**Jeff Angius, Jeffrey Nguyen**

**Johnathon Chenvert, Arthur Hoang**

**CIS44 Project Report**

**TABLE OF CONTENTS (they are linked, so you can**

**index without having to scroll to search : hold ctrl and click on the link)**

[**Introduction**](#Introduction)

[**Body**](#Body)

[**Conclusion**](#Conclusion)

[**LoginWIndow.java**](#LoginWindow)

[**RegisterWindow.java**](#RegisterWindow)

[**searchWIndow.java**](#searchWindow)

[**toDoListWindow.java**](#toDoListWindow)

[**userWindow.java**](#userWindow)

[**EventWindow.java**](#EventWindow)

[**WelcomeWindow.java**](#WelcomeWindow)

[**addressPanel.java**](#addressPanel)

[**descriptionPanel.java**](#descriptionPanel)

[**datePanel.java**](#datePanel)

[**LinkedUserFile.java**](#LinkedUserFile)

[**CalendarScanner.java**](#CalendarScanner)

[**EventLinkedBag.java**](#EventLinkedBag)

[**Event.java**](#Event)

**Introduction**

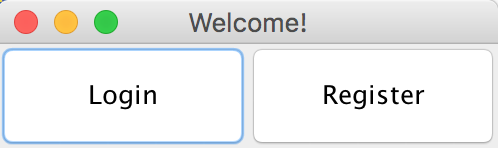
This team’s collaboration led to an idea of developing some sort of event-based to-do list application. We wanted to create something that could, not only connect to a certain date and/or time, but also allow the user to search through the events log for a certain event. In this inspiration, we began to heavily research in event-based programming, Java GUI modeling, and the amalgamation of these core parts to the data structures that we have learned in class, the heart of the program. In doing so, we learned a great deal in those capacities.

First, we began with the Java GUI model, endeavoring to understand the framing, the panels, the layouts, and all of the pertaining components. As we explored those interfaces, we also explored event programming, programming based on object that represents a user's interaction with a GUI component that can be "handled" to create interactive elements. Once the event occurs, it necessitated the listener, an object that waits for events and responds to them. For example, to handle an event, we attach a listener to a component, and the listener will be notified when the event occurs.

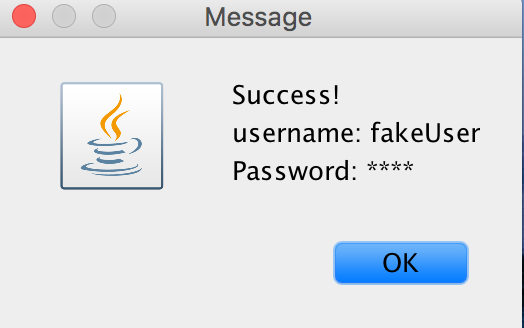
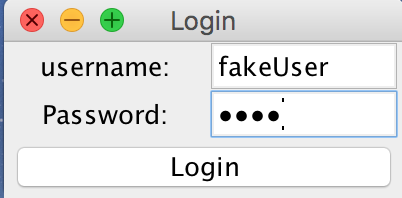
Most importantly, we examined our lessons to modify the data structure codes so that we could implement it into our program. The data structure we chose that best suited our needs was the Linked Bag system. In implementing it, we required a means of sorting through the user input data. For this task, we realized that the search algorithm best suited for our requirements was the Merge Sort process.

**Body**

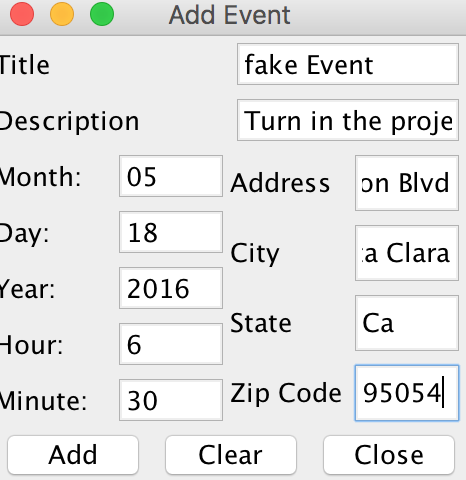
We built the application with the explicit focus on the user. The program initiates by displaying a window with two button options, Register or Login.



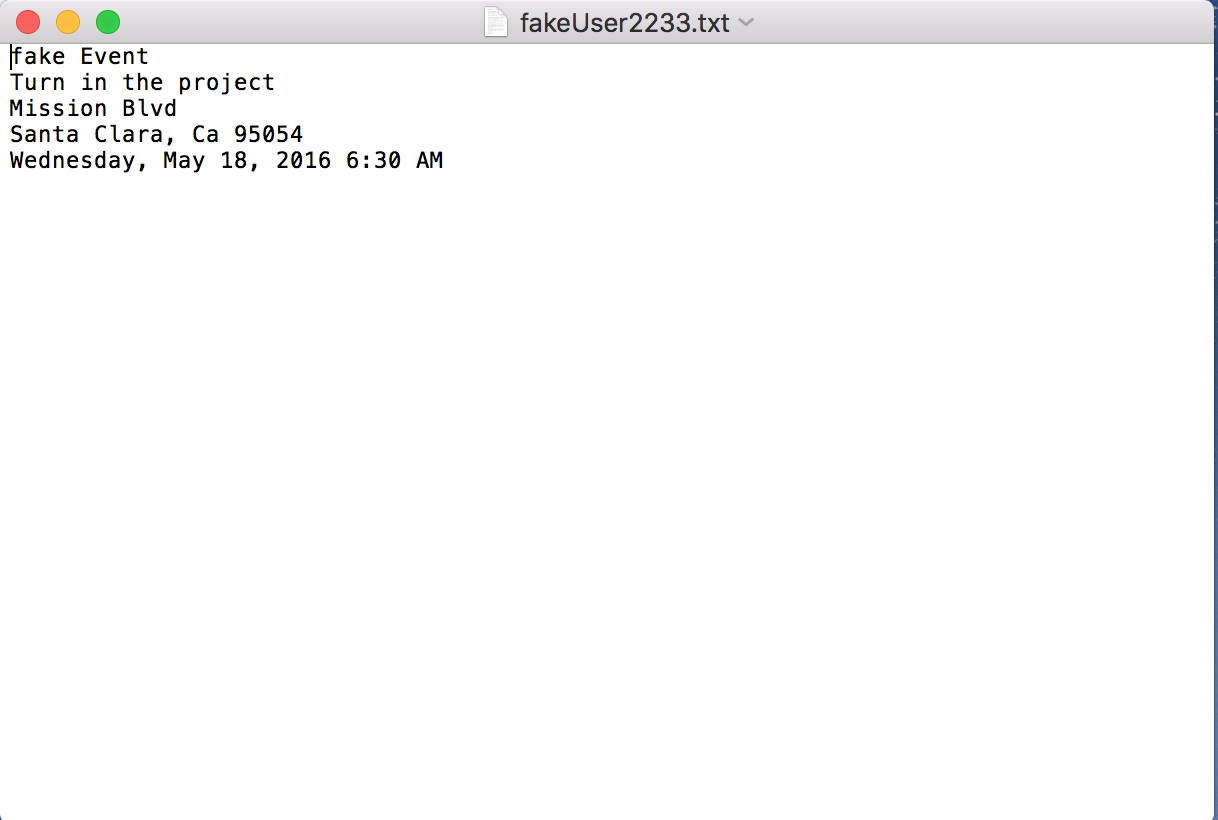
This window was formed with the Java GUI (Java graphical user interface). From the Swing libraries, we were able to create the button components Login and Register and place them in a JFrame. Once the components were settled into their containers, we used the ActionLIstener Interface to check whether an action or event (which could be a mouse click or the press of a button) has occurred. If it has, then ActionListener will accept it as an event variable. This variable is then checked in an if statement parameters to see if it is valid. If so, it then begins a try / catch condition. In the try block, three variables are created. Two of the variables are there to hold the user input of username and password, and the third variable holds the combination of the username and password and creates a .txt file of that. The catch block, checks whether or not the user has inputted valid information. If not, the catch block throws a warning.

The same code structure was constructed for the Login button. Only now we are not making a file, but scanning for it. If the user input is correct, the program would continue. If the user input was incorrect, it would trigger a pop-up window to the user stating the invalidity. Once the login was successful, the user is prompted with the window to create an event he or she wants recorded. 

