1. Introduction:

The programming language used in this program is Java, and is programmed on NetBeans. NetBeans was the application that was used, because of the Java Swing Tools which allowed Graphic User Interface. The program is a programmed design for sports teams to keep track of their data while they are working out.

Words: 52

2. Summary Lists of Techniques:

- parameter passing
- random number generation
- for loop
- while loop
- nested loops
- method returning a value
- arrays, 2d arrays
- user-defined objects made from an OOP "template" class
- encapsulation of private methods that work on public attribute of a "template" class
- simple and compound selection (if/else)
- error handling (for example catching a divide by 0 error, or a null pointer while using an array of object…)
- GUI tabs
- GUI popup menus
- Use of a flag value (such as -999, or "not set yet")
- overloaded constructors, which work differently depending on the parameters sent
- inheritance between a superclass and a subclass

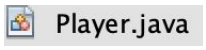
Words: 0 (Bulleted List)

3. Structure Program

The structure of the program is object-oriented programing, this had lead to being able to decompose bigger problems into smaller problems. The MainGUI Class is the main class, it allows the user to interact with the graphic user interface. The user can input problems, and see the output. This allows the players to be able to keep track of their working out programs.

MainGUI.java

The class, Player, is an object template class to allow information to be stored in an organized order, this is an aggregation of the MainGUI, the MainGUI would ‘has a’ Player Class. In this class, it has a string of player’s username and password, along with the ArrayList of cardioLogs and liftingLogs, this allows the information to be stored within each instance. By having the Player class it is able to organize each player and workout program. This helps the programming easier to track down if there are any problems with the username and password not being able to work or the array lists not working.

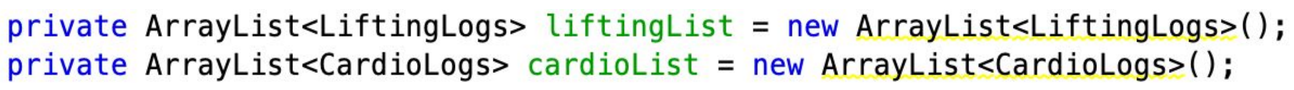

Player.java

Another class that that aggregates from the MainGUI is Logs. The class Logs contains two Strings, which are dates and name. By having this log class, it would make the two easier because it doesn’t require the code to repeat for both of them. Such as coding for the names, and the date. The two-class that inherent from Logs are the class LiftingLogs and CardioLogs, which also contains ArrayList. The variable from two of these classes can be accessed from the main class by getCardioLogs().get(0) and getLiftingLogs().get(0). The point of these two classes is to make it easier to spot a problem, for example, if CardioLogs works but LiftingLogs doesn't then it would be easier to just look at the LiftingLogs to figure it out.

