the user for login operation:

public JButton loginBut: This button applies the login operation or fails according the information the user inputs.

public JLabel goToSignUpBut: This button is for the people who does not have an account and tries to use the program.

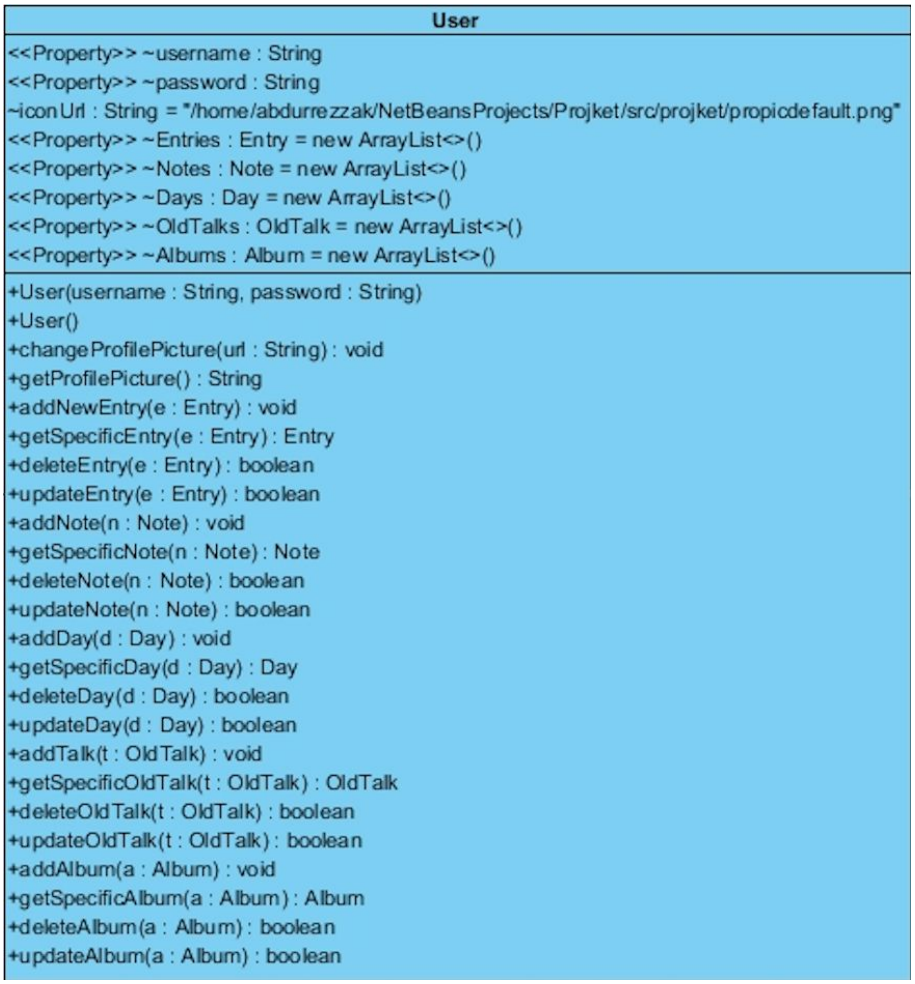
Constructor:

public Login: Initializes everything with default structures. Background colors, pictures etc. Additionally, it gets the attributes of the logged in user with file I/O operations to initialize corresponding parts.

Methods:

There are MouseClickListeners for all of the components above; however, as the attributes are explained clearly, we decided not to repeat what they do.

User Class:



Attributes

private String username: This simply stores the username of the user.

private String password: This is the password of a user.

public String iconUrl: This string stores the address of the image for the profile picture of the user.

public ArrayList<Day> Days: This arraylist stores the days of the user.

public ArrayList<Note> Notes: This arraylist stores the notes of the user.

public ArrayList<Entry> Entries: This arraylist stores the entries of the user.

public ArrayList<Album> Albums: This arraylist stores the albums of the user.

public ArrayList<OldTalk> OldTalks: This arraylist stores the old talks of the user with Nyx.

Constructors

public User(String u, String p): This constructor takes the username and password as parameters and create an User object.

public User(): This constructor is just for convention to create default user objects.