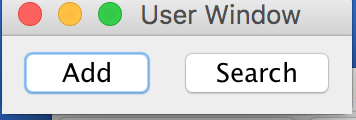
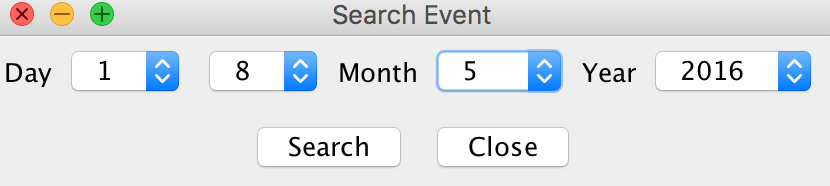
The Event Window allows the user to add unique events to his/her collection of Events. To add an event properly the user must type in all required information in the Event Window’s TextFields, and then click the “Add” button. When the “Add” button is clicked, an ActionEvent is fired to gather all the data in the EventWindow’s TextFields, store that data in a new Event object, and finally add that Event object to the user’s linkedBag. When the user wants to clear all of the EventWindow’s TextFields, he/she must click the Clear button. Lastly, when the user wants to print his/her linkedBag to their TextFile, the user must click the close button, which will fire an ActionEvent to print the entire linkedBag using the method LinkedUserFile.print().



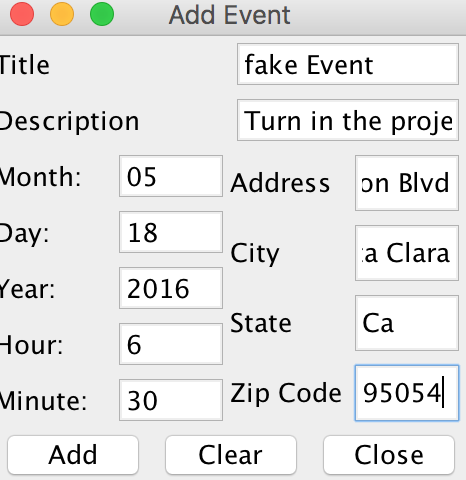
For the event class itself, we had variable types that are typical to an event: the title of the event, the description, the month, day, year, hour, minute, address, city, state, and zip. The class has typical set and get methods, though there are three methods that set it to be different: getMilitaryTime(), compareTo(), and equals() method. The equals() method allows us to be able to compare all the event dates within a given bag and see if they are of equal values. This helps since we have a method within LinkedBag that relies on this method (arrayTransfer). The getMilitaryTime() method returns to us the actual military time by concatenating both the hour and the minute integers. Our compareTo() method compares the military time of both event objects, and checks to see whether the military time is greater than or less than. Thus, the user has created their first .txt file storage of data.



Now, in order for the user to sift through its events on some specific dayand maybe add some more reminders, we needed to create a search and add window. The Search Window allows the user to search for specific dates in the user’s TextFile that holds his/her Events. To search for a date the user must select integers from the fields, “Day”, “Month”, “Year” and then click the button “Search”. The “Search” button fires an ActionEvent that gathers the integer values of “Date” fields, “Month” fields and “Year” fields. The ActionEvent then passes that information to the method, LinkedUserFile.arrayTransfer(int day, int month, int year). The arrayTransfer method searches through the user’s TextFile of events and finds every Event that matches the date that was passed to the arrayTransfer method. As the arrayTransfer method finds Events that match, it adds them to an array of Events and returns that array to the user\. Finally, the array of Events that the arrayTransfer method returns is passed to the method, LinkedUserFile.sort(Event[]) method. After the array of Events is sorted, the array is printed in the toDoListWindow().

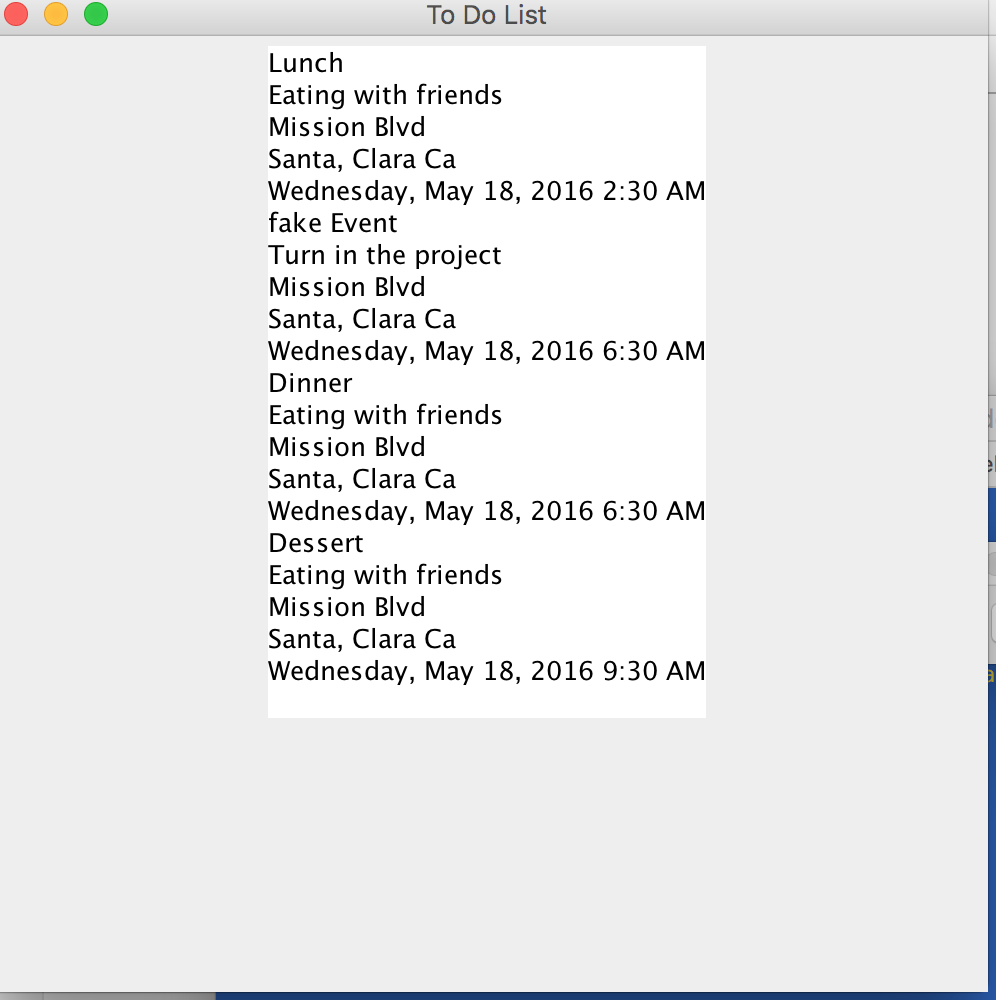


The Add window allow the person to add or search data that is being implemented to a LinkedBag which it converts into a text file using it’s print method. The add button fires an action event which opens the Event window, which allows the user to input their data. In this case, it would be the: Title (name of event), Description (describing the event), Month (adds current month/past/future events from 1 to 12), Day (from 1-31 days), Year (from 1976 to any near future), Street (your current valid address), City (any valid city), State (any state, 50 states), and ZipCode (a valid 5-digit number). This will be stored into a LinkedBag in a text file



            Within the LinkedBag class, everything is the same, except for the arrayTransfer() method. The arrayTransfer() method accepts parameters for month, day, and year, which is all given via user input. This then becomes a new event object that can be used to check and see if all events in the bag are equal to the user’s search for the newly created event object. If it is, then the event object in the bag gets written to an event array, ultimately returning the event array that has the same event dates as the search criteria given. In this fashion, we are able to search through the inputted dates of the given events.

            In trying to sort through the data, the compareTo() method was utilized in the event class to be able to sort with the Merge Sort method that we worked on earlier in the semester to be able to sort the event array. With the event array from arrayTransfer(), the array can be thrown to be sorted, and return completely sorted. We searched through the created To-Do List.



The toDoListWindow accepts an array of Events and prints them to a JTextArea. It is the window that allows the user to see the chronological order of events that correspond to a given date.