Word Count:298

4. Data Structured Used

1. Array List



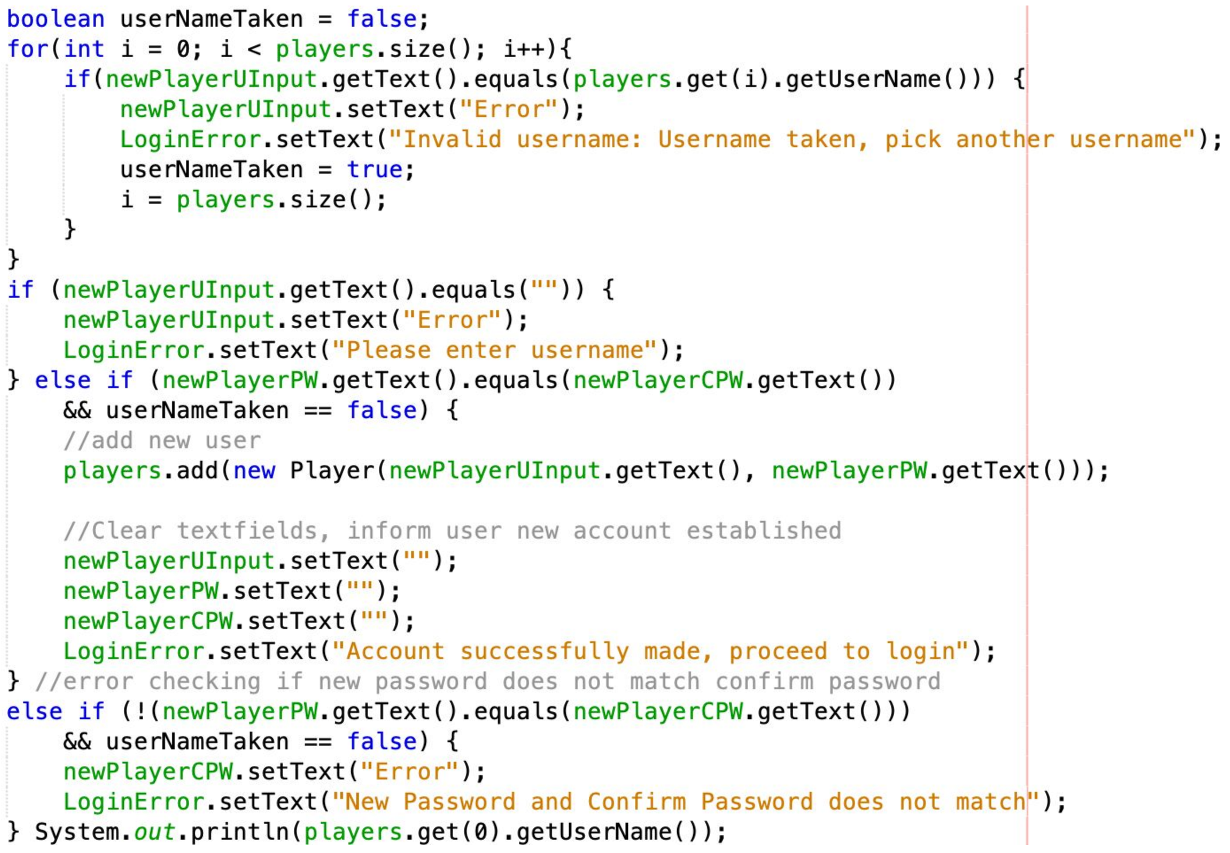
```
private ArrayList<LiftingLogs> liftingList = new ArrayList<LiftingLogs>();
private ArrayList<CardioLogs> cardioList = new ArrayList<CardioLogs>();
```

ArrayList was used in the program to store different data without having a limit, it would adjust according to how much the user inputs. if the array were used then there would be a set number that the user is able to store their cardio and lifting information. Therefore, having ArrayList there's no limitation to how much the user could input into the programming.