Methods

public void changeProfilePicture(): This function takes an url as a parameter and sets iconUrl attribute to it.

public String getProfilePicture(): This function returns the iconUrl.

public void addNewEntry(Entry e): This function takes an Entry as a parameter and adds to the Entries arraylist.

public Entry getSpecificEntry(Entry e): This function takes an Entry as a parameter and searches for it inside Entries arraylist.

public boolean deleteEntry(Entry e): This function takes an Entry as a parameter and deletes it inside Entries arraylist if possible.

public boolean updateEntry(Entry e): This function takes an Entry as a parameter and updates it inside Entries arraylist if possible.

public void addNote(Note n): This function takes a Note as a parameter and adds to the Notes arraylist.

public Note getSpecificNote(Note n): This function takes a Note as a parameter and searches for it inside Notes arraylist then returns it.

public boolean deleteNote(Note n): This function takes a Note as a parameter and deletes it inside Notes arraylist if possible.

public boolean updateNote(Note n): This function takes a Note as a parameter and updates it inside Notes arraylist if possible.

public void addDay(Day d): This function takes a Day as a parameter and adds to the Days arraylist.

public Day getSpecificDay(Day d): This function takes a Day as a parameter and searches for it inside Days arraylist then returns it.

public boolean deleteDay(Day d): This function takes a Day as a parameter and deletes it inside Days arraylist if possible.

public boolean updateDay(Day d): This function takes a Day as a parameter and updates it inside Days arraylist if possible.

public void addTalk(OldTalk t): This function takes an OldTalk object as a parameter and adds to the OldTalks arraylist.

public OldTalk getSpecificOldTalk(OldTalk t): This function takes an OldTalk as a parameter and searches for it inside OldTalks arraylist then returns it.

public boolean deleteOldTalk(OldTalk t): This function takes an OldTalk as a parameter and deletes it inside OldTalks arraylist if possible.

public boolean updateOldTalk(OldTalk t): This function takes an OldTalk as a parameter and updates it inside OldTalks arraylist if possible.

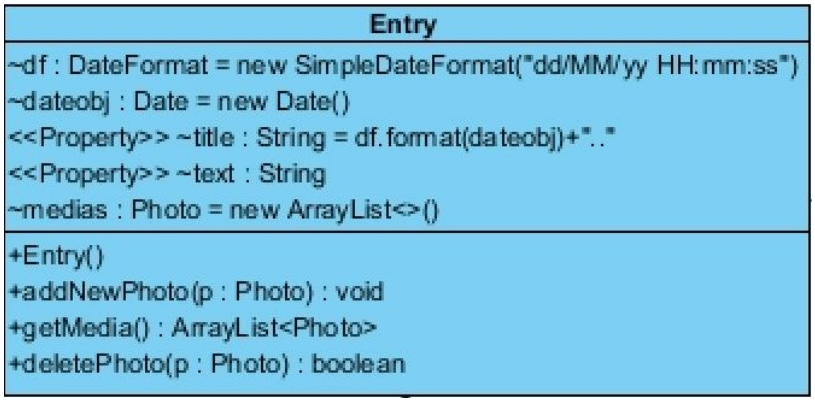
public void addAlbum(Album a): This function takes an Album as a parameter and adds to the Albums arraylist.

public Album getSpecificAlbum(Album a): This function takes an Album as a parameter and searches for it inside Albums arraylist then returns it.

public boolean deleteAlbum(Album a): This function takes an Album as a parameter and deletes it inside Albums arraylist if possible.

public boolean updateAlbum(Album a): This function takes an Album as a parameter and updates it inside Albums arraylist if possible.

Entry Class:



Attributes

public String title: This attribute stores the title of an Entry. If the user creates an entry with no title, its title is the date it is created.

public String text: This attribute is to store the content of the entry being created.

public ArrayList<Photo> medias: this arraylist stores the imaged added by the user while writing the entry.

Constructor

public Entry(): This constructor creates an empty Entry object.

Methods

public void setTitle(String s): This function sets the title of the entry.

public void setText(String s): This function sets the content of the entry.

public String getTitle: This function returns the title of the entry.

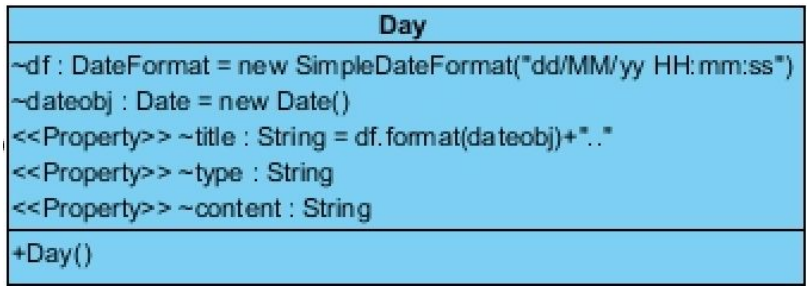
public String getText: This function returns the text of the entry.

public ArrayList<Photo> getMedia: This function returns the photos of the entry.

public void addNewPhoto(Photo p): This function adds new image to the entry.

public boolean deletePhoto(Photo p): This function deletes a specific image from the entry.