**Conclusion**

Hence, we have a number of plans in store for the future of this project. One of our intentions is to implement a remove button within the GUI interface, and have a remove method that consists with the button. This method should remove a specified method that the user requests to be removed. Another plan for the future of this project is to make the GUI look a lot better, and be ready for other platforms, such as Android and iOS. Before porting to a different platform, however, we need to have two things: have the option to edit the event in question should the user need to change something about the event, and the program should also be able to throw notifications to the user, letting them know that their event is coming up soon, or be a modifiable notification that the user can preset to send them a reminder when the event will take place. For the user, we also would like to show them when their event was created, and when the event was modified as well. Lastly, we would like to store the information that the user inputs into an LDAP server, and modify our program to search from the LDAP server. We will also work on securing our LDAP server to prevent people from hacking into our server. These are, essentially, our plans for future modification of our project before we have an official beta.

**//LoginWindow.java**

import javax.swing.\*;

import javax.swing.JPasswordField;

import java.awt.event.\*;

import java.awt.\*;

import java.util.\*;

import java.io.\*;

class LoginWindow extends JFrame

{

private JPanel m\_namePanel;

private JPanel m\_passwordPanel;

private JPanel m\_loginButtonPanel;

private JTextField m\_nameField;

private JPasswordField m\_passwordField;

private JLabel m\_nameLabel;

private JLabel m\_passwordLabel;

private JButton m\_loginButton;

private String m\_userName;

private String m\_password;

private JOptionPane m\_pane;

private JPanel m\_centerPanel;

/\*\*

constructor

\*/

public LoginWindow()

{

//set the title of the window to "login"

setTitle("Login");

//allow the window to close when the user clicks the exit button

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

//set up the layout for the loginWindow to have 3 rows and 1 column

setLayout(new BorderLayout());

//make the name panel

makeNamePanel();

//make the password panel

makePasswordPanel();

//make the loginButtonPanel

makeLoginButtonPanel();

//makeCenterPanel

makeCenterPanel();

//add all the panels to the middle of the loginWindow

add(m\_centerPanel, BorderLayout.CENTER);

add(m\_loginButtonPanel, BorderLayout.SOUTH);

//set the size of the window

setSize(200, 100);

//set up where the window will pop up on the user's screen

setLocationRelativeTo(null);

//allow the window to be visible

setVisible(true);

}

/\*\*

makeNamePanel method makes the panel that asks the user to enter his/her name

\*/

public void makeNamePanel()

{

//make the jpanel for the username portion of the login window

m\_namePanel = new JPanel();

//set the layout of the namePanel to gridLayout with 1 row and 2 columns

m\_namePanel.setLayout(new GridLayout(1, 2));

//create a label that properly labels the text field that will hold usernames

m\_nameLabel = new JLabel("username:", SwingConstants.CENTER);

//initialize the nameField to allow users to type their names

m\_nameField = new JTextField();

//allow the name field to be edited

m\_nameField.setEditable(true);

//add the label and the text field to the namePanel

m\_namePanel.add(m\_nameLabel);

m\_namePanel.add(m\_nameField);

}

/\*\*

makePasswordPanel method makes the panel that asks the user for his/her password

\*/

public void makePasswordPanel()

{

//make the password jpanel

m\_passwordPanel = new JPanel();

//set the layout of the passwordPanel to 1 row and 2 columns

m\_passwordPanel.setLayout(new GridLayout(1, 2));

//make the password label that properly labels the textfield that holds passwords

m\_passwordLabel = new JLabel("Password:", SwingConstants.CENTER);

//make the password field to hold passwords

m\_passwordField = new JPasswordField();

//allows the passwordfield to be editable

m\_passwordField.setEditable(true);

//add the passwordlabel and the password field to the passwordPanel

m\_passwordPanel.add(m\_passwordLabel);

m\_passwordPanel.add(m\_passwordField);

}

/\*\*

makeCenterPanel method adds the password panel and username panel to the centerPanel

\*/

public void makeCenterPanel()

{

m\_centerPanel = new JPanel();

m\_centerPanel.setLayout(new GridLayout(2, 1));

m\_centerPanel.add(m\_namePanel);

m\_centerPanel.add(m\_passwordPanel);

}

/\*\*

makeLoginButtonPanel method makes the panel that holds the login button that the user clicks on

\*/

public void makeLoginButtonPanel()

{

//make the jpanel that will hold the login button

m\_loginButtonPanel = new JPanel();

//set the layout to grid layout with 1 row and now columns

m\_loginButtonPanel.setLayout(new GridLayout(1, 0));

//set the title of the login button

m\_loginButton = new JButton("Login");

//add the loginListener to the loginButton

m\_loginButton.addActionListener(new loginListener());

//add the loginButton to the loginButtonPanel

m\_loginButtonPanel.add(m\_loginButton);

}

/\*\*

loginListener class allows the login button to work correctly

\*/

private class loginListener implements ActionListener

{

/\*\*

actionPerformed deals with the mouseClick on the login button appropriately

\*/

public void actionPerformed(ActionEvent event)

{

if(event.getSource() == m\_loginButton)

{

//retrieve the text in the username and the password fields

m\_userName = m\_nameField.getText();

m\_password = m\_passwordField.getText();

String filename = m\_userName + m\_password + ".txt";

try

{

makeFile(filename);

new userWindow(filename);

}

catch(IOException exception)

{

m\_pane.showMessageDialog(null, "Please enter a valid username and password");

}

}

}

/\*\*

makeFile method takes the username and makes a textFile entitled the username

@param username

\*/

public void makeFile(String filename) throws IOException

{

Scanner myFile = new Scanner(new File(filename));

}

}

/\*\*

main method

\*/

public static void main(String[] args)

{

LoginWindow myWindow = new LoginWindow();

}

}

**//RegisterWindow.java**

/\*\*

\*The RegisterWindow is a window that appears in the CalendarProgram.

\*It allows the user to set up a textfile that corresponds with his/her username.

\*

\*

\*@author Jeff Angius

\*@version 1.0

\*@since 5-16-2016

\*/

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import java.util.\*;

import java.io.\*;

import java.lang.String;

public class RegisterWindow extends JFrame

{

private JPanel m\_namePanel;

private JLabel m\_firstNameLabel;

private JLabel m\_lastNameLabel;

private JTextField m\_firstNameField;

private JTextField m\_lastNameField;

private JPanel m\_userPasswordPanel;

private JLabel m\_userNameLabel;

private JLabel m\_passwordLabel;

private JTextField m\_userNameField;

private JPasswordField m\_passwordField;

private JPanel m\_registerButtonPanel;

private JButton m\_registerButton;

private JPanel m\_centerPanel;

private String m\_password;

private String m\_userName;

private JOptionPane m\_pane;

/\*\*

constructor of the main window that asks the user

\*/

public RegisterWindow()

{

//set the title of the window to "login"

setTitle("Register");

//allow the window to close when the user clicks the exit button

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

//set up the layout for the loginWindow to have 3 rows and 1 column

setLayout(new BorderLayout());

//make the name panel

makeNamePanel();

//make the password panel

makeUserPasswordPanel();

//make main panel

makeCenterPanel();

//make the registerButtonPanel

makeRegisterButtonPanel();

//add all the panels to the register Window

add(m\_centerPanel, BorderLayout.CENTER);

add(m\_registerButtonPanel, BorderLayout.SOUTH);

//set the size of the window

setSize(300, 150);

//set up where the window will pop up on the user's screen

setLocationRelativeTo(null);

//allow the window to be visible

setVisible(true);

}

/\*\*

makeNamePanel method makes the panel that holds the labels, "first name", "last name", "username", "password", as well as the textfields that correspond with them

\*/

public void makeNamePanel()

{

//make the jpanel for the name portion of the register window

m\_namePanel = new JPanel();

//set the layout of the namePanel to gridLayout with 1 row and 2 columns

m\_namePanel.setLayout(new GridLayout(2, 2));

//create labels that properly labels the text field that will hold first and last names

m\_firstNameLabel = new JLabel("First Name:", SwingConstants.CENTER);

m\_lastNameLabel = new JLabel("Last Name:", SwingConstants.CENTER);

//initialize the m\_firstNameField and the m\_lastNameField to allow users to type their names

m\_firstNameField = new JTextField();

m\_lastNameField = new JTextField();

//allow the name field to be edited

m\_firstNameField.setEditable(true);

m\_lastNameField.setEditable(true);

//add the label and the text field to the namePanel

m\_namePanel.add(m\_firstNameLabel);

m\_namePanel.add(m\_firstNameField);

m\_namePanel.add(m\_lastNameLabel);

m\_namePanel.add(m\_lastNameField);

}

/\*\*

makeUserPasswordPanel makes a panel that holds the labels, "username" and "password" as well as one textfield and one password field that corresponds with those labels

\*/

public void makeUserPasswordPanel()

