**Development**

All Classes:

JForm -

Java Class -

Graphical user interface

Description automatically generated with medium confidence

List of Techniques used:

1. Web-scraping with JSoup library to fetch data from an online source (www.magicseaweed.com)
2. CSV reading/writing with the openCSV library, used for the login system
3. File reading/writing/Creation used for the Surf Log system
4. Current Date and Time using the LocalDateTime Library
5. Linear Search

**The Login**

Login Window:

Graphical user interface, application

Description automatically generated

The login window features three buttons:

**Close:**

When this button is pressed the Login form, along with the rest of the program is terminated completely this is seen in the following code snippet.

Text

Description automatically generated

An action listener was attached to the button causes the program to exit with a code of 0, or in other words success.

**Login:**

Linear search algorithm used to search through the CSV database, as seen in the annotated code snippet below. Throwing an error is ID / password is incorrect.

Text

Description automatically generated

Depending on the output of validate() either an error message or a new Home JFrame will be opened

Text

Description automatically generated

**Home**

Home Form

A picture containing text, screenshot, outdoor

Description automatically generated

The image seen in the bottom will change every time on startup, it draws from the magicseaweed.com image archive, with over 300,000 images uploaded by its users worldwide. It’s safe to say you won’t see the same image twice!

The Forecast that is listed is based off the break ID that the user inputs in their account creation (See New Account). This is the ID that the InfoFetcher Class uses to fetch surf forecast data. More on that later.

Text

Description automatically generated

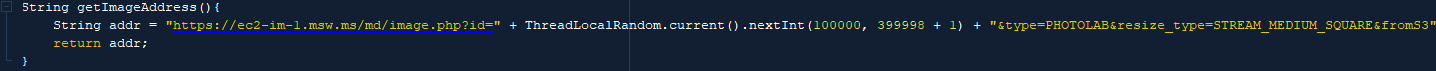


On instantiation of a Home object the **setImage()** method of the class is called to set the image that you see at the bottom.

Text

Description automatically generated

The setImage method then uses the **getImageAddress()** method from the InfoFetcher class. This returns the link to a random image in the database.



I did some experimentation with the image IDs and the ones that actually had images attached and I found that IDs between 100,000 and 400,000 were the most consistent. Thus the getImageAddress() chooses a random ID between 100,000 and 400,000 and returns it with the random ID embedded in the link for example:

[**https://ec2-im-1.msw.ms/md/image.php?id=223412&type=PHOTOLAB&resize\_type=STREAM\_MEDIUM\_SQUARE&fromS3**](https://ec2-im-1.msw.ms/md/image.php?id=223412&type=PHOTOLAB&resize_type=STREAM_MEDIUM_SQUARE&fromS3)

If in the off chance the image ID fetched does not have a picture attached to it, the following error message (client requested) will pop up:

Graphical user interface, application, Word

Description automatically generated

If this error occurs, as it says, the refresh button on the home page should be pressed and a new image ID will be chosen

**setImage()** then creates a new image object and uses **ImageIO.read()** to fetch the image from the link, then scales it to the correct size and then sets the **image2 label** icon to the fetched image.

**Refresh**

**A screenshot of a computer

Description automatically generated with medium confidence**

The refresh button when pressed simply calls the same methods on startup, allowing both the forecast and the image to be changed without having to completely restart the application.

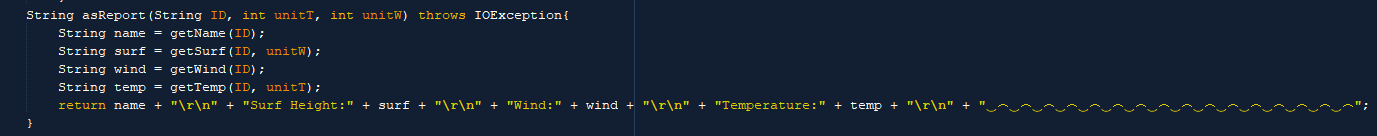
**New Entry**

New Entry Form

**Graphical user interface, text

Description automatically generated**

Shown above is the generated text that appears when a new log is instantiated. The user is passed into the Log and then their break ID used to generate the text using the **asReport()** method of the InfoFetcher class. As seen below:

The units that the user initially set are also taken in as parameters.

