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.

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.

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.

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.

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.
işte şey demek lazım the speed of the running application is compensated by the d extra memory usage.   ??????

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

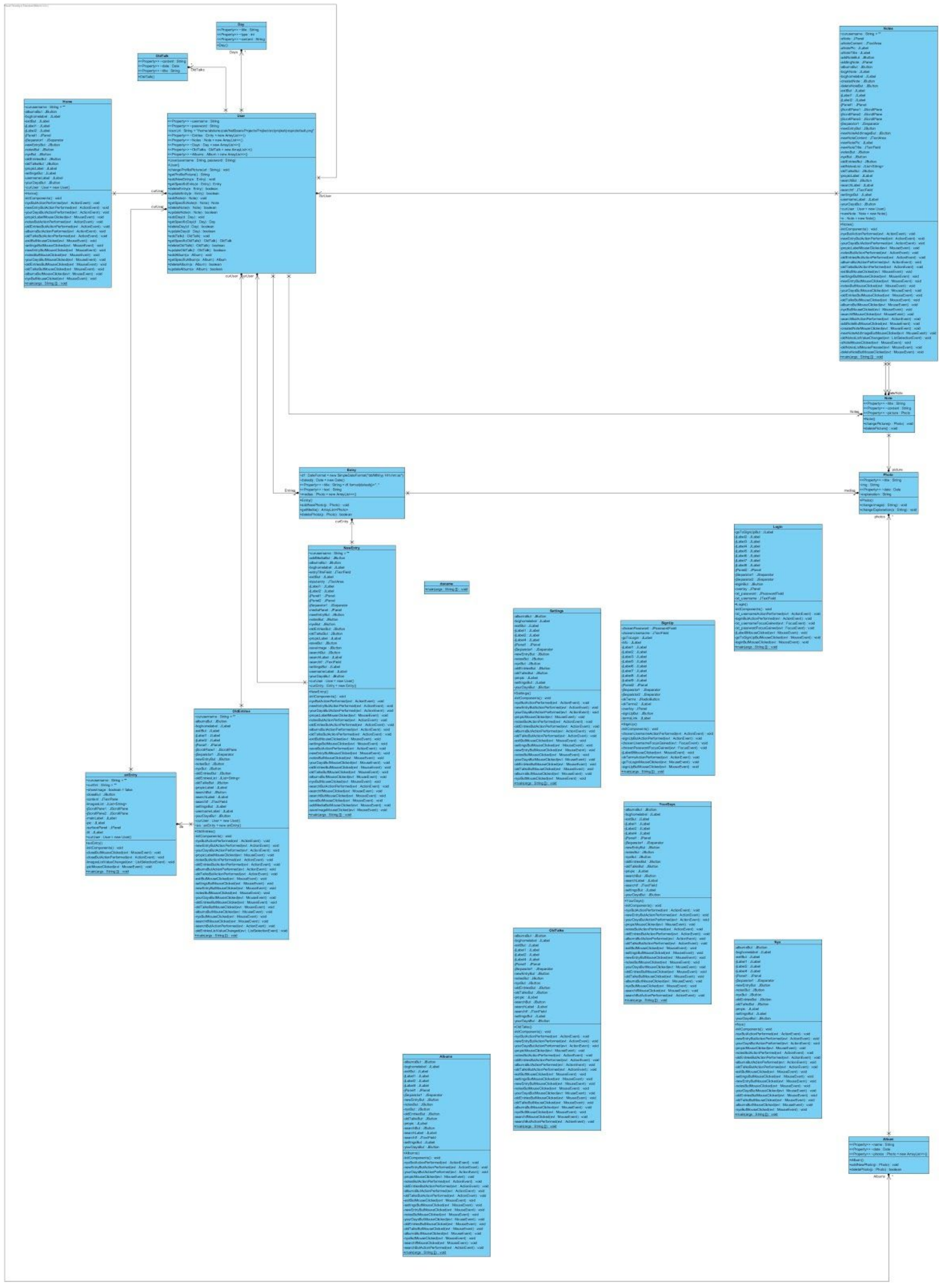
As the program is coded with Java, it handles exceptions very well. Whenever, a file is corrupted the system simply do not load the data stored in that file. On the other hand, although the program is robust enough to be run many times consecutively, cannot save data in some situations like non-responding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