{

//make the password pannel

m\_userPasswordPanel = new JPanel();

//set the layout of the password panel

m\_userPasswordPanel.setLayout(new GridLayout(2,2));

//make the username label and the username field

m\_userNameLabel = new JLabel("username:", SwingConstants.CENTER);

m\_userNameField = new JTextField();

m\_userNameField.setEditable(true);

//make the password field and the password label

m\_passwordLabel = new JLabel("Password:", SwingConstants.CENTER);

m\_passwordField = new JPasswordField();

m\_passwordField.setEditable(true);

//add the components to the password panel

m\_userPasswordPanel.add(m\_userNameLabel);

m\_userPasswordPanel.add(m\_userNameField);

m\_userPasswordPanel.add(m\_passwordLabel);

m\_userPasswordPanel.add(m\_passwordField);

}

/\*\*

makeCenterPanel adds the name panel and the password panel to one panel called, m\_centerPanel

\*/

public void makeCenterPanel()

{

m\_centerPanel = new JPanel();

m\_centerPanel.setLayout(new GridLayout(2, 1));

m\_centerPanel.add(m\_namePanel);

m\_centerPanel.add(m\_userPasswordPanel);

}

/\*\*

makeRegisterButtonPanel method makes the panel that holds the register button that the user clicks on

\*/

public void makeRegisterButtonPanel()

{

//make the jpanel that will hold the login button

m\_registerButtonPanel = new JPanel();

//set the layout to grid layout with 1 row and now columns

m\_registerButtonPanel.setLayout(new GridLayout(1, 0));

//set the title of the login button

m\_registerButton = new JButton("Register");

//add the loginListener to the loginButton

m\_registerButton.addActionListener(new RegisterListener());

//add the loginButton to the loginButtonPanel

m\_registerButtonPanel.add(m\_registerButton);

}

/\*\*

closeWindow method closes the jframe without ending the program

\*/

public void closeWindow()

{

this.dispose();

}

private class RegisterListener implements ActionListener

{

/\*\*

actionPerformed method allows for the action event of clicking on the registerbutton. once it is clicked, a file is made to store the user's information

\*/

public void actionPerformed(ActionEvent event)

{

if(event.getSource() == m\_registerButton)

{

m\_userName = m\_userNameField.getText();

String fakePassword = getEncryptedPassword(m\_passwordField);

m\_password = m\_passwordField.getText();

String filename = m\_userName + m\_password +".txt";

try

{

makeFile(filename);

m\_pane = new JOptionPane();

m\_pane.showMessageDialog(null, "Success!\nusername: " + m\_userName + "\n" + "Password: " + fakePassword);

closeWindow();

}

catch(IOException exception)

{

m\_pane.showMessageDialog(null, "Please enter a valid username and password");

}

}

}

/\*\*

makeFile method makes a text file that is entitled the username and password.txt

@param filename the name of the file that the user wants to register under

@throws IOException

\*/

public void makeFile(String filename)throws IOException

{

FileWriter myFile = new FileWriter(filename, true);

}

/\*\*

getEncryptedPassword returns a string object of symbols instead of the actual password

@param password the user's password represented as a PasswordField object

@return string the string of symbols

\*/

public String getEncryptedPassword(JPasswordField password)

{

char[] symbols = password.getPassword();

char star = '\*';

String newPassword = "";

for(int count = 0; count < symbols.length; count++)

{

newPassword += String.valueOf(star);

}

return newPassword;

}

}

/\*\*

main method

\*/

public static void main(String[] args)

{

new RegisterWindow();

}

}

**//searchWindow.java**

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import java.util.\*;

import java.io.\*;

public class searchWindow extends JFrame

{

private JComboBox <Integer> m\_dayBox1;

private JComboBox <Integer> m\_dayBox2;

private JComboBox <Integer> m\_monthBox;

private JComboBox <Integer> m\_yearBox;

private LinkedUserFile m\_file;

private JLabel m\_dayLabel;

private JLabel m\_monthLabel;

private JLabel m\_yearLabel;

private JPanel m\_centerPanel;

private JPanel m\_southPanel;

private JButton m\_searchButton;

private JButton m\_closeButton;

private Integer[] days = {0,1,2,3};

private Integer[] days2 = {0,1,2,3,4,5,6,7,8,9};

private Integer[] months = {0,1,2,3,4,5,6,7,8,9,10,11,12};

private Integer[] years = {2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024};

/\*\*

main

\*/

public static void main(String[] args)throws IOException

{

new searchWindow("jMoney2233.txt");

}

/\*\*

constructor

\*/

public searchWindow(String filename)throws IOException

{

//make the text file that user will add events to

m\_file = new LinkedUserFile(filename);

//set the title

setTitle("Search Event");

//set the close operation

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

//set the layout for the window

setLayout(new BorderLayout());

//make the center panel

makeCenterPanel();

//make south panel

makeSouthPanel();

//add the components

add(m\_centerPanel, BorderLayout.CENTER);

add(m\_southPanel, BorderLayout.SOUTH);

//set up where the window will pop up on the screen

setLocationRelativeTo(null);

//make the window visible

setVisible(true);

pack();

}

/\*\*

makeCenterPanel

\*/

public void makeCenterPanel()

{

//make the panel

m\_centerPanel = new JPanel();

//set the layout

m\_centerPanel.setLayout(new FlowLayout());

//make the label

m\_dayLabel = new JLabel("Day");

m\_monthLabel = new JLabel("Month");

m\_yearLabel = new JLabel("Year");

//make the boxes

m\_dayBox1 = new JComboBox<Integer>(days);

m\_dayBox2 = new JComboBox<Integer>(days2);

m\_monthBox = new JComboBox<Integer>(months);

m\_yearBox = new JComboBox<Integer>(years);

//set size of boxes

m\_dayBox1.setSize(10,10);

m\_dayBox2.setSize(10,10);

m\_monthBox.setSize(10,10);

m\_yearBox.setSize(10,10);

//add components

m\_centerPanel.add(m\_dayLabel);

m\_centerPanel.add(m\_dayBox1);

m\_centerPanel.add(m\_dayBox2);

m\_centerPanel.add(m\_monthLabel);

m\_centerPanel.add(m\_monthBox);

m\_centerPanel.add(m\_yearLabel);

m\_centerPanel.add(m\_yearBox);

}

/\*\*

closeWindow method

\*/

public void closeWindow()

{

this.dispose();

}

/\*\*

makeSouthPanel

\*/

public void makeSouthPanel()

{

m\_southPanel = new JPanel();

m\_southPanel.setLayout(new FlowLayout());

//make buttons

m\_searchButton = new JButton("Search");

m\_closeButton = new JButton("Close");

//add action listeners

m\_searchButton.addActionListener(new buttonListener());

m\_closeButton.addActionListener(new buttonListener());

//add the buttons to the panel

m\_southPanel.add(m\_searchButton);

m\_southPanel.add(m\_closeButton);

}

/\*\*

getDate method

\*/

public int getDate()

{

Integer date1 = (Integer)m\_dayBox1.getSelectedItem();

Integer date2 = (Integer)m\_dayBox2.getSelectedItem();

int date = Integer.valueOf(String.valueOf(date1) + String.valueOf(date2));

return date;

}

/\*\*

public getMonth

\*/

public int getMonth()

{

Integer month = (Integer)m\_monthBox.getSelectedItem();

return month;

}

/\*\*

public getYear

\*/

public int getYear()

{

Integer year = (Integer)m\_yearBox.getSelectedItem();

return year;

}

/\*\*

button listener class

\*/

private class buttonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

if(e.getSource() == m\_searchButton)

{

//traverse the file and add all its contents to the array bag

try{

m\_file.readFile();

}

catch(IOException ex)

{

System.out.println("Noooooo");

}

int date = getDate();

int month = getMonth();

int year = getYear();

Event[] arr = m\_file.arrayTransfer(month, date, year);

//sort the array

m\_file.sort(arr);

new toDoListWindow(arr);

}

if(e.getSource() == m\_closeButton)

{

closeWindow();

}

}

}

}

**//addressPanel.java**

import javax.swing.\*;

import java.awt.\*;

import java.util.\*;

public class addressPanel extends JPanel

{

private JLabel m\_addressLabel; //initiates a LABEL for ADDRESS

private JLabel m\_cityLabel; //initiates a LABEL for CITY

private JLabel m\_stateLabel; //initiates a LABEL for STATE

private JLabel m\_zipLabel; //initiates a LABEL for ZIP CODE

private JTextField m\_addressField; //initiates a TEXTFIELD

private JTextField m\_cityField; //initiates a TEXTFIELD

private JTextField m\_stateField; //initiates a TEXTFIELD

private JTextField m\_zipField; //initiates a TEXTFIELD

/\*\*

constructor

\*/

public addressPanel()