Word Count: 64

5. Main Unique Algorithm

1. Create New Username and Error Handling



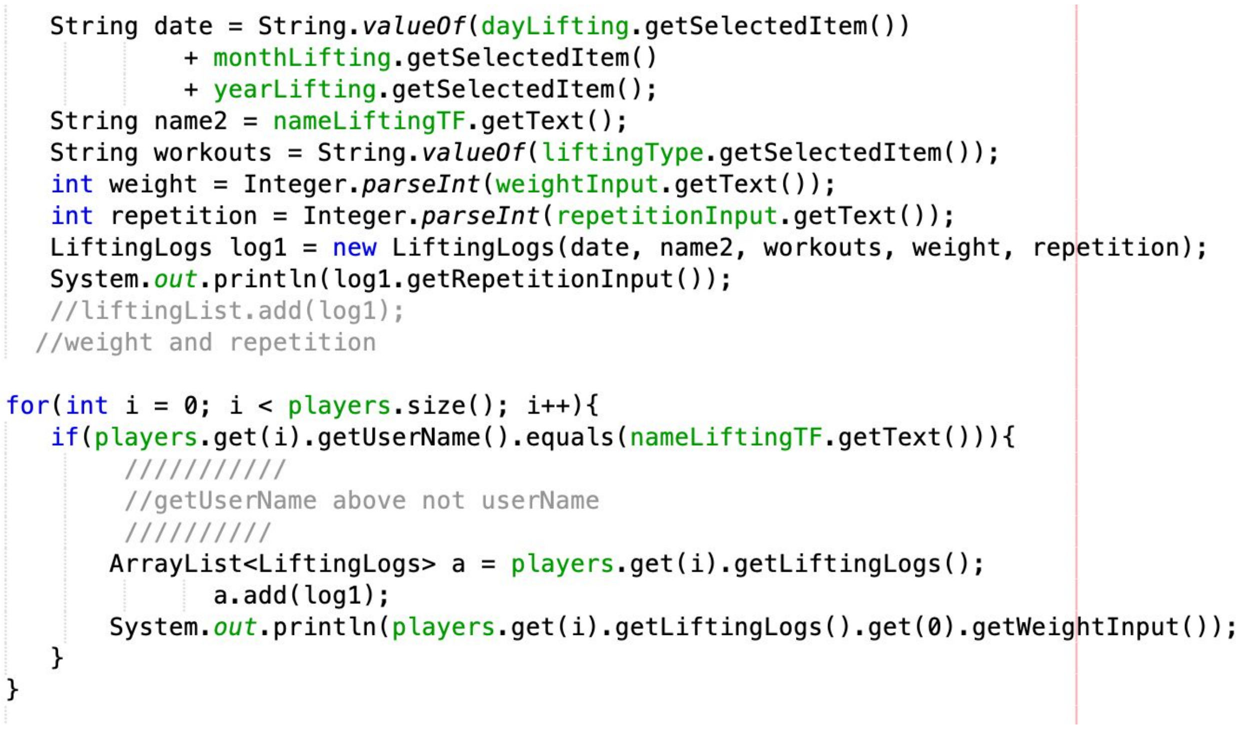
```
boolean userNameTaken = false;
for(int i = 0; i < players.size(); i++){
    if(newPlayerUInput.getText().equals(players.get(i).getUserName())) {
        newPlayerUInput.setText("Error");
        LoginError.setText("Invalid username: Username taken, pick another username");
        userNameTaken = true;
        i = players.size();
    }
}
if (newPlayerUInput.getText().equals("")) {
    newPlayerUInput.setText("Error");
    LoginError.setText("Please enter username");
} else if (newPlayerPW.getText().equals(newPlayerCPW.getText())
    && userNameTaken == false) {
    //add new user
    players.add(new Player(newPlayerUInput.getText(), newPlayerPW.getText()));

    //Clear textfields, inform user new account established
    newPlayerUInput.setText("");
    newPlayerPW.setText("");
    newPlayerCPW.setText("");
    LoginError.setText("Account successfully made, proceed to login");
} //error checking if new password does not match confirm password
else if (!(newPlayerPW.getText().equals(newPlayerCPW.getText()))
    && userNameTaken == false) {
    newPlayerCPW.setText("Error");
    LoginError.setText("New Password and Confirm Password does not match");
} System.out.println(players.get(0).getUserName());
```

Every time the button "create" is released, the codes above are being run through. The first part of the code checks if the username has been used before or not. If it has then there would be an error that pops up. The second part is if the user didn't input anything in the user name an error would pop up. The first else if, is if everything is put incorrectly, there's no user name taken and password and confirm password matched, then the account would be created. The part is if the user's password and confirm password did not match.

By having this in the program, it allows each player to have their own personal logs, rather than having to share with others and having it unorganized. This would help during the process of entering cardio and lifting logs along with displaying tables.