Day Class:



Attributes

public String title: This attribute stores the title of a Day. If the user creates a Day with no title, its title is the date it is created.

public String content: This attribute is to store the content of the Day being created.

public String type: This attribute is to store the type of the Day being created- Happy, Sad or Birthday.

Constructor

public Day(): This constructor creates an empty Day object.

Methods

public void setTitle(String s): This function sets the title of the day.

public void setContent(String s): This function sets the content of the day.

public String getTitle: This function returns the title of the day.

public String getContent: This function returns the text of the day.

public void setType(String s): This function sets the type of the day.

public String getType: This function returns the type of the day.

OldTalk Class:

Attributes

public String title: This attribute stores the title of an OldTalk. If the user creates an OldTalk with no title, its title is the date it is created.

public String content: This attribute is to store the content of the OldTalk being created.

public Date date: This attribute is to store the date of the OldTalk being created.

Constructor

public OldTalk(): This constructor creates an empty OldTalk object.

Methods

public void setTitle(String s): This function sets the title of the OldTalk.

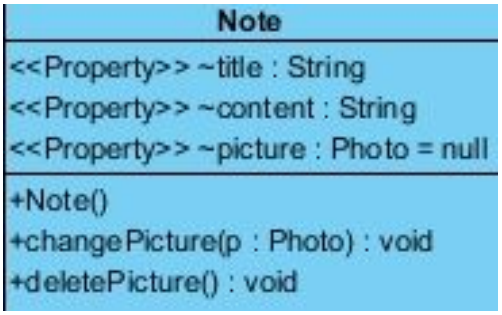
public void setContent(String s): This function sets the content of the OldTalk.

public String getTitle: This function returns the title of the OldTalk.

public String getContent: This function returns the text of the OldTalk.

public Date getDate: This function returns the date of the OldTalk.

Note Class:



Attributes

public String title: This attribute stores the title of a Note. If the user creates a Note with no title, its title is the date it is created.

public String content: This attribute is to store the content of the Note being created.

public Photo picture: This attribute is to store the image of the Note being created.

Constructor

public Note(): This constructor creates an empty Note object.

Methods

public void setTitle(String s): This function sets the title of the Note.

public void setContent(String s): This function sets the content of the Note.

public String getTitle: This function returns the title of the Note.

public String getContent: This function returns the text of the Note.

public void changeImage(Photo p): This function is to change the image of the Note.

public Photo getPicture: This function returns the picture of the Note.

public void deletePicture: This function simply deletes the image of the Note setting it to null.

Album Class:

Attributes

public String name: This attribute stores the name of an Album.

public Date date: This attribute is to store the date of the Album being created.

Constructor

public Album(): This constructor creates an empty Album object.

Methods

public void setName(String s): This function sets the name of the Album.

public String getName: This function returns the name of the Album.

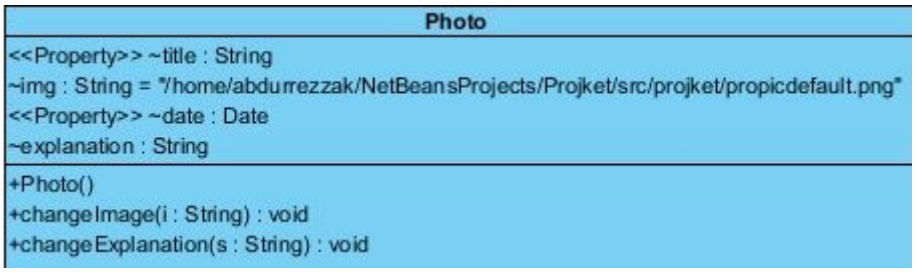
public Date getDate: This function returns the date of the Album.

public Arraylist<Photo> getPhotos: This function return the photos of the album as an arraylist.

public void addNewPhoto(Photo p): This function merely adds a new photo to the album.

public boolean deletePhoto(Photo p): This function simply deletes a photo from the arraylist if it exists.

Photo class:



Attributes

public String title: This attribute stores the title of a Photo. If the user creates a Photo with no title, its title is the date it is created.

public String explanation: This attribute is to store the explanation of the Photo being created.

public String img: This attribute is to store the url of the Photo being created.

public Date date: This attribute holds the value of the date of the photo.

Constructor

public Photo(): This constructor creates an empty Photo object.

Methods

public void setTitle(String s): This function sets the title of the Photo.

public String getTitle: This function returns the title of the Photo.

public Date getDate: This function returns the date of the Photo.

public void changeImage(String i): This function sets img to another url.

public void changeExplanation(String s): This function changes the explanation of the Photo.