{

//sets up the layout for the Location to have 3 rows and 1 column

setLayout(new GridLayout(4, 2));

//make labels

m\_addressLabel = new JLabel("Address");

m\_cityLabel = new JLabel("City");

m\_stateLabel = new JLabel("State");

m\_zipLabel = new JLabel("Zip Code");

//make text fields

m\_addressField = new JTextField();

m\_addressField.setEditable(true);

m\_cityField = new JTextField();

m\_cityField.setEditable(true);

m\_stateField = new JTextField();

m\_stateField.setEditable(true);

m\_zipField = new JTextField();

m\_zipField.setEditable(true);

//add all components in order

add(m\_addressLabel);

add(m\_addressField);

add(m\_cityLabel);

add(m\_cityField);

add(m\_stateLabel);

add(m\_stateField);

add(m\_zipLabel);

add(m\_zipField);

//sets the size of the window

//setSize(400, 150);

//sets up where the window will pop up on the user's screen

//setLocationRelativeTo(null);

//allows the window to be visible

setVisible(true);

}

/\*\*

getcity method

\*/

public String getCity()

{

return m\_cityField.getText();

}

/\*\*

getState method

\*/

public String getState()

{

return m\_stateField.getText();

}

/\*\*

getAddress

\*/

public String getAddress()

{

return m\_addressField.getText();

}

/\*\*

getZip

\*/

public String getZip()

{

return m\_zipField.getText();

}

/\*\*

clearCity

\*/

public void clearCity()

{

m\_cityField.setText("");

}

/\*\*

clearState

\*/

public void clearState()

{

m\_stateField.setText("");

}

/\*\*

clearAddress

\*/

public void clearAddress()

{

m\_addressField.setText("");

}

public void clearZip()

{

m\_zipField.setText("");

}

}

**//toDoListWindow.java**

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import java.util.\*;

import java.io.\*;

public class toDoListWindow extends JFrame

{

private JTextArea m\_textArea;

public toDoListWindow(Event[] arr)

{

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new FlowLayout());

setTitle("To Do List");

String events = toString(arr);

m\_textArea = new JTextArea(events);

setSize(500, 500);

add(m\_textArea);

setVisible(true);

}

/\*\*

toString

\*/

private String toString(Event [] arr)

{

String str = "";

for(int count = 0; count < arr.length; count++)

{

str += arr[count].toString();

}

return str;

}

public static void main(String[] args)throws IOException

{

LinkedUserFile myFile = new LinkedUserFile("jMoney2233.txt");

Event myEvent = new Event(2016, 2, 28, 15, 30, "My Birthday", "The best day of my life", "2018 Foxworthy Ave.", "San Jose", "95124", "California");

Event myEvent2 = new Event(2018, 2, 28, 15, 30, "My Love", "The best day of my life", "2018 Foxworthy Ave.", "San Francisco", "95124", "California");

Event myEvent3 = new Event(2017, 2, 28, 15, 30, "Afro Juice", "The best day of my life", "2018 Foxworthy Ave.", "Fresno", "95124", "California");

myFile.add(myEvent);

myFile.add(myEvent2);

myFile.add(myEvent3);

Event[] myArr = myFile.arrayTransfer(2, 28, 2018);

new toDoListWindow(myArr);

}

}

**//userWindow.java**

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import java.util.\*;

import java.io.\*;

public class userWindow extends JFrame

{

private JButton m\_addButton;

private JButton m\_searchButton;

private JButton m\_removeButton;

private String m\_filename;

public userWindow(String filename)

{

m\_filename = filename;

//set the title of the window to "User Window"

setTitle("User Window");

//allow the window to close when the user clicks the exit button

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

//set up the layout for the userWindow to have 3 rows and 1 column

setLayout(new FlowLayout());

m\_addButton = new JButton("Add");

m\_searchButton = new JButton("Search");

m\_removeButton = new JButton("Remove");

//add all the panels to the add Window

add(m\_addButton);

m\_addButton.addActionListener(new buttonListener());

add(m\_searchButton);

m\_searchButton.addActionListener(new buttonListener());

add(m\_removeButton);

//set the size of the window

setSize(300, 150);

//set up where the window will pop up on the user's screen

setLocationRelativeTo(null);

//allow the window to be visible

setVisible(true);

pack();

}

public static void main(String[] args) {

userWindow myWindow = new userWindow("jMoney2233.txt");

}

/\*\*

buttonListener class

\*/

private class buttonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

if(e.getSource() == m\_addButton)

{

try{

new EventWindow(m\_filename);

}

catch(IOException exception)

{

System.out.println("NO GOOD");

}

}

if(e.getSource() == m\_searchButton)

{

try{

searchWindow mySearch = new searchWindow(m\_filename);

}

catch(IOException exception)

{

System.out.println("NO GOOD");

}

}

}

}

}

**//EventWindow.java**

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import java.util.\*;

import java.io.\*;

public class EventWindow extends JFrame

{

private JPanel m\_middlePanel;

private JPanel m\_centerPanel;

private JPanel m\_southPanel;

private UserFile m\_file;

private Event m\_event;

private datePanel m\_datePanel;

private addressPanel m\_addressPanel;

private descriptionPanel m\_descriptionPanel;

private JButton m\_addButton;

private JButton m\_clearButton;

private JButton m\_closeButton;

public static void main(String[] args)throws IOException

{

new EventWindow("jon2222.txt");

}

/\*\*

constructor

\*/

public EventWindow(String filename)throws IOException

{

//make the text file that user will add events to

m\_file = new UserFile(filename);

//set the title

setTitle("Add Event");

//set the close operation

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

//set the layout for the window

setLayout(new BorderLayout());

makeCenterPanel();

add(m\_centerPanel, BorderLayout.CENTER);

makeSouthPanel();

add(m\_southPanel, BorderLayout.SOUTH);

//set up where the window will pop up on the screen

setLocationRelativeTo(null);

//make the window visible

setVisible(true);

pack();

}

/\*\*

makeSouthPanel method

\*/

public void makeSouthPanel()

{

m\_southPanel = new JPanel();

m\_southPanel.setLayout(new GridLayout(1,3));

//make a button

m\_addButton = new JButton("Add");

m\_clearButton = new JButton("Clear");

m\_closeButton = new JButton("Close");

//add the loginListener to the loginButton

m\_addButton.addActionListener(new ButtonListener());

m\_clearButton.addActionListener(new ButtonListener());

m\_closeButton.addActionListener(new ButtonListener());

//add the loginButton to the loginButtonPanel

m\_southPanel.add(m\_addButton);

m\_southPanel.add(m\_clearButton);

m\_southPanel.add(m\_closeButton);

}

/\*\*

makeCenterPanel

\*/

public void makeCenterPanel()

{

m\_centerPanel = new JPanel();

//set the layout for the window

m\_centerPanel.setLayout(new BorderLayout());

makeMiddlePanel();

//make the description panel

m\_descriptionPanel = new descriptionPanel();

//add all the panels to the window

m\_centerPanel.add(m\_descriptionPanel, BorderLayout.NORTH);

m\_centerPanel.add(m\_middlePanel, BorderLayout.CENTER);

}

/\*\*

makeMiddlePanel

\*/

public void makeMiddlePanel()

{

m\_middlePanel = new JPanel();

m\_middlePanel.setLayout(new GridLayout(1,2));

//make the date panel

m\_datePanel = new datePanel();

//make the address panel

m\_addressPanel = new addressPanel();

m\_middlePanel.add(m\_datePanel);

m\_middlePanel.add(m\_addressPanel);

}

/\*\*

closeWindow method

\*/

public void closeWindow()

{

this.dispose();

}

private class ButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

if(e.getSource() == m\_addButton)

{

String title = m\_descriptionPanel.getTitle();

String description = m\_descriptionPanel.getDescription();

String city = m\_addressPanel.getCity();

String state = m\_addressPanel.getState();

String zip = m\_addressPanel.getZip();

String address = m\_addressPanel.getAddress();

int year = m\_datePanel.getYear();

int month = m\_datePanel.getMonth();

int day = m\_datePanel.getDay();

int hour = m\_datePanel.getHour();

int minute = m\_datePanel.getMinute();

m\_event = new Event(year, month, day, hour, minute, title, description, address, city, zip, state);

m\_file.add(m\_event);