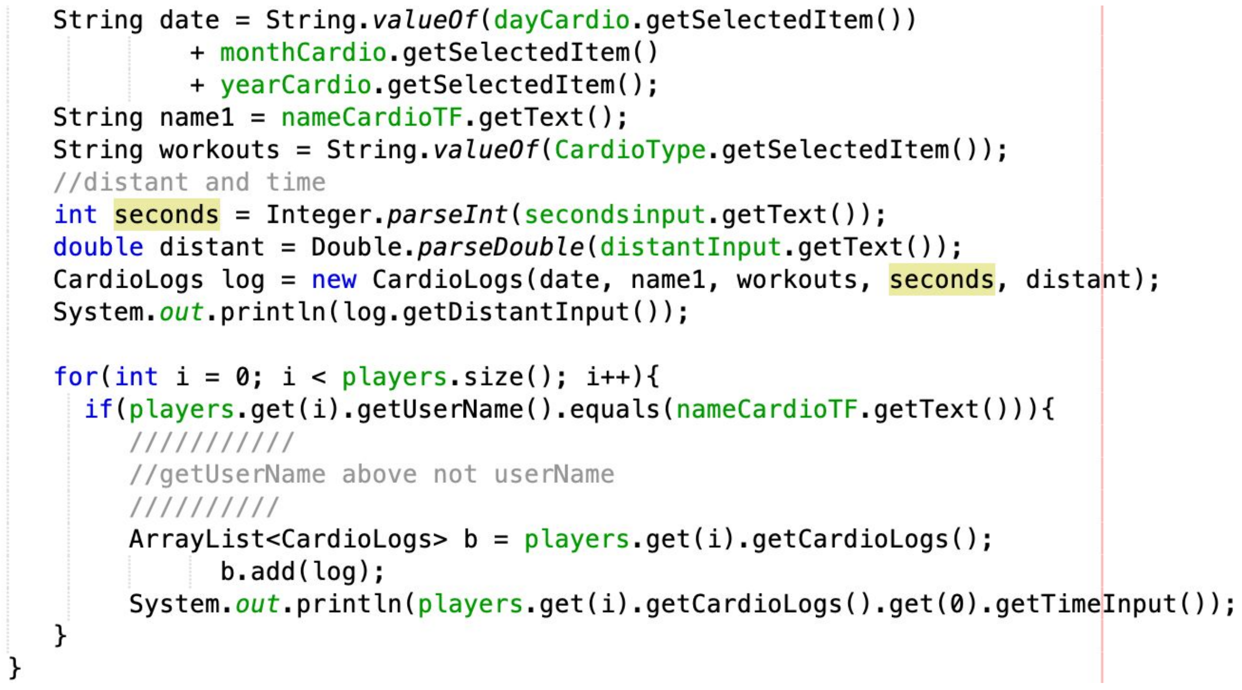
2. Entering Cardio and Lifting Logs



```
   String date = String.valueOf(dayLifting.getSelectedItem())
           + monthLifting.getSelectedItem()
           + yearLifting.getSelectedItem();
   String name2 = nameLiftingTF.getText();
   String workouts = String.valueOf(liftingType.getSelectedItem());
   int weight = Integer.parseInt(weightInput.getText());
   int repetition = Integer.parseInt(repetitionInput.getText());
   LiftingLogs log1 = new LiftingLogs(date, name2, workouts, weight, repetition);
   System.out.println(log1.getRepetitionInput());
   //liftingList.add(log1);
  //weight and repetition

for(int i = 0; i < players.size(); i++){
   if(players.get(i).getUserName().equals(nameLiftingTF.getText())){
        ///////////
        //getUserName above not userName
        //////////
       ArrayList<LiftingLogs> a = players.get(i).getLiftingLogs();
              a.add(log1);
       System.out.println(players.get(i).getLiftingLogs().get(0).getWeightInput());
   }
}
```

Every time the enter button is clicked, the date, name, workouts, weight, and repetition would be saved into the player's data. The second part of the code is to search for the player's name to see if they are in the database and if they are then it would be added for this case to their LiftingLogs. Having this is a an organized way to keep track of the player's workouts and what they have done. This would then later be displayed if the player would like to see their progress.