3. Subsystems

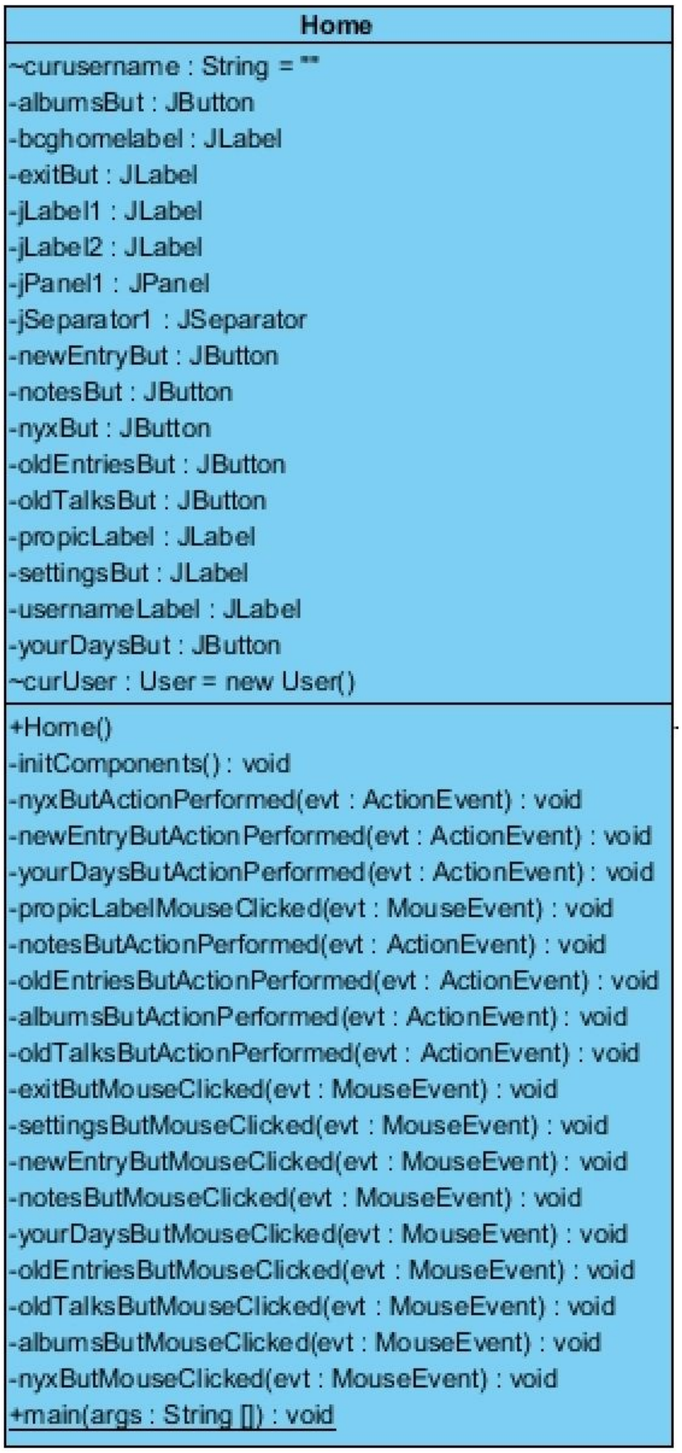
In this section we will provide the detailed information about the interfaces of our subsystems.

3.2. 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.

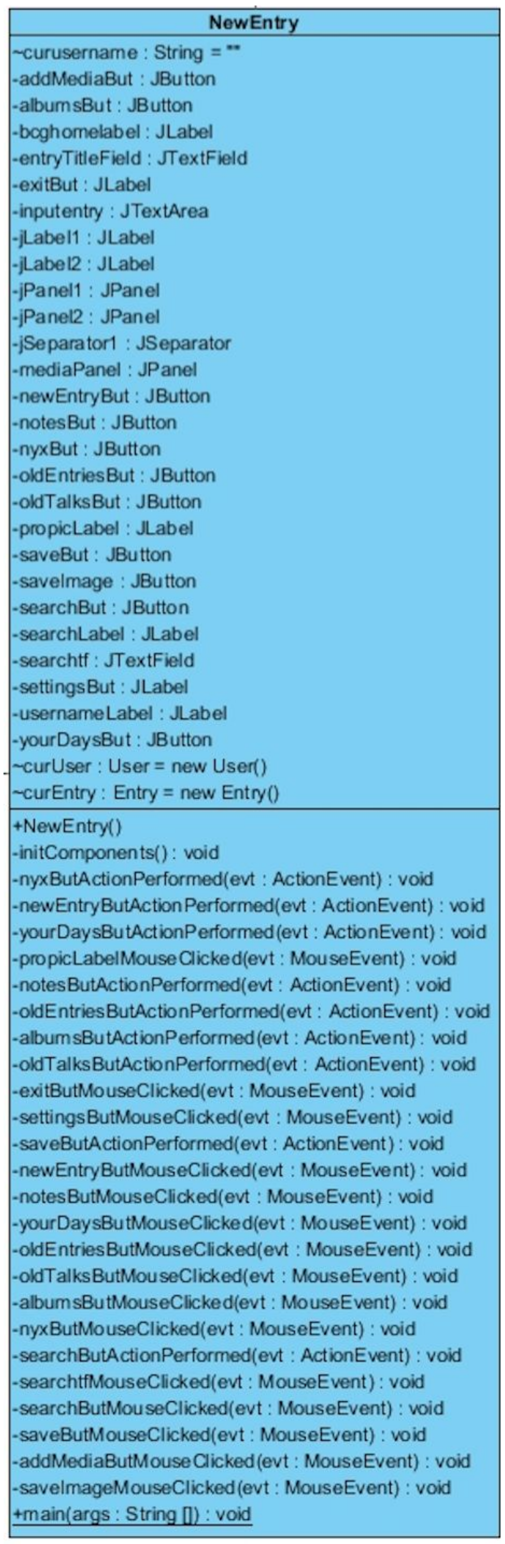
Methods:

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


NewEntry
~curusername : String = ""
-addMediaBut : JButton
-albumsBut : JButton
-bcghomelabel : JLabel
-entryTitleField : JTextField
-exitBut : JLabel
-inputentry : JTextArea
-jLabel1 : JLabel
-jLabel2 : JLabel
-jPanel1 : JPanel
-jPanel2 : JPanel
-jSeparator1 : JSeparator
-mediaPanel : JPanel
-newEntryBut : JButton
-notesBut : JButton
-nyxBut : JButton
-oldEntriesBut : JButton
-oldTalksBut : JButton
-propicLabel : JLabel
-saveBut : JButton
-saveImage : JButton
-searchBut : JButton
-searchLabel : JLabel
-searchtf : JTextField
-settingsBut : JLabel
-usernameLabel : JLabel
-yourDaysBut : JButton
~curUser : User = new User()
~curEntry : Entry = new Entry()
+NewEntry()
-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
-saveButActionPerformed(evt : ActionEvent) : 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
-searchButActionPerformed(evt : ActionEvent) : void
-searchtfMouseClicked(evt : MouseEvent) : void
-searchButMouseClicked(evt : MouseEvent) : void
-saveButMouseClicked(evt : MouseEvent) : void
-addMediaButMouseClicked(evt : MouseEvent) : void
-saveImageMouseClicked(evt : MouseEvent) : void
+main(args : String []) : void

Attributes:

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

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.

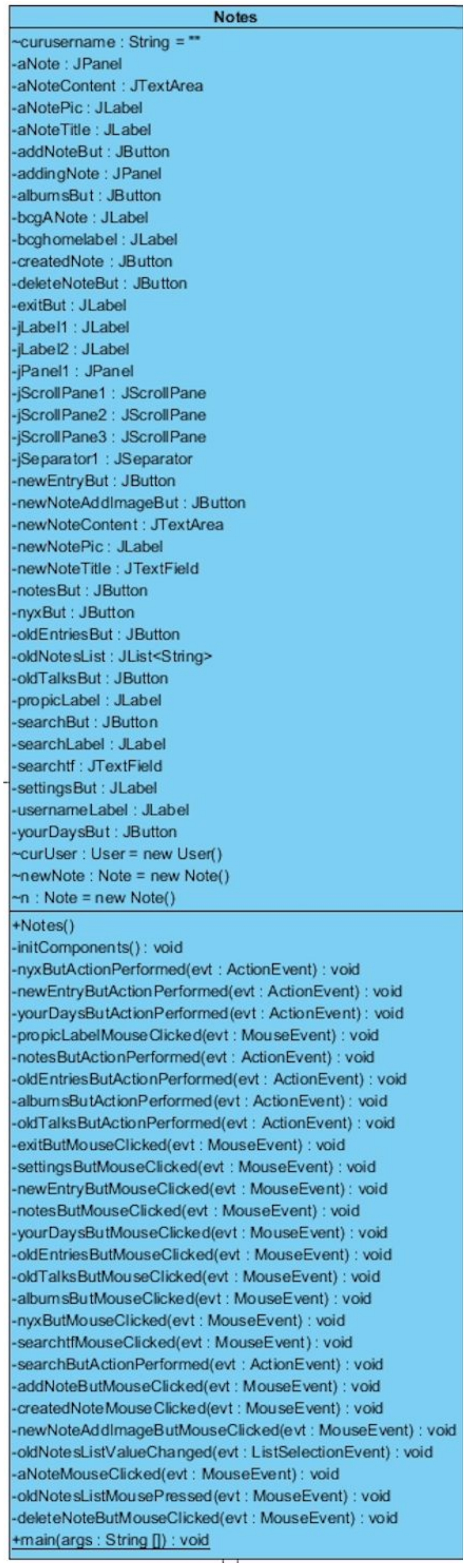
private void notesButMouseClicked, oldEntriesButMouseClicked, newEntryButMouseClicked, yourDaysButMouseClicked, albumsButMouseClicked, oldTalksButMouseClicked, nyxButMouseClicked, settingsButMouseClicked, exitButMouseClicked:
These functions simply change the frame to the corresponding frames proposed 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 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.

