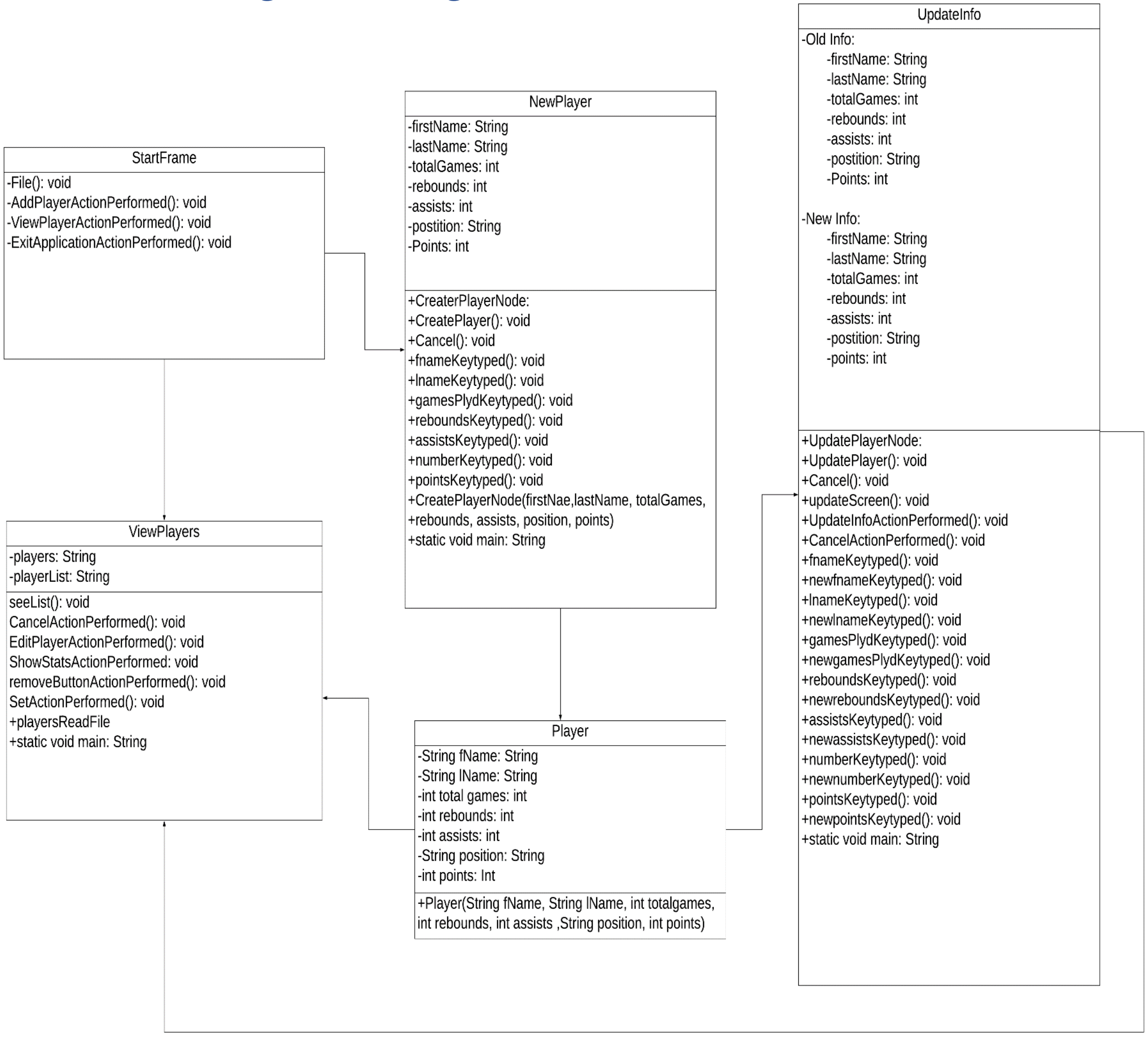
Criterion C: Development

UML Diagram of Program



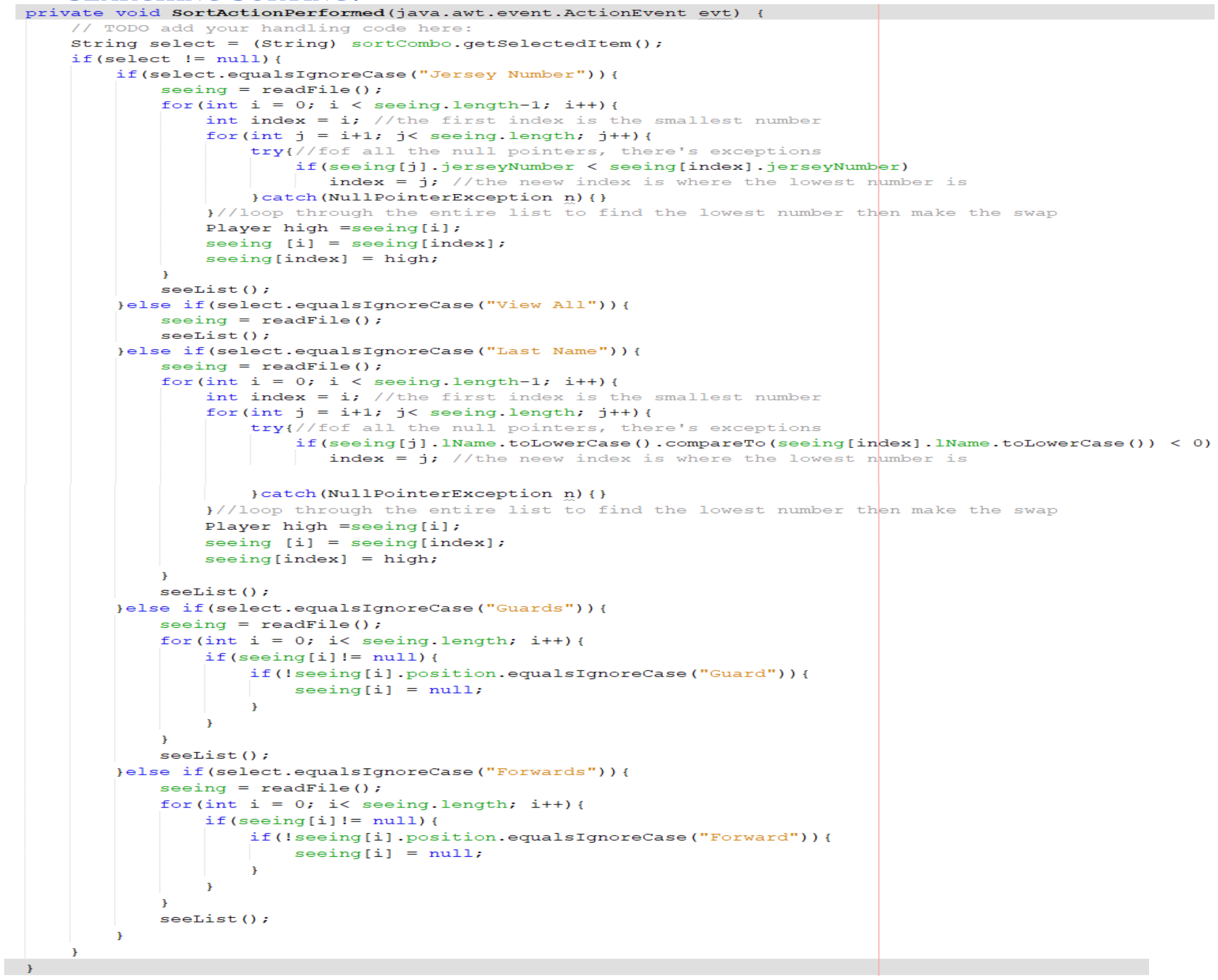
Complex Code Explanations and Techniques

List of techniques used in the program:

1. Searching and Sorting
2. Trycatch
3. Complex Loop
4. File Handling
5. Polymorphism
6. Inheritance
7. Graphical User Interface

Explanation of code with screenshots of code:

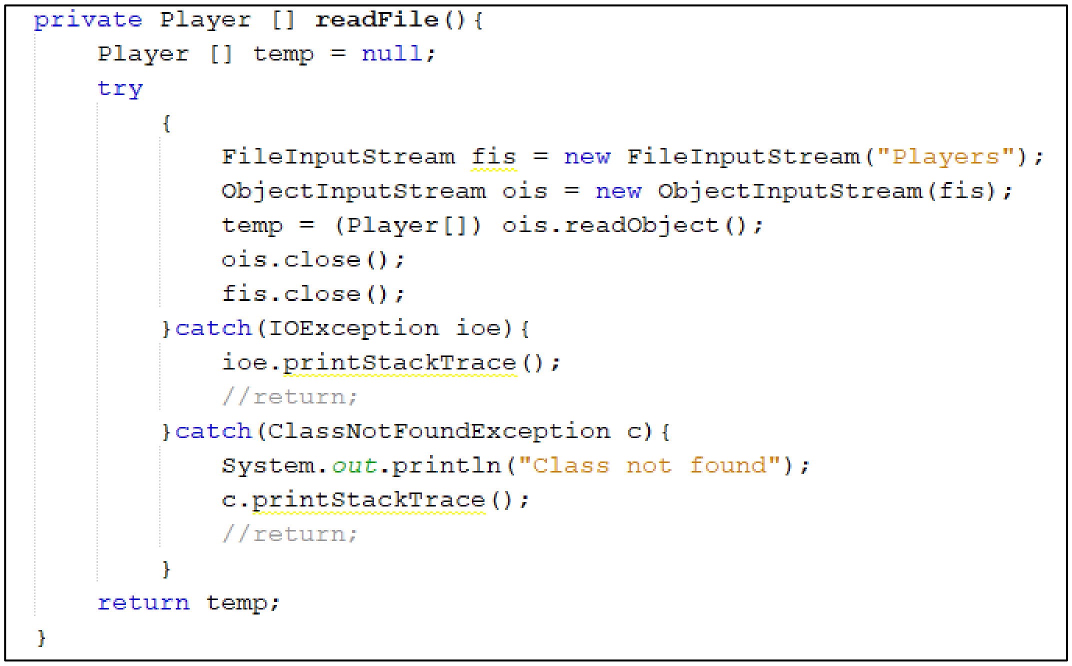
Searching Sorting:



```java
private void SortActionPerformed(java.awt.event.ActionEvent evt)  {
    // TODO add your handling code here:
    String select = (String) sortCombo.getSelectedItem();
    if(select != null){
        if(select.equalsIgnoreCase("Jersey Number")){
            seeing = readFile();
            for(int i = 0; i < seeing.length-1; i++){
                int index = i; //the first index is the smallest number
                for(int j = i+1; j< seeing.length; j++){
                    try{//fof all the null pointers, there's exceptions
                        if(seeing[j].jerseyNumber < seeing[index].jerseyNumber)
                            index = j; //the neew index is where the lowest number is
                    }catch(NullPointerException n){}
                }//loop through the entire list to find the lowest number then make the swap
                Player high =seeing[i];
                seeing [i] = seeing[index];
                seeing[index] = high;
            }
            seeList();
        }else if(select.equalsIgnoreCase("View All")){
            seeing = readFile();
            seeList();
        }else if(select.equalsIgnoreCase("Last Name")){
            seeing = readFile();
            for(int i = 0; i < seeing.length-1; i++){
                int index = i; //the first index is the smallest number
                for(int j = i+1; j< seeing.length; j++){
                    try{//fof all the null pointers, there's exceptions
                        if(seeing[j].lName.toLowerCase().compareTo(seeing[index].lName.toLowerCase()) < 0)
                            index = j; //the neew index is where the lowest number is

                    }catch(NullPointerException n){}
                }//loop through the entire list to find the lowest number then make the swap
                Player high =seeing[i];
                seeing [i] = seeing[index];
                seeing[index] = high;
            }
            seeList();
        }else if(select.equalsIgnoreCase("Guards")){
            seeing = readFile();
            for(int i = 0; i< seeing.length; i++){
                if(seeing[i]!= null){
                    if(!seeing[i].position.equalsIgnoreCase("Guard")){
                        seeing[i] = null;
                    }
                }
            }
            seeList();
        }else if(select.equalsIgnoreCase("Forwards")){
            seeing = readFile();
            for(int i = 0; i< seeing.length; i++){
                if(seeing[i]!= null){
                    if(!seeing[i].position.equalsIgnoreCase("Forward")){
                        seeing[i] = null;
                    }
                }
            }
            seeList();
        }
    }
}
```