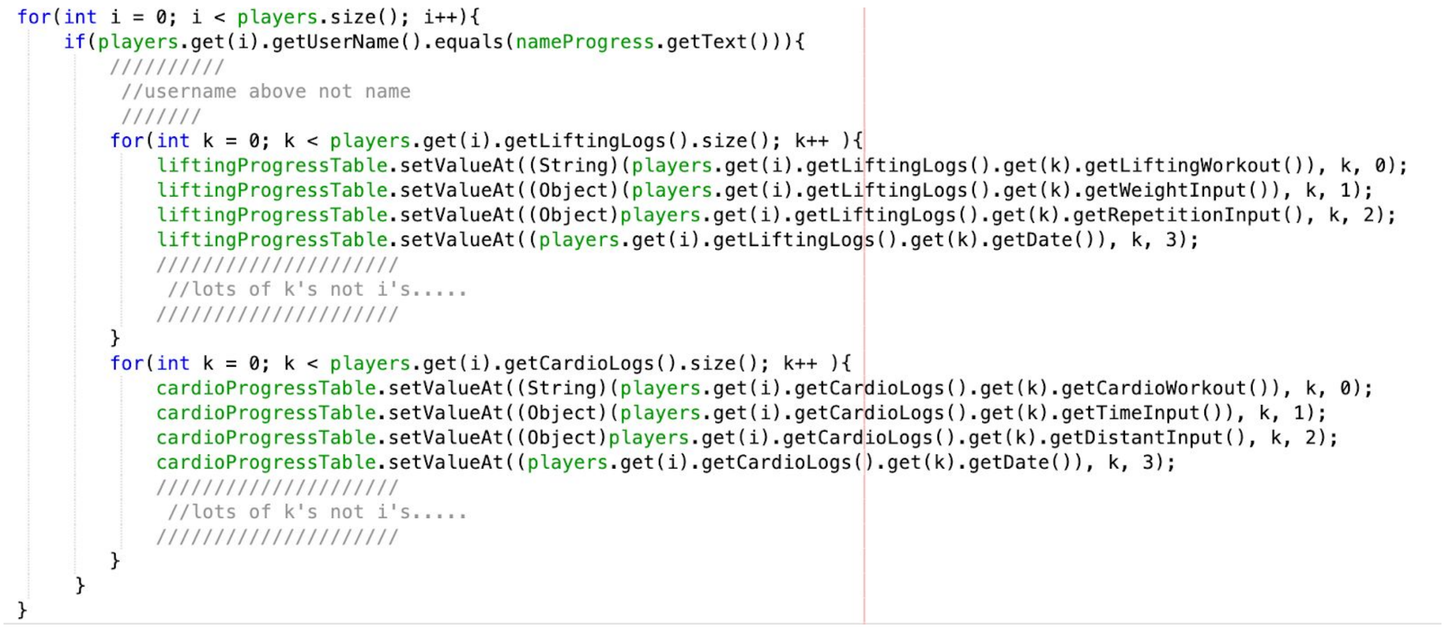


```
   String date = String.valueOf(dayCardio.getSelectedItem())
           + monthCardio.getSelectedItem()
           + yearCardio.getSelectedItem();
   String name1 = nameCardioTF.getText();
   String workouts = String.valueOf(CardioType.getSelectedItem());
   //distant and time
   int seconds = Integer.parseInt(secondsinput.getText());
   double distant = Double.parseDouble(distantInput.getText());
   CardioLogs log = new CardioLogs(date, name1, workouts, seconds, distant);
   System.out.println(log.getDistantInput());

   for(int i = 0; i < players.size(); i++){
     if(players.get(i).getUserName().equals(nameCardioTF.getText())){
        ///////////
        //getUserName above not userName
        //////////
        ArrayList<CardioLogs> b = players.get(i).getCardioLogs();
              b.add(log);
        System.out.println(players.get(i).getCardioLogs().get(0).getTimeInput());
   }
}
```

This is the exact same as the LiftingLogs, however, it’s just the variable that has been changed. The variables are date, name, workouts, seconds and distance instead.

3. Displaying in Tables

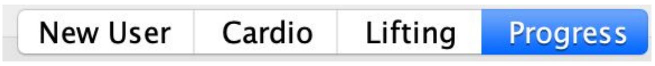


```
for(int i = 0; i < players.size(); i++){
    if(players.get(i).getUserName().equals(nameProgress.getText())){
        //////////
         //username above not name
         ///////
        for(int k = 0; k < players.get(i).getLiftingLogs().size(); k++ ){
            liftingProgressTable.setValueAt((String)(players.get(i).getLiftingLogs().get(k).getLiftingWorkout()), k, 0);
            liftingProgressTable.setValueAt((Object)(players.get(i).getLiftingLogs().get(k).getWeightInput()), k, 1);
            liftingProgressTable.setValueAt((Object)players.get(i).getLiftingLogs().get(k).getRepetitionInput(), k, 2);
            liftingProgressTable.setValueAt((players.get(i).getLiftingLogs().get(k).getDate()), k, 3);
            ////////////////////
             //lots of k's not i's.....
            ////////////////////
        }
        for(int k = 0; k < players.get(i).getCardioLogs().size(); k++ ){
            cardioProgressTable.setValueAt((String)(players.get(i).getCardioLogs().get(k).getCardioWorkout()), k, 0);
            cardioProgressTable.setValueAt((Object)(players.get(i).getCardioLogs().get(k).getTimeInput()), k, 1);
            cardioProgressTable.setValueAt((Object)players.get(i).getCardioLogs().get(k).getDistantInput(), k, 2);
            cardioProgressTable.setValueAt((players.get(i).getCardioLogs().get(k).getDate()), k, 3);
            ////////////////////
             //lots of k's not i's.....
            ////////////////////
        }
    }
}
```