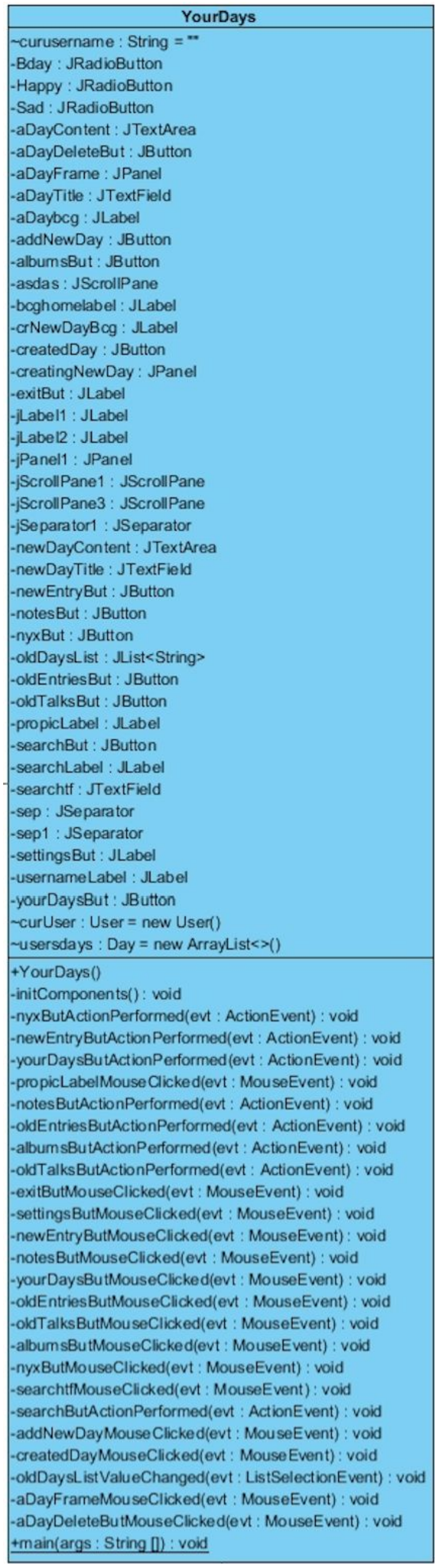
private void addNoteButMouseClicked: This function is to open a frame for the user to create a new note.

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:



```
YourDays
~curusername : String = ""
-Bday : JRadioButton
-Happy : JRadioButton
-Sad : JRadioButton
-aDayContent : JTextArea
-aDayDeleteBut : JButton
-aDayFrame : JPanel
-aDayTitle : JTextField
-aDaybcg : JLabel
-addNewDay : JButton
-albumsBut : JButton
-asdas : JScrollPane
-bcghomelabel : JLabel
-crNewDayBcg : JLabel
-createdDay : JButton
-creatingNewDay : JPanel
-exitBut : JLabel
-jLabel1 : JLabel
-jLabel2 : JLabel
-jPanel1 : JPanel
-jScrollPane1 : JScrollPane
-jScrollPane3 : JScrollPane
-jSeparator1 : JSeparator
-newDayContent : JTextArea
-newDayTitle : JTextField
-newEntryBut : JButton
-notesBut : JButton
-nyxBut : JButton
-oldDaysList : JList<String>
-oldEntriesBut : JButton
-oldTalksBut : JButton
-propicLabel : JLabel
-searchBut : JButton
-searchLabel : JLabel
-searchtf : JTextField
-sep : JSeparator
-sep1 : JSeparator
-settingsBut : JLabel
-usernameLabel : JLabel
-yourDaysBut : JButton
~curUser : User = new User()
~usersdays : Day = new ArrayList<>()
+YourDays()
-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
-addNewDayMouseClicked(evt : MouseEvent) : void
-createdDayMouseClicked(evt : MouseEvent) : void
-oldDaysListValueChanged(evt : ListSelectionEvent) : void
-aDayFrameMouseClicked(evt : MouseEvent) : void
-aDayDeleteButMouseClicked(evt : MouseEvent) : void
+main(args : String []) : void
```

Attributes:

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

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.

private