In the sorting action performed method, it allows the user to sort it into different ways. In the code above, for instance to sort the players by their position, specifically the guard position, the if –else statement will be read from another method from the readFile method. An else if statement is used in a program and it is a statement that is used after an if statement performs a true function. The for loop is traversed through the array and then it is put through an else –if statement.Else –if statements are used to create additional statements and are used when there are more than one decision to choose from. If the if statement is false, then it will go to the next else statement and will go through all of them until one decision is chosen. In the code above, it also shows a search method. The search method in the code above for an example is for the search for the players for the “Forward” position. It will search through the position variable and will show null for players with the position of a guard, and will only visibly show players with the position of “forwards” and this is vice versa for the guards position.

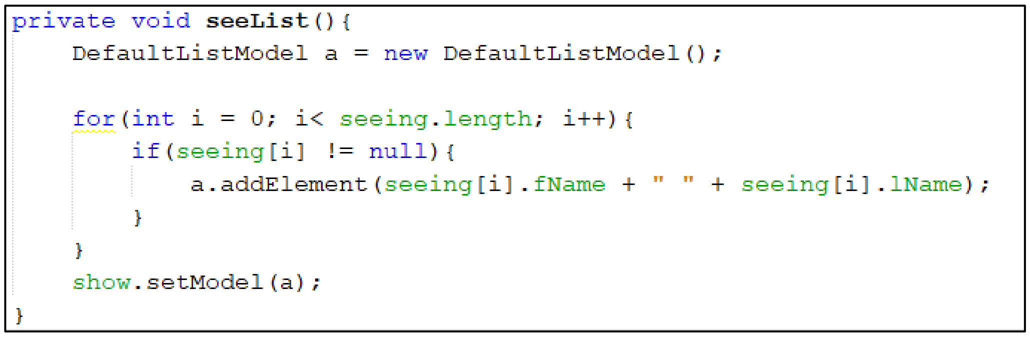
TRY CATCH:



```
private Player [] readFile(){
    Player [] temp = null;
    try
        {
            FileInputStream fis = new FileInputStream("Players");
            ObjectInputStream ois = new ObjectInputStream(fis);
            temp = (Player[]) ois.readObject();
            ois.close();
            fis.close();
        }catch(IOException ioe){
            ioe.printStackTrace();
            //return;
        }catch(ClassNotFoundException c){
            System.out.println("Class not found");
            c.printStackTrace();
            //return;
        }
    return temp;
}
```

In the Stats Manager, there are many error handles in the program that catches any file errors or runtime errors. Try catch is also known as exception handling. This is used within a method and a try statement is used to enclose code that may throw an exception and it is then followed by a catch statement which will then catch the exception. In the screenshot above, it shows if the file is found than the first catch statement will go through and will display what was chosen to be displayed. If the file is not found, the second catch statement wile catch the exception and then display a message that the class was not found.

