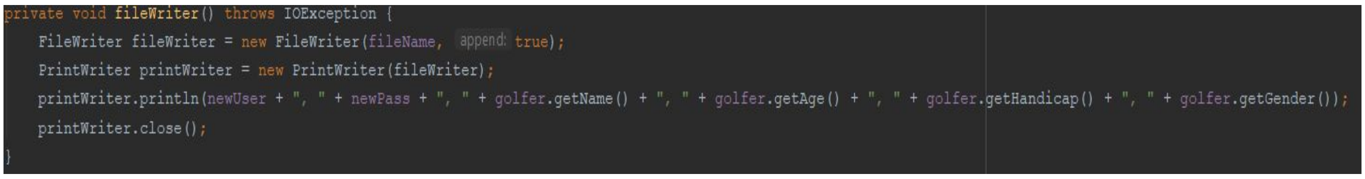
CRITERION C

Complex Code

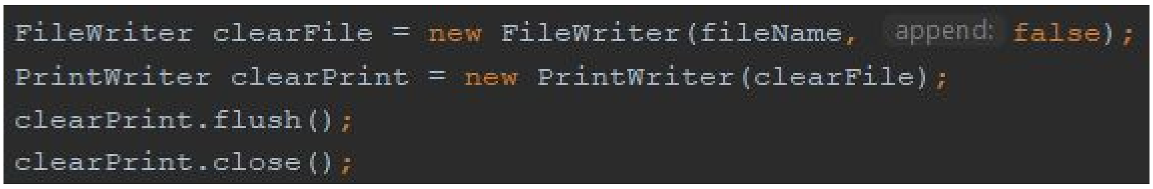
File Handling

The program uses file handling to store users details.



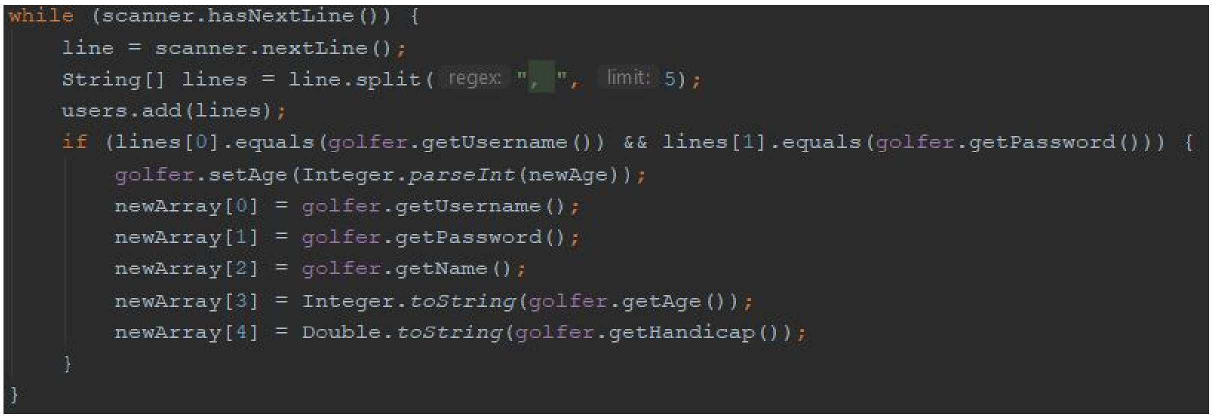
```
private void fileWriter() throws IOException {
    FileWriter fileWriter = new FileWriter(fileName,  append: true);
    PrintWriter printWriter = new PrintWriter(fileWriter);
    printWriter.println(newUser + ", " + newPass + ", " + golfer.getName() + ", " + golfer.getAge() + ", " + golfer.getHandicap() + ", " + golfer.getGender());
    printWriter.close();
}
```

The code above is a method that writes the users details to the file using the golfer object. The code uses imported libraries FileWriter, PrintWriter, FileReader, and Scanner as well as IOException to find the text file and write to it.



```
FileWriter clearFile = new FileWriter(fileName,  append: false);
PrintWriter clearPrint = new PrintWriter(clearFile);
clearPrint.flush();
clearPrint.close();
```