Further the date displayed in the uppermost text box is generated by the following, it uses the built-in **LocalDateTime** library that comes by default in the Java NetBeans IDE. I chose to set dots in between the dd.mm.yyyy, due to the fact that it was originally in the documentation as backslashes which cause errors in saving, as the fileWriter/Reader reads it as a pathway not a filename.

A computer screen capture

Description automatically generated with medium confidence

**Save**

Once a user has inputted their surf session description, they simply have to press the save button. This then through the **saveLog()** method of the class instantiates a new Log() object. And then through Logs **newLog()** and **writeLog()** methods the log is saved as “USERID.dd.mm.yyyy.txt”. While the data entered in the Large text area is written to the file.

Text

Description automatically generated

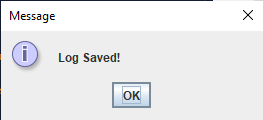
**Log.java:**

**Text

Description automatically generated**

To read/write/create new files my program utilizes the FileWriter, FileReader and File libraries that are default with java.

After writing of the log the following information popup is displayed:



**Open**

To open logs, the user simply has to input the date of the surf session they would like to view in the top text field then press the “open” button, this then usen the **retrieveLog()** method of Log(), as seen below, which outputs the contents of the log that the user requested as a string.

Text

Description automatically generated

The large text areas text value is then set to the contents of the file.

A screenshot of a computer

Description automatically generated with medium confidence

And the following info message is displayed:

Graphical user interface, application, Word

Description automatically generated

**New User**

New User Form

Graphical user interface

Description automatically generated

The ID is auto generated, and un-editable by the user, to prevent users choosing the same ID number. The Surf Break ID Field, is a 4 digit number that is unique to each surf break on magicseaweed.com’s website. For example(highlighted numbers).





I found that it doesn’t matter what is before the surf break ID, and as long as there is an ID you will be redirected to the right page, this is especially useful because then the user isnt required to type in the whole name of their surf break, which could open up room for errors.

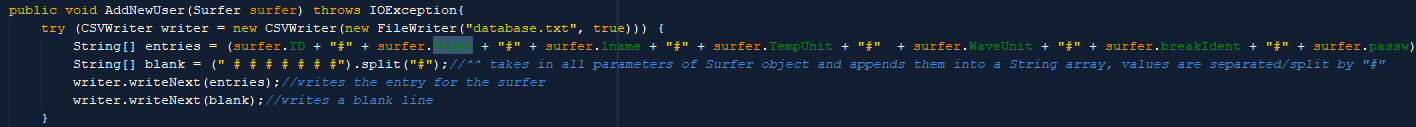
For example:



And



Redirect to the same page. Which is a useful quirk of the sites structure that I took advantage of.

Then through the Database class, and its AddNewUser() method, the new surfer is added to the csv database through the use of openCSV’s **CSVWriter()**

**InfoFetcher Class**

Arguably the webscraping portion of this program is one of the most interesting and useful features of this program. For the webscraping in this program I chose to use JSoup.

A picture containing chart

Description automatically generatedAn annotated example code snippet for fetching surf-height data from magicseaweed.com from my program is below

For example I will use one of my favorite beaches to surf on with a break ID of 3930:

<https://magicseaweed.com/Southlands-Surf-Report/3930/>

Graphical user interface, text, application

Description automatically generated

Here you can see that through google chromes inspection tool, I can pinpoint exactly the element I want to select from the webpage, in this case it’s the <li> tag with a class of “rating-text text-dark” that contains the text that has the surf height information that is wanted. This process was repeated for the other aspects of the site that I wanted to include (tides, temperatures, etc)

Graphical user interface, application

Description automatically generated

Word Count: 1045