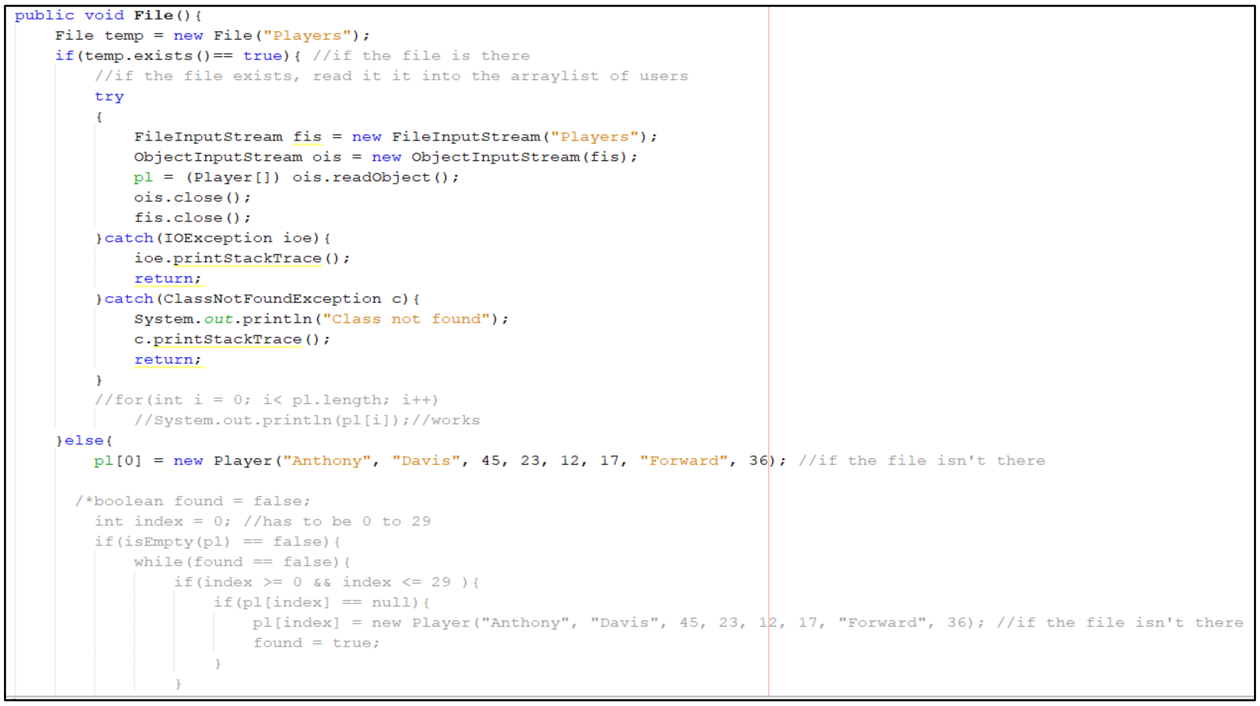
COMPLEX LOOP:



```java
private void seeList(){
    DefaultListModel a = new DefaultListModel();

    for(int i = 0; i< seeing.length; i++){
        if(seeing[i] != null){
            a.addElement(seeing[i].fName + " " + seeing[i].lName);
        }
    }
    show.setModel(a);
}
```

This is a technique of a complex loop. In the complex loop, there is a for loop which contains an if statement with in it. When player are created, they are put into a list and in the if statement, in the for loop, it will see how many players were created and will then display how many players there are. The for loop traverses through the array and then prints out every players created.

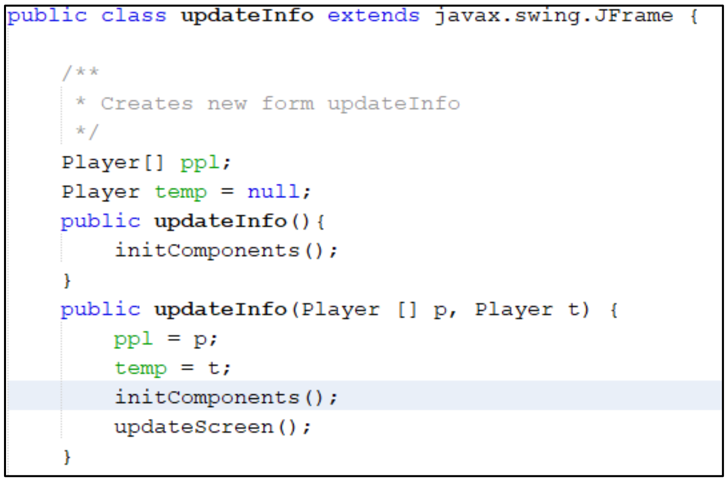
FILE HANDLING:



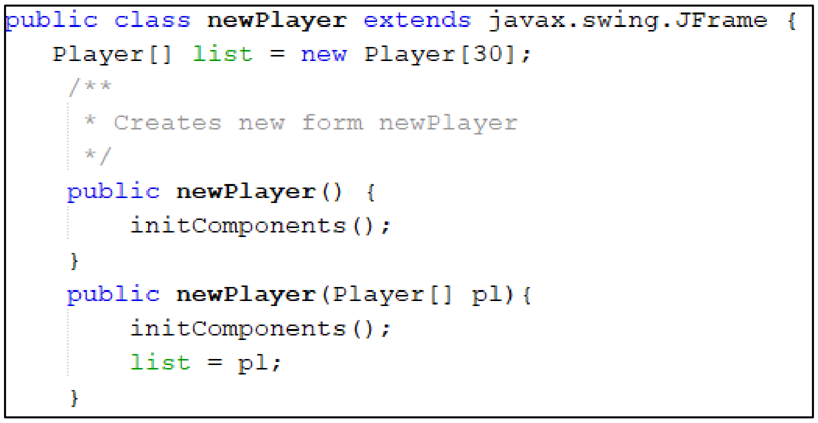
```java
public void File(){
    File temp = new File("Players");
    if(temp.exists()== true){ //if the file is there
        //if the file exists, read it it into the arraylist of users
        try
        {
            FileInputStream fis = new FileInputStream("Players");
            ObjectInputStream ois = new ObjectInputStream(fis);
            pl = (Player[]) ois.readObject();
            ois.close();
            fis.close();
        }catch(IOException ioe){
            ioe.printStackTrace();
            return;
        }catch(ClassNotFoundException c){
            System.out.println("Class not found");
            c.printStackTrace();
            return;
        }
        //for(int i = 0; i< pl.length; i++)
            //System.out.println(pl[i]);//works
    }else{
        pl[0] = new Player("Anthony", "Davis", 45, 23, 12, 17, "Forward", 36); //if the file isn't there

      /*boolean found = false;
        int index = 0; //has to be 0 to 29
        if(isEmpty(pl) == false){
            while(found == false){
                if(index >= 0 && index <= 29 ){
                    if(pl[index] == null){
                        pl[index] = new Player("Anthony", "Davis", 45, 23, 12, 17, "Forward", 36); //if the file isn't there
                        found = true;
                    }
                }
```

File handling is having inputs and outputs from a file. Having an output to a file, will save information from the player's class which is then placed in a try catch. In the try catch, when the information is saved into the players file, if any exception is throw, it will be handled by the catch statement. Once exiting the program, all the information will be stored in the file, "Player" and once return back into the program. The outputs that were saved in the player's file will then be inputted back into the program, where the user can still view previous information.

POLYMORPHISM:



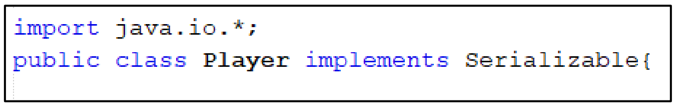
```
public class updateInfo extends javax.swing.JFrame {

    /**
     * Creates new form updateInfo
     */
    Player[] ppl;
    Player temp = null;
    public updateInfo(){
        initComponents();
    }
    public updateInfo(Player [] p, Player t) {
        ppl = p;
        temp = t;
        initComponents();
        updateScreen();
    }
```



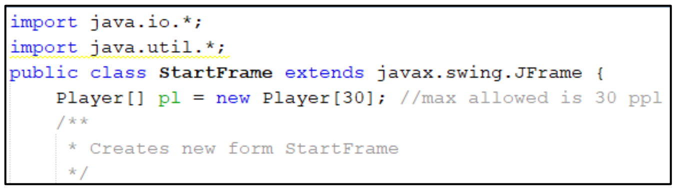
```
public class newPlayer extends javax.swing.JFrame {
    Player[] list = new Player[30];
    /**
     * Creates new form newPlayer
     */
    public newPlayer() {
        initComponents();
    }
    public newPlayer(Player[] pl){
        initComponents();
        list = pl;
    }
```

This is the technique of polymorphism, it is used when there is two methods which are called the same name which in the left image is called updateInfo, but they hold different parameters. The difference between these two methods is the parameters within the methods. One method is to store and update the file in its components and the other method is to update certain information that the user changed.

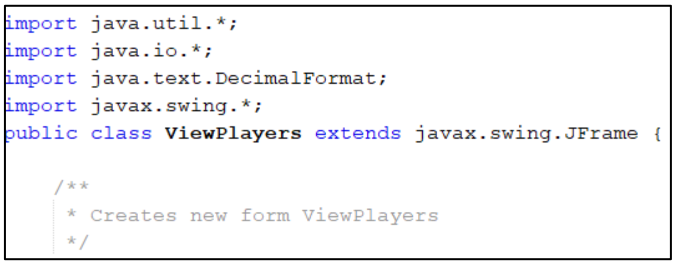
INHERITANCE:



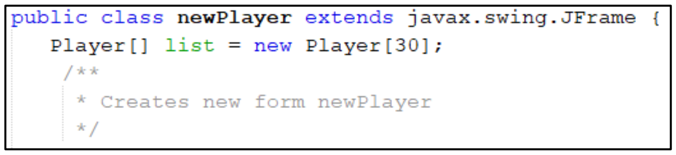
```
import java.io.*;
public class Player implements Serializable{
```



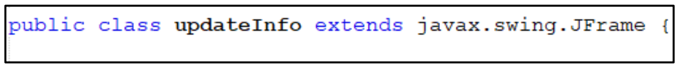
```
import java.io.*;
import java.util.*;
public class StartFrame extends javax.swing.JFrame {
    Player[] pl = new Player[30]; //max allowed is 30 ppl
    /**
     * Creates new form StartFrame
     */
```



```
import java.util.*;
import java.io.*;
import java.text.DecimalFormat;
import javax.swing.*;
public class ViewPlayers extends javax.swing.JFrame {

    /**
     * Creates new form ViewPlayers
     */
```



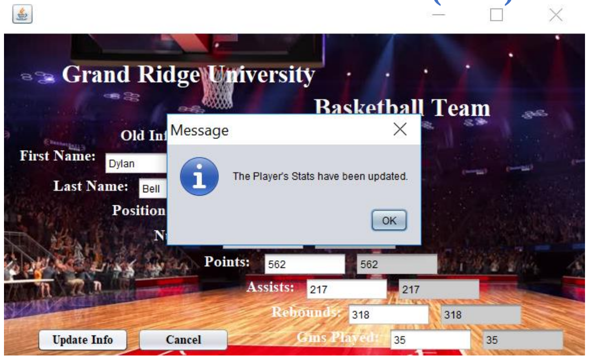
```
public class newPlayer extends javax.swing.JFrame {
    Player[] list = new Player[30];
    /**
     * Creates new form newPlayer
     */
```



```
public class updateInfo extends javax.swing.JFrame {
```

This technique is inheritance. Inheritance is a feature that lets you create a classes that are derived from other classes. This is where there is one super class and there are multiple sub classes within it. In the code above, all the methods such as startFrame, newPlayer, updateInfo, and viewPlayers are all sub classes which extend to javax.swing.Jframe. “Extending” a class means that you are making a new class that is derived from a class that already exists. Extending a class means that you are increasing the functions of that class. As you can see in the top right screenshot, the public class startFrame extends to java.swing.Jframe which means that the startFrame class is being derived into another class which is java.swingJframe.

Graphical User Interface (GUI):

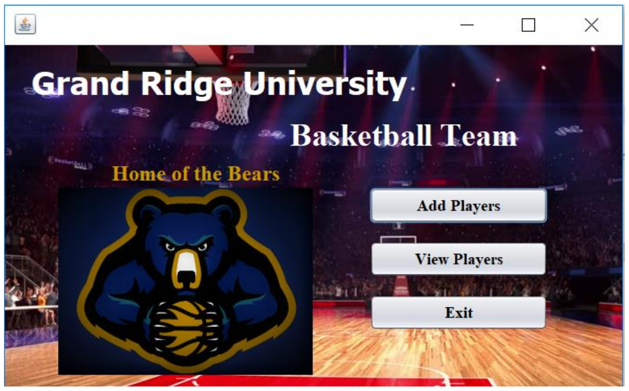


Message

The Player's Stats have been updated.

OK

First Name: Dylan
Last Name: Bell
Points: 562 562
Assists: 217 217
Rebounds: 318 318
Gms Played: 35 35



```
private void initComponents() {

    GRU = new javax.swing.JLabel();
    ViewPlayers = new javax.swing.JButton();
    AddPlayer = new javax.swing.JButton();
    Exit = new javax.swing.JButton();
    BBallTeam = new javax.swing.JLabel();
    Logo = new javax.swing.JLabel();
    HomeOfTheBears = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
```

The graphical user interface is a very important and useful part of the program. GUI makes it easier for the use to handle and go through all the features of the program. GUI makes it very useful for the user to guide through the program to successfully meet the success criteria. GUI helps the user preventing any mistakes as if any mistakes or errors were made, an error handling massage will pop up telling the user that something has occurred. A dialogue box may pop up to show that something has occurred or that something successfully has been done or created.

Word Count: 861