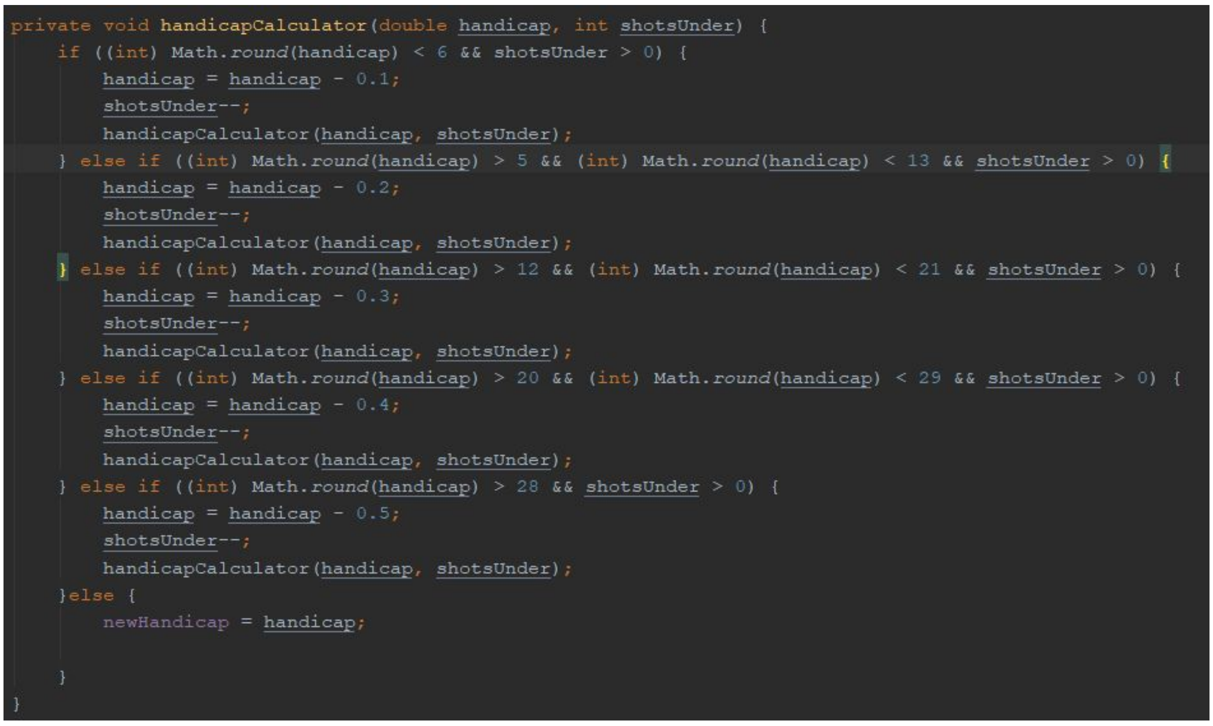
This part of code clears the file so that it is ready for the edited details of the users to be printed back to the file



```
while (scanner.hasNextLine()) {
    line = scanner.nextLine();
    String[] lines = line.split( regex: ", ",  limit: 5);
    users.add(lines);
    if (lines[0].equals(golfer.getUsername()) && lines[1].equals(golfer.getPassword())) {
        golfer.setAge(Integer.parseInt(newAge));
        newArray[0] = golfer.getUsername();
        newArray[1] = golfer.getPassword();
        newArray[2] = golfer.getName();
        newArray[3] = Integer.toString(golfer.getAge());
        newArray[4] = Double.toString(golfer.getHandicap());
    }
}
```

This part of the code saves the new details so that they are ready to be reprinted back to the files. It does this by reading the file to check if the details entered by the user are matched by the ones stored in the file. Similar code is used during the login process