}

if(e.getSource() == m\_clearButton)

{

m\_descriptionPanel.clearTitle();

m\_descriptionPanel.clearDescription();

m\_addressPanel.clearCity();

m\_addressPanel.clearState();

m\_addressPanel.clearZip();

m\_addressPanel.clearAddress();

m\_datePanel.clearYear();

m\_datePanel.clearMonth();

m\_datePanel.clearDay();

m\_datePanel.clearHour();

m\_datePanel.clearMinute();

}

if(e.getSource() == m\_closeButton)

{

try

{

m\_file.print();

closeWindow();

}

catch(IOException exception)

{

System.out.println("Exception, can't print");

}

}

}

}

}

**//WelcomeWindow.java**

import javax.swing.\*;

import java.awt.event.\*;

import java.awt.\*;

import java.util.\*;

import java.io.\*;

public class WelcomeWindow extends JFrame

{

private JPanel m\_loginButtonPanel;

private JPanel m\_registerButtonPanel;

private JButton m\_loginButton;

private JButton m\_registerButton;

/\*\*

constructor

\*/

public WelcomeWindow()

{

setLayout(new GridLayout(1,2));

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setTitle("Welcome!");

makeLoginButtonPanel();

makeRegisterButtonPanel();

add(m\_loginButtonPanel);

add(m\_registerButtonPanel);

setSize(250, 75);

setLocationRelativeTo(null);

setVisible(true);

}

/\*\*

makeLoginButtonPanel method makes the login button panel

\*/

public void makeLoginButtonPanel()

{

m\_loginButtonPanel = new JPanel();

m\_loginButtonPanel.setLayout(new BorderLayout());

m\_loginButton = new JButton("Login");

m\_loginButton.addActionListener(new ButtonListener());

m\_loginButtonPanel.add(m\_loginButton, BorderLayout.CENTER);

}

/\*\*

makeRegisterButton method makes the register button

\*/

public void makeRegisterButtonPanel()

{

m\_registerButtonPanel = new JPanel();

m\_registerButtonPanel.setLayout(new BorderLayout());

m\_registerButton = new JButton("Register");

m\_registerButton.addActionListener(new ButtonListener());

m\_registerButtonPanel.add(m\_registerButton, BorderLayout.CENTER);

}

/\*\*

main method

\*/

public static void main(String[] args)

{

new WelcomeWindow();

}

private class ButtonListener implements ActionListener

{

/\*\*

actionPerformed method deals with the mouse click on both of our buttons appropriately

\*/

public void actionPerformed(ActionEvent event)

{

if(event.getSource() == m\_loginButton)

{

new LoginWindow();

}

if(event.getSource() == m\_registerButton)

{

new RegisterWindow();

}

}

}

}

**//descriptionPanel.java**

import java.awt.GridLayout;

import javax.swing.\*;

public class descriptionPanel extends JPanel

{

private JLabel titlelabel;

private JLabel descriptionlabel;

private JTextField titlefield;

private JTextField descriptionfield;

public descriptionPanel()

{

setLayout(new GridLayout(2,1));

titlelabel = new JLabel("Title");

descriptionlabel = new JLabel("Description");

titlefield = new JTextField();

titlefield.setEditable(true);

descriptionfield = new JTextField();

descriptionfield.setEditable(true);

add(titlelabel);

add(titlefield);

add(descriptionlabel);

add(descriptionfield);

}

/\*\*

getDescription method

\*/

public String getDescription()

{

return descriptionfield.getText();

}

/\*\*

getTitle method

\*/

public String getTitle()

{

return titlefield.getText();

}

/\*\*

setTitle method

\*/

public void clearTitle()

{

titlefield.setText("");

}

/\*\*

setDescription

\*/

public void clearDescription()

{

descriptionfield.setText("");

}

}

**//datePanel.java**

import java.awt.GridLayout;

import javax.swing.\*;

import static javax.swing.JFrame.EXIT\_ON\_CLOSE;

public class datePanel extends JPanel {

private JTextField monthField;

private JTextField dayField;

private JTextField yearField;

private JTextField hourField;

private JTextField minuteField;

private JLabel monthLabel;

private JLabel dayLabel;

private JLabel yearLabel;

private JLabel hourLabel;

private JLabel minuteLabel;

private JButton createDateButton;

private JOptionPane pane;

private String month, day, year, hour, minute;

public datePanel()

{

setLayout(new GridLayout(5, 2));

monthLabel = new JLabel("Month:");

monthField = new JTextField();

monthField.setEditable(true);

dayLabel = new JLabel("Day:");

dayField = new JTextField();

dayField.setEditable(true);

yearLabel = new JLabel("Year:");

yearField = new JTextField();

yearField.setEditable(true);

hourLabel = new JLabel("Hour:");

hourField = new JTextField();

hourField.setEditable(true);

minuteLabel = new JLabel("Minute:");

minuteField = new JTextField();

minuteField.setEditable(true);

add(monthLabel);

add(monthField);

add(dayLabel);

add(dayField);

add(yearLabel);

add(yearField);

add(hourLabel);

add(hourField);

add(minuteLabel);

add(minuteField);

setVisible(true);

}

/\*\*

getMonth method

\*/

public int getMonth()

{

String month = monthField.getText();

month = month.trim();

int numMonth = Integer.parseInt(month);

return numMonth;

}

/\*\*

getDay method

\*/

public int getDay()

{

String day = dayField.getText();

day = day.trim();

int numDay = Integer.parseInt(day);

return numDay;

}

/\*\*

getYear method

\*/

public int getYear()

{

String year = yearField.getText();

year = year.trim();

int numYear = Integer.parseInt(year);

return numYear;

}

/\*\*

getHour

\*/

public int getHour()

{

String hour = hourField.getText();

hour = hour.trim();

int numHour = Integer.parseInt(hour);

return numHour;

}

/\*\*

getMinute

\*/

public int getMinute()

{

String minute = minuteField.getText();

minute = minute.trim();

int numMinute = Integer.parseInt(minute);

return numMinute;

}

/\*\*

clearMonth

\*/

public void clearMonth()

{

monthField.setText("");

}

/\*\*

clearYear method

\*/

public void clearYear()

{

yearField.setText("");

}

/\*\*

clearDay

\*/

public void clearDay()

{

dayField.setText("");

}

/\*\*

clear hour

\*/

public void clearHour()

{

hourField.setText("");

}

/\*\*

clearMinute

\*/

public void clearMinute()

{

minuteField.setText("");

}

}

**//LinkedUserFile.java**

import java.io.\*;

import java.util.\*;

public class LinkedUserFile extends FileWriter

{

private String m\_fileName;

private EventLinkedBag m\_eventBag;

private PrintWriter m\_writer;

private CalendarScanner m\_reader;

/\*\*

Constructor

\*/

public LinkedUserFile(String fileName)throws IOException

{

super(fileName, true);

m\_reader = new CalendarScanner(fileName);

m\_writer = new PrintWriter(this);

m\_eventBag = new EventLinkedBag();

}

/\*\*

add method

\*/

public boolean add(Event e)

{

return m\_eventBag.add(e);

}

/\*\*

remove method

@return event

\*/

public Event remove()

{

return m\_eventBag.remove();

}

/\*\*

remove a specific event

@return boolean

\*/

public boolean remove(Event e)

{

return m\_eventBag.remove(e);

}

/\*\*

getNumberOfEvents

@return int number of events on the calander

\*/

public int getNumberOfEvents()

{

return m\_eventBag.getCurrentSize();

}

/\*\*

isEmpty method

@return boolean

\*/

public boolean isEmpty()

{

return m\_eventBag.isEmpty();

}

/\*\*

isFull method

@return boolean

\*/

public boolean isFull()

{

return m\_eventBag.isFull();

}

/\*\*

clear method

\*/

public void clear()

{

m\_eventBag.clear();

}

/\*\*

getFrequencyOf

@return int

\*/

public int getFrequencyOf(Event e)

{

return m\_eventBag.getFrequencyOf(e);

}

/\*\*

contains method checks to see if a given event is in the file

@return boolean

\*/

public boolean contains(Event e)

{

return m\_eventBag.contains(e);

}

/\*\*

toArray method returns an array of events

@return Event[]

\*/

public Object[] toArray()

{

Object[] arr = m\_eventBag.toArray();

return arr;