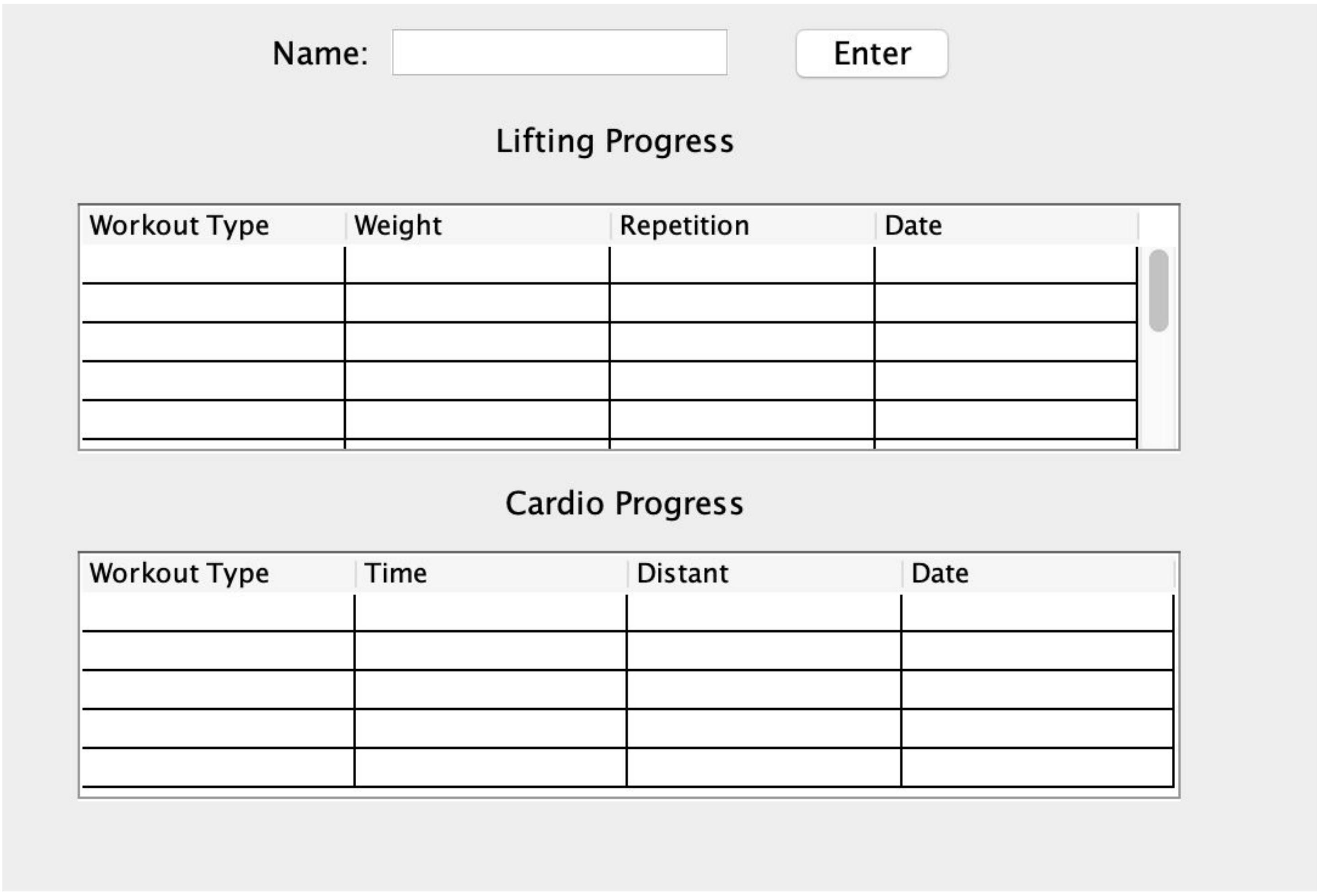
The point of this code is to search for the name that the user has inputted. Then it would look for any input that the user has put in previously. By having this it would display all the user’s data. To get the data it would have to go to the cardio or lifting logs class and then it would then have to search for specifics data, such as weight or time.

Word Count: 334

6. User Interface/GUI Work:



The tabbed panel is put in to make it convenient for the user to easily input data and seeing their output, it is also easier for them to maneuver around the program.