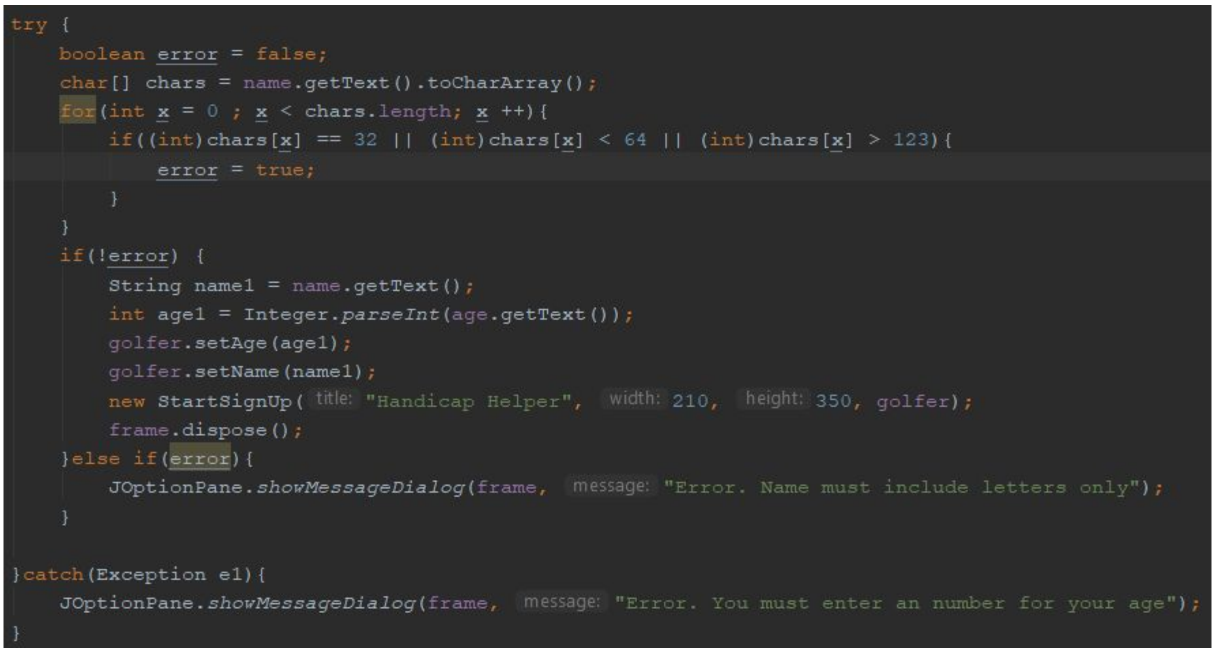
Recursion



```java
private void handicapCalculator(double handicap, int shotsUnder) {
    if ((int) Math.round(handicap) < 6 && shotsUnder > 0) {
        handicap = handicap - 0.1;
        shotsUnder--;
        handicapCalculator(handicap, shotsUnder);
    } else if ((int) Math.round(handicap) > 5 && (int) Math.round(handicap) < 13 && shotsUnder > 0) {
        handicap = handicap - 0.2;
        shotsUnder--;
        handicapCalculator(handicap, shotsUnder);
    } else if ((int) Math.round(handicap) > 12 && (int) Math.round(handicap) < 21 && shotsUnder > 0) {
        handicap = handicap - 0.3;
        shotsUnder--;
        handicapCalculator(handicap, shotsUnder);
    } else if ((int) Math.round(handicap) > 20 && (int) Math.round(handicap) < 29 && shotsUnder > 0) {
        handicap = handicap - 0.4;
        shotsUnder--;
        handicapCalculator(handicap, shotsUnder);
    } else if ((int) Math.round(handicap) > 28 && shotsUnder > 0) {
        handicap = handicap - 0.5;
        shotsUnder--;
        handicapCalculator(handicap, shotsUnder);
    }else {
        newHandicap = handicap;

    }
}
```

This piece of code is a method that calculates the new handicap of the user recursively. It uses the base case ‘shotsUnder’ which is decremented every time before the program recurses. When ‘shotsUnder’ reaches 0 the recursion stops. Also, each recursion depends on the category of the golfer - the higher the category the more shots reducted from the handicap value. Additionally, the recursion is tail-ended so it avoids a stack overflow error.