}

/\*\*

print method

\*/

public void print(Event e)throws IOException

{

m\_writer.print(e.toString());

this.close();

}

/\*\*

print

\*/

public void print()throws IOException

{

Object[] arr = toArray();

for(int count = 0; count < arr.length; count++)

{

m\_writer.print(arr[count]);

}

this.close();

}

/\*\*

readFile method

\*/

public void readFile()throws IOException

{

while(m\_reader.hasNextLine())

{

Event newEvent = m\_reader.getEvent();

add(newEvent);

}

}

/\*\*

sort

\*/

public void sort(Event[] e)

{

LinkedMergeSort.mergeSort(e, 0, e.length - 1);

}

/\*\*

arrayTransfer

\*/

public Event[] arrayTransfer(int month, int day, int year)

{

return m\_eventBag.arrayTransfer(month, day, year);

}

}

**//CalendarScanner.java**

import java.util.StringTokenizer;

import java.util.Scanner;

import java.io.\*;

import java.time.\*;

public class CalendarScanner

{

private File m\_file;

private Scanner m\_fileScanner;

public static void main(String[] args)throws IOException, DateTimeException

{

CalendarScanner myScanner = new CalendarScanner("date.txt");

Event myEvent = myScanner.getEvent();

System.out.println(myEvent);

}

public CalendarScanner(String filename)throws IOException

{

m\_file = new File(filename);

m\_fileScanner = new Scanner(m\_file);

}

/\*\*

traverseDate

\*/

private String[] traverseLine(String line)

{

StringTokenizer myTokenizer = new StringTokenizer(line, ",: ");

String[] arr = new String[myTokenizer.countTokens()];

for(int count = 0; count < arr.length; count++)

{

arr[count] = myTokenizer.nextToken();

}

return arr;

}

/\*\*

getEvent method

\*/

public Event getEvent()throws DateTimeException

{

String title = getTitle();

String description = getDescription();

String address = getAddress();

String location = getLocation();

String city = getCity(location);

String state = getState(location);

String zip = getZip(location);

String date = getDate();

String dayOfWeek = getDayOfWeek(date);

int month = getMonthOfYear(date);

int day = getDayOfMonth(date);

int year = getYear(date);

int hour = getHour(date);

int minute = getMinute(date);

String AmPm = getAmPm(date);

Event myEvent = new Event(year, month, day, hour, minute, title, description, address, city, zip, state);

return myEvent;

}

/\*\*

getDescription method

\*/

public String getDescription()

{

String description = m\_fileScanner.nextLine();

return description;

}

/\*\*

getTitle method

\*/

public String getTitle()

{

String title = m\_fileScanner.nextLine();

return title;

}

/\*\*

getAddress method

\*/

public String getAddress()

{

String address = m\_fileScanner.nextLine();

return address;

}

/\*\*

get

/\*\*

getDate method

\*/

public String getDate()

{

String date = m\_fileScanner.nextLine();

return date;

}

/\*\*

getLocation method

\*/

public String getLocation()

{

String location = m\_fileScanner.nextLine();

return location;

}

/\*\*

getCity method

\*/

public String getCity(String location)

{

String [] arr = traverseLine(location);

String city = arr[0];

return city;

}

/\*\*

getState method

\*/

public String getState(String location)

{

String [] arr = traverseLine(location);

String state = arr[1];

return state;

}

/\*\*

getZip method

\*/

public String getZip(String location)

{

String[] arr = traverseLine(location);

String zip = arr[2];

return zip;

}

/\*\*

getDayOfWeek method

\*/

public String getDayOfWeek(String date)

{

String[] arr = traverseLine(date);

String dayOfWeek = arr[0];

return dayOfWeek;

}

/\*\*

getMonthOfYear method

\*/

public int getMonthOfYear(String date)

{

String[] arr = traverseLine(date);

String monthOfYear = arr[1];

int month;

switch(monthOfYear)

{

case "January":

month = 1;

break;

case "February":

month = 2;

break;

case "March":

month = 3;

break;

case "April":

month = 4;

break;

case "May":

month = 5;

break;

case "June":

month = 6;

break;

case "July":

month = 7;

break;

case "August":

month = 8;

break;

case "September":

month = 9;

break;

case "October":

month = 10;

break;

case "November":

month = 11;

break;

case "December":

month = 12;

break;

default:

month = -1;

}

return month;

}

/\*\*

getDayOfMonth method

\*/

public int getDayOfMonth(String date)

{

String[] arr = traverseLine(date);

String dayOfMonth = arr[2];

int numDayOfMonth = Integer.parseInt(dayOfMonth);

return numDayOfMonth;

}

/\*\*

getYear method

\*/

public int getYear(String date)

{

String[] arr = traverseLine(date);

String year = arr[3];

int numYear = Integer.parseInt(year);

return numYear;

}

/\*\*

getHour method

\*/

public int getHour(String date)

{

String[] arr = traverseLine(date);

String hour = arr[4];

int numHour = Integer.parseInt(hour);

return numHour;

}

/\*\*

getMinute method

\*/

public int getMinute(String date)

{

String[] arr = traverseLine(date);

String minute = arr[5];

int numMinute = Integer.parseInt(minute);

return numMinute;

}

/\*\*

getAmPm method

\*/

public String getAmPm(String date)

{

String[] arr = traverseLine(date);

String amPM = arr[6];

return amPM;

}

/\*\*

hasNextLine method

\*/

public boolean hasNextLine()

{

return m\_fileScanner.hasNextLine();

}

}

**//EventLinkedBag.java**

// Only changes made was changing LinkedBag from generic types to Event types

// Added arrayTransfer method, see method for details

public class EventLinkedBag {

private Node firstNode;

private int numberOfEntries;

public EventLinkedBag()

{

firstNode = null;

numberOfEntries = 0;

}

//@Override

public int getCurrentSize()

{

return numberOfEntries;

}

//@Override

public boolean isFull()

{

return false;

}

//@Override

public boolean isEmpty()

{

return firstNode == null;

}

//@Override

public boolean add(Event newEntry)

{

Node newNode = new Node(newEntry);

newNode.next = firstNode;

firstNode = newNode;

numberOfEntries++;

return true;

}

//@Override

public Event remove() //self-attempt failed

{

Event result = null;

if (firstNode != null)

{

result = firstNode.data;

firstNode = firstNode.next;

numberOfEntries--;

}

return result;

}

private Node getReferenceTo(Event anEntry)

{

boolean located = false;

Node currentNode = firstNode;

while(!located && (currentNode != null))

{

if (anEntry == currentNode.data)

located = true;

else

currentNode = currentNode.next;

}

return currentNode;

}

//@Override

public boolean remove(Event anEntry)

{

boolean result = false;

Node removal = getReferenceTo(anEntry);

if(removal != null)

{

removal.data = firstNode.data;

remove();

result = true;

}

return result;

}

//@Override

public void clear()

{

while (!isEmpty())

remove();

}

//@Override

public int getFrequencyOf(Event anEntry) //self-attempt

{

int counter = 0;

Node currentNode = firstNode;

for(int index = 0; index < numberOfEntries && currentNode != null; index++)

{

if(anEntry == currentNode.data)

counter++;

currentNode = currentNode.next;

}

return counter;

}

//@Override

public boolean contains(Event anEntry) //self-attempt

{

boolean within = false;

Node currentNode = firstNode;

while(!within && (currentNode != null))

{

if(anEntry == currentNode.data)

within = true;

else

currentNode = currentNode.next;

}

return within;

}

//@Override

public Event[] toArray()

{

Event[] result = new Event[numberOfEntries];

Node currentNode = firstNode;

for(int index = 0; index < numberOfEntries && currentNode != null; index++)

{

result[index] = currentNode.data;

currentNode = currentNode.next;

}

return result;

}

/\*\* arrayTransfer

\* Transfers events with the same dates into an event array

\* @param month month inputted by user

\* @param day day inputted by user

\* @param year year inputted by user

\* @result result Event object created by user input

\* @return sameDates Event array to hold exact number of objects with same dates

\*/

public Event[] arrayTransfer(int month, int day, int year)

{

Event dateSearch = new Event(month, day, year);

Event[] tempEvent = new Event[numberOfEntries];

Node currentNode = firstNode;

int index = 0;

int newArrayIndex = 0;

while(currentNode != null)

{

if (currentNode.data.equals(dateSearch))

{

tempEvent[index] = currentNode.data;

index++;

newArrayIndex++;