Name: Enter

Lifting Progress

Workout Type	Weight	Repetition	Date

Cardio Progress

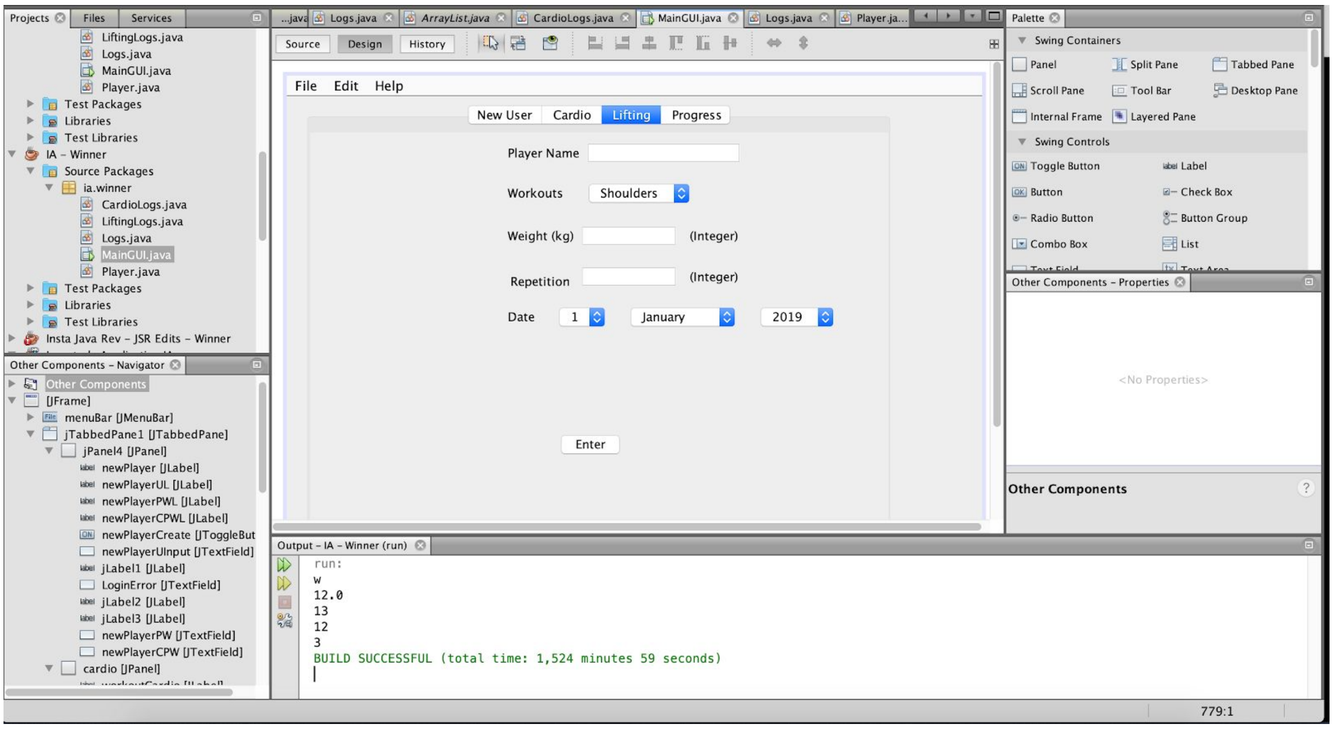
Workout Type	Time	Distant	Date

There are labels here, to labels for the user so they know what they are inputting into the text field. The text field is made for the user to put in their data. There are also buttons to allow the user to click once they have inputted their data into the text field, the button allows the code to run. Along with that, there are display tables to allow the user to see their outputs.