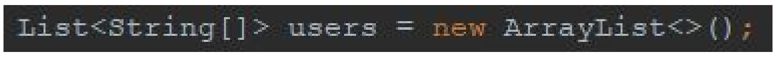
Try Catch Statements



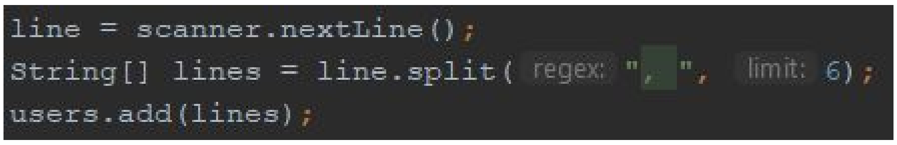
```java
try {
    boolean error = false;
    char[] chars = name.getText().toCharArray();
    for(int x = 0 ; x < chars.length; x ++){
        if((int)chars[x] == 32 || (int)chars[x] < 64 || (int)chars[x] > 123){
            error = true;
        }
    }
    if(!error) {
        String name1 = name.getText();
        int age1 = Integer.parseInt(age.getText());
        golfer.setAge(age1);
        golfer.setName(name1);
        new StartSignUp( title: "Handicap Helper", width: 210, height: 350, golfer);
        frame.dispose();
    }else if(error){
        JOptionPane.showMessageDialog(frame, message: "Error. Name must include letters only");
    }

}catch(Exception e1){
    JOptionPane.showMessageDialog(frame, message: "Error. You must enter an number for your age");
}
```

This part of the code is a try-catch statement that catches an exception if the user enters a String for their age. The code catches the exception and returns the error message saying that the user must enter a number for their age.

ArrayList of Arrays



```java
List<String[]> users = new ArrayList<>();
```

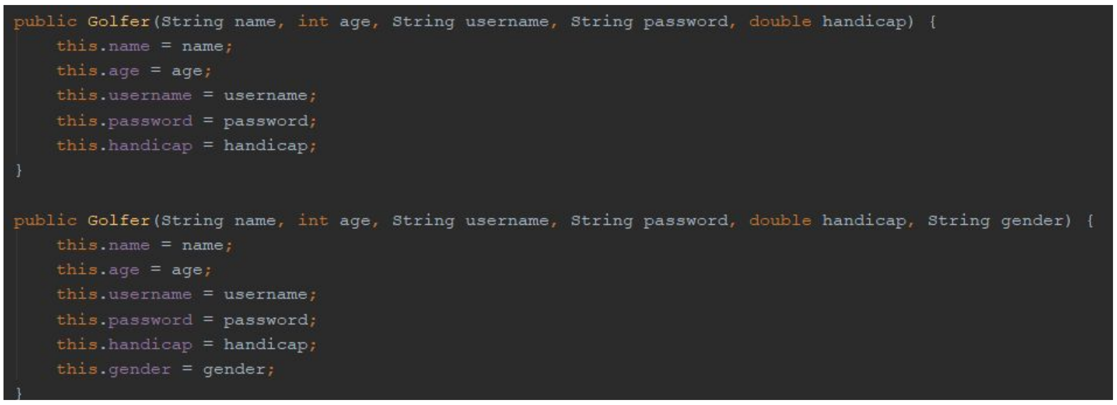


```java
line = scanner.nextLine();
String[] lines = line.split( regex: ", ", limit: 6);
users.add(lines);
```

This part of the code is used whenever the file is being edited. The user's details are temporarily stored in an ArrayList of arrays where it can be edited and then reprinted to the file. This is the best possible data structure for the program as it allows for indexing therefore it is better than other structures such as a LinkedList. By having indexing it means the program will have direct access to any node in the list rather than having to step through.