}

currentNode = currentNode.next;

}

// Creates new array with exact number of elements

// Used to prevent null entries

Event[] sameDates = new Event[newArrayIndex];

for(int eventIndex = 0; eventIndex < newArrayIndex; eventIndex++)

sameDates[eventIndex] = tempEvent[eventIndex];

return sameDates;

}

@Override

public String toString() //self-attempt

{

String str = "";

Node currentNode = firstNode;

for(int index = 0; index < numberOfEntries && currentNode != null; index++)

{

str = str + currentNode.data;

currentNode = currentNode.next;

}

return str;

}

private class Node

{

private Event data;

private Node next;

private Node()

{

this(null, null);

}

private Node(Event dataPortion)

{

this(dataPortion, null);

}

private Node(Event dataPortion, Node nextNode)

{

data = dataPortion;

next = nextNode;

}

}

}

**//Event.java**

import java.util.\*;

import java.time.\*;

import java.time.Month;

import java.time.format.TextStyle;

import java.time.format.DateTimeFormatter;

import java.text.DecimalFormat;

public class Event implements Comparable<Event>

{

    private LocalDateTime m\_event;

    private Month m\_month;

    private int m\_dayOfMonth;

    private int m\_year;

    private int m\_hourOfDay;

    private int m\_minute;

    private Locale m\_locale = Locale.getDefault();

    private DateTimeFormatter m\_dateFormatter = DateTimeFormatter.ofPattern("EEEE, MMMM dd, yyyy h:mm a");

    private DecimalFormat m\_minuteFormatter = new DecimalFormat("##");

    private String m\_title;

    private String m\_description;

    private String m\_address;

    private String m\_city;

    private String m\_state;

    private String m\_zipCode;

    public Event()

    {

        this(2016, 12, 31, 8, 51, "Birthday", "My Birthday", "Address", "City", "Zip", "State");

    }

    public Event(int year, int month, int dayOfMonth, int hour, int minute, String title, String description, String address, String city, String zip, String state)throws DateTimeException

    {

        //title

        m\_title = title;

        //description of event

        m\_description = description;

        //address

        m\_address = address;

        //city

        m\_city = city;

        //state

        m\_state = state;

        //zipCode

        m\_zipCode = zip;

        //year

        m\_year = year;

        //month

        m\_month = Month.of(month);

        //dayOfMonth

        m\_dayOfMonth = dayOfMonth;

        //hour of day

        m\_hourOfDay = hour;

        //minute of day

        m\_minute = minute;

        m\_event = LocalDateTime.of(m\_year, m\_month, m\_dayOfMonth, m\_hourOfDay, m\_minute);

    }

    public Event(int month, int day, int year)

    {

        this(year, month, day, 0, 0);

    }

    public Event(int year, int month, int dayOfMonth, int hour, int minute)throws DateTimeException

    {

        this(year, month, dayOfMonth, hour, minute, "", "", "", "", "", "");

    }

    /\*\*

     setTitle

     @param title String representation of the description's title

     \*/

    public void setTitle(String title)

    {

        m\_title = title;

    }

    /\*\*

     getTitle

     @return String  String representation of the description's title

     \*/

    public String getTitle()

    {

        return m\_title;

    }

    /\*\*

     setDescription

     @param description

     \*/

    public void setDescription(String description)

    {

        m\_description = description;

    }

    /\*\*

     getDescription

     @return the event description

     \*/

    public String getDescription()

    {

        return m\_description;

    }

    /\*\*

     setAddress

     @param String  String representation of the description's address

     \*/

    public void setAddress(String address)

    {

        m\_address = address;

    }

    /\*\*

     getAddress

     @return String  String representation of the description's address

     \*/

    public String getAddress()

    {

        return m\_address;

    }

    /\*\*

     setCity

     @param String  String representation of the description's city

     \*/

    public void setCity(String city)

    {

        m\_city = city;

    }

    /\*\*

     getCity

     @return String  String representation of the description's city

     \*/

    public String getCity()

    {

        return m\_city;

    }

    /\*\*

     setZip

     @param String  String representation of the description's zip

     \*/

    public void setZip(String zip)

    {

        m\_zipCode = zip;

    }

    /\*\*

     getZip

     @return String  String representation of the description's zip

     \*/

    public String getZip()

    {

        return m\_zipCode;

    }

    /\*\*

     setState

     @param String  String representation of the description's state

     \*/

    public void setState(String state)

    {

        m\_state = state;

    }

    /\*\*

     getState

     @return String  String representation of the description's state

     \*/

    public String getState()

    {

        return m\_state;

    }

    /\*\*

     toString

     @return str    string that represents the event

     \*/

    public String toString()

    {

        String str = m\_title + "\n" + m\_description + "\n" + m\_address + "\n" + m\_city + ", " + m\_state + " " + m\_zipCode + "\n" + m\_event.format(m\_dateFormatter)+ "\n";

        return str;

    }

    /\*\*

     setYear

     @param year    int representation the event's year field

     \*/

    public void setYear(int year)

    {

        m\_year = year;

        m\_event = m\_event.withYear(m\_year);

    }

    /\*\*

     getYear

     @return int representation of the event's year field

     \*/

    public int getYear()

    {

        return m\_event.getYear();

    }

    /\*\*

     setMonth

     @param month   int representation of the event's month field

     \*/

    public void setMonth(int month)

    {

        m\_month = Month.of(month);

        m\_event = m\_event.withMonth(m\_month.getValue());

    }

    /\*\*

     getMonth

     @return String representation of the event's month field

     \*/

    public String getMonth()

    {

        return m\_event.getMonth().getDisplayName(TextStyle.FULL, m\_locale);

    }

    public int getIntMonth()

    {

        return m\_month.getValue();

    }

    /\*\*

     setDayOfMonth

     @param dayOfMonth  int representation of the event's day of the month field

     \*/

    public void setDayOfMonth(int dayOfMonth)

    {

        m\_dayOfMonth = dayOfMonth;

        m\_event = m\_event.withDayOfMonth(m\_dayOfMonth);

    }

    /\*\*

     getDayOfMonth

     @return int representation of the event's day of the month field

     \*/

    public int getDayOfMonth()

    {

        return m\_event.getDayOfMonth();

    }

    /\*\*

     setHour

     @param hour    int representation of the event's hour field

     \*/

    public void setHour(int hour)

    {

        m\_hourOfDay = hour;

        m\_event = m\_event.withHour(m\_hourOfDay);

    }

    /\*\*

     getHour

     @return int representation of the event's hour field

     \*/

    public int getHour()

    {

        return m\_event.getHour();

    }

    /\*\*

     setMinute

     @param minute  int representation of the event's minute field

     \*/

    public void setMinute(int minute)

    {

        int m\_minute = minute;

        m\_event = m\_event.withMinute(m\_minute);

    }

    /\*\*

     getMinute

     @return int representation of the event's minute field

     \*/

    public int getMinute()

    {

        return m\_event.getMinute();

    }

    /\*\*

     \* getMilitaryTime

     \* Method concatenates hours and minutes into one integer for easy sorting

     \* Adds a number 0 in front of minutes should the minutes be less than 10 minutes

     \* @return integer concatenated time of hours and minutes

     \*/

    public int getMilitaryTime()

    {

        if (getMinute() < 10)

            return Integer.valueOf(String.valueOf(getHour()) +

                    String.valueOf("0" + getMinute()));

        else

            return Integer.valueOf(String.valueOf(getHour()) +

                String.valueOf(getMinute()));

    }

    /\*\*

     getDayOfWeek

     @return String representation of the event's day of the week

     \*/

    public String getDayOfWeek()

    {

        DayOfWeek m\_day = m\_event.getDayOfWeek();

        return m\_day.getDisplayName(TextStyle.FULL, m\_locale);

    }

    /\*\*

     \* compareTo method

     \* compare the military time of event objects in order to sort

     \* @param event object that will be compared to other event object

     \* @return -1 if the event passed has a larger time, 1 if the event passed

     \*   has less time

     \*/

    @Override

    public int compareTo(Event event)

    {

         if (getMilitaryTime() < event.getMilitaryTime())

             return -1;

         else if (getMilitaryTime() > event.getMilitaryTime())

             return 1;

         else

             return 0;

    }

    public boolean equals(Event e)

    {

        Event e1 = (Event)e;

        return (this.m\_year == e1.m\_year) && (this.m\_month == e1.m\_month) && (this.m\_dayOfMonth == e1.m\_dayOfMonth);

    }

}