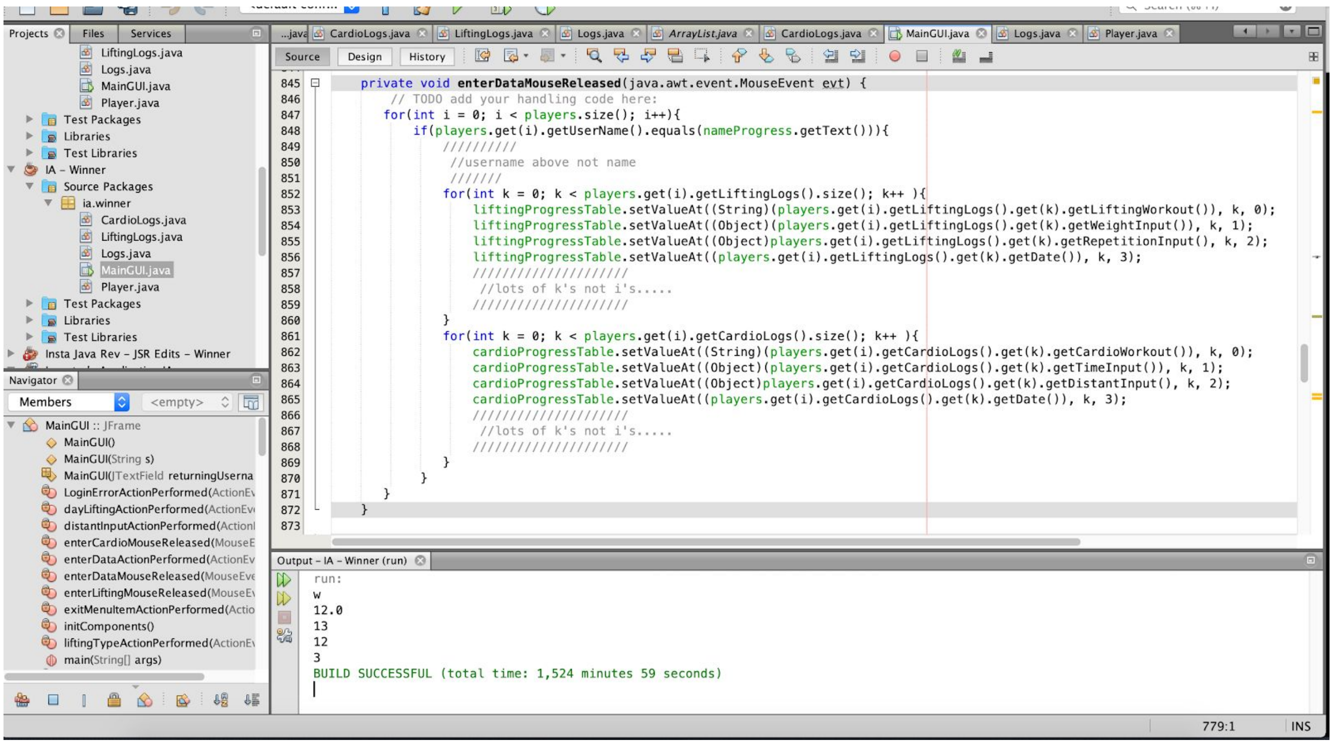
The Combo Box allows the users to choose what the program has set out for them, this would minimize mistakes, such as dates, there might be a typo such as 32 December 2023.

Word Count: 140

7. Software Tools Used



```
run:
w
12.0
13
12
3
BUILD SUCCESSFUL (total time: 1,524 minutes 59 seconds)
```



```java
    private void enterDataMouseReleased(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
       for(int i = 0; i < players.size(); i++){
           if(players.get(i).getUserName().equals(nameProgress.getText())){
               //////////
                //username above not name
                ///////
               for(int k = 0; k < players.get(i).getLiftingLogs().size(); k++ ){
                   liftingProgressTable.setValueAt((String)(players.get(i).getLiftingLogs().get(k).getLiftingWorkout()), k, 0);
                   liftingProgressTable.setValueAt((Object)(players.get(i).getLiftingLogs().get(k).getWeightInput()), k, 1);
                   liftingProgressTable.setValueAt((Object)players.get(i).getLiftingLogs().get(k).getRepetitionInput(), k, 2);
                   liftingProgressTable.setValueAt((players.get(i).getLiftingLogs().get(k).getDate()), k, 3);
                   /////////////////////
                    //lots of k's not i's.....
                   /////////////////////
               }
               for(int k = 0; k < players.get(i).getCardioLogs().size(); k++ ){
                   cardioProgressTable.setValueAt((String)(players.get(i).getCardioLogs().get(k).getCardioWorkout()), k, 0);
                   cardioProgressTable.setValueAt((Object)(players.get(i).getCardioLogs().get(k).getTimeInput()), k, 1);
                   cardioProgressTable.setValueAt((Object)players.get(i).getCardioLogs().get(k).getDistantInput(), k, 2);
                   cardioProgressTable.setValueAt((players.get(i).getCardioLogs().get(k).getDate()), k, 3);
                   /////////////////////
                    //lots of k's not i's.....
                   /////////////////////
               }
           }
       }
    }
```

```
run:
w
12.0
13
12
3
BUILD SUCCESSFUL (total time: 1,524 minutes 59 seconds)
```

NetBeans was chosen because of the pre-library code, the language used is Java and it's GUI friendly. Along with that, it could easily be used with object-oriented programming, which allows it to be easier for the programmer.

Word Count: 37

Full Document Word Count: 925