OOP:

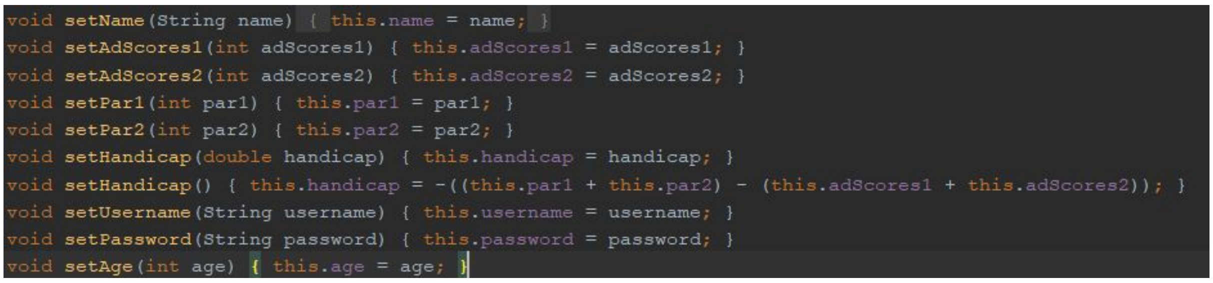
Polymorphism



```java
public Golfer(String name, int age, String username, String password, double handicap) {
    this.name = name;
    this.age = age;
    this.username = username;
    this.password = password;
    this.handicap = handicap;
}

public Golfer(String name, int age, String username, String password, double handicap, String gender) {
    this.name = name;
    this.age = age;
    this.username = username;
    this.password = password;
    this.handicap = handicap;
    this.gender = gender;
}
```

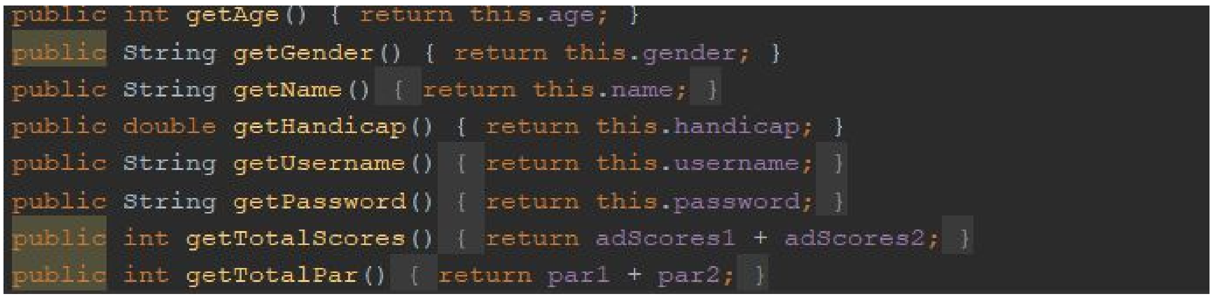
This part of the code is in the golfer class and means that golfer can be instantiated in more than one way. By having methods with different parameters, method overloading occurs meaning that the Golfer can be instantiated in different ways, whether it be through logging in or signing up. This is good because if the user decides to log in the object will still be instantiated even though it has different parameters to signing up, so depending what the user does, the object is still instantiated. (This is an example of constructor overloading)

Getters and Setters



```java
void setName(String name) { this.name = name; }
void setAdScores1(int adScores1) { this.adScores1 = adScores1; }
void setAdScores2(int adScores2) { this.adScores2 = adScores2; }
void setPar1(int par1) { this.par1 = par1; }
void setPar2(int par2) { this.par2 = par2; }
void setHandicap(double handicap) { this.handicap = handicap; }
void setHandicap() { this.handicap = -((this.par1 + this.par2) - (this.adScores1 + this.adScores2)); }
void setUsername(String username) { this.username = username; }
void setPassword(String password) { this.password = password; }
void setAge(int age) { this.age = age; }
```

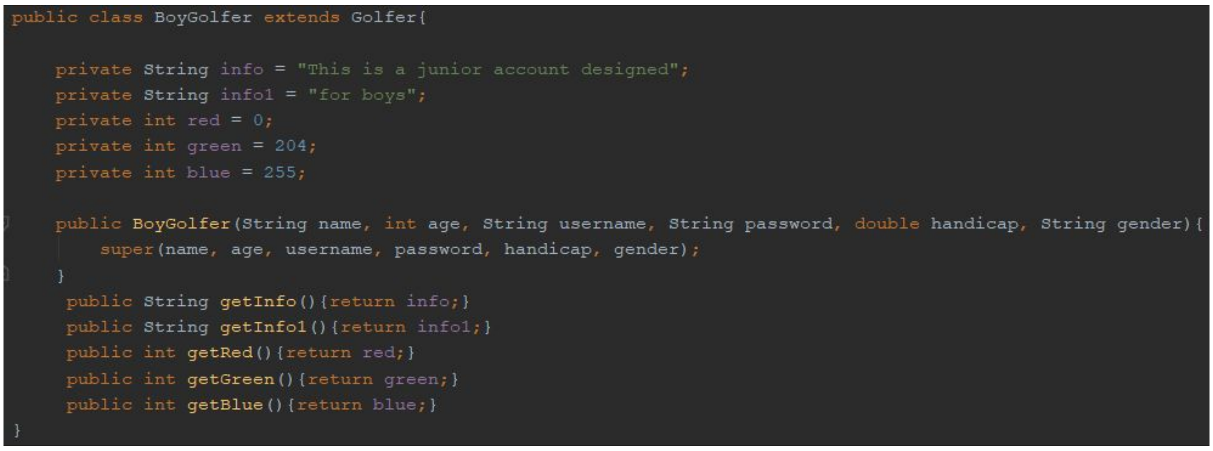
This part of the code is in the Golfer object. These are called from other parts of the code to save different aspects of the user's account. For example if setName("Name") is called from another class the name variable in the golfer class will be set to "Name".



```
public int getAge() { return this.age; }
public String getGender() { return this.gender; }
public String getName() { return this.name; }
public double getHandicap() { return this.handicap; }
public String getUsername() { return this.username; }
public String getPassword() { return this.password; }
public int getTotalScores() { return adScores1 + adScores2; }
public int getTotalPar() { return par1 + par2; }
```

This part of the code is also in the Golfer object class and if called will return something. For example, in the Homepage GUI, ‘getUsername()’ is called to check the username entered against the one stored in the golfer class.

Inheritance



```
public class BoyGolfer extends Golfer{

    private String info = "This is a junior account designed";
    private String info1 = "for boys";
    private int red = 0;
    private int green = 204;
    private int blue = 255;

    public BoyGolfer(String name, int age, String username, String password, double handicap, String gender){
        super(name, age, username, password, handicap, gender);
    }
     public String getInfo(){return info;}
     public String getInfo1(){return info1;}
     public int getRed(){return red;}
     public int getGreen(){return green;}
     public int getBlue(){return blue;}
}
```

This part of the code is inherited from the Golfer object class using the keyword extends. It has the same functions as Golfer but with some additional fields and classes. For example, it has red, green and blue attributes which when called using the getters can be used to change the colour of the Homepage JPanel. This is good because it allows for some specialisation for a Boy or Girl golfer who would be attracted to this type of feature.

581 words

<u>Sources for code:</u>

ArrayList (Java Platform SE 8), 6 Jan. 2020, docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html.

Chan, Jamie. *Learn Java in One Day and Learn It Well: Java for Beginners with Hands-on Project: the Only Book You Need to Start Coding in Java Immediately*. Publisher Not Identified, 2018.

Cho, James S. *The Beginner's Guide to Android Game Development*. Glasnevin Publishing, 2014.

"File Handling in Java Using FileWriter and FileReader." *GeeksforGeeks*, 3 Jan. 2019, www.geeksforgeeks.org/file-handling-java-using-filewriter-filereader/.

"Java Exceptions - Try...Catch." *Java Exceptions (Try...Catch)*, www.w3schools.com/java/java_try_catch.asp.

SumithraSumithra