CS5 Software Development – Philip Mortimer

Prototype and Post-Prototype Refinement of Design Summary

Both the prototype and post-prototype refinement of design have been removed from the coursework. However, I still completed these sections. My design was altered slightly with my client requesting a few changes. First – the client requested that club members do not actually play the fixtures using the software. This was because my client felt that face-to-face chess may become possible in the near future. He requested that I create some tool that allows him to generate fixtures and keep track / set the outcome of these fixtures. This means that the club leader is also responsible for starting a new season once the old one has been completed. I sought clarification on how Lewis James handled new members joining the club midway through the season. He stated that they essentially play friendly games against other late joiners. He said that the system does not need to handle this and should only worry about generating fixtures at the start of the season. I also stated that I was looking to tailor the system towards Windows devices as I planned on using a chess engine that only runs on Windows for one of the computer opponents. My client (Lewis James) confirmed that this was not an issue as club member only owned Windows computers. These were the only notable changes. Apart from these things, the feedback on my proposed system and prototype was very positive. Following this process, a slightly updated set of aims and objectives was agreed upon. Here are those updated aims and objectives:

1. The system will allow users to log in by providing an email address, username, and password. Both the email address and username must be unique.
    a. To create a database that stores all relevant details, including user details, in the third normal form.
        i. All data stored must have no repeating groups, partial dependencies or transitive dependencies.
        ii. Users will be able to log in to the system by providing their email address / username and password. Once logged in, they will be able to view their details such as their ranking or fixtures (if they are a member of the club).
    b. To check that both the email address and username provided are not already the username or email address of another user. This will be done by searching the file storing all user details.
        i. When attempting to create an account, users who input the email address or username of another user will be unable to use that name and will be asked to choose a different one.
        ii. When creating an account, a username / email address that has not already been used by another user should be accepted.
    c. To search both user email address and username to find a match when logging in.
        i. Users should not be able to create a username that could be a valid email address. This will be achieved by not allowing @ character in usernames and requiring the @ character in email addresses.
        ii. Users will be able to log-in by providing either their email address or username.
2. Players may choose to play as guests who require no passwords but will have more limited features available to them. Guests will be unable to access features such as the club section but will be able to access offline play.
    a. When a guest uses the system, they will be given one id that all guests have. This id will be impossible for a normal user to have. With this guest id, the guest will have

the ability to play offline games but will not have a ranking. Hence, the system will have four access levels (for guests, users, club members and the club leader).

i. A guest will be able to play offline but will not be able to access the club section.

3. There will be an option for users to play online against random people or against friends. People can play against friends by typing in their friend's username or by sharing a random 6-digit code generated by the system. Users will be able to customise the ruleset for online play to allow for correspondence play and playing on a timer.
    a. The server will have a text file storing all currently ongoing online games. Each time a new game is created, a game id will be generated for it. This will be the six-digit code that users can share. Furthermore, it will store details such as the board state and the user id of players, as well as the ruleset.
        i. Users will be able to play against friends either by sharing a six-digit code or by typing their username. The ruleset for these games will be customisable.
    b. Players will be able to play against random people by pressing a button. They will be matched against another user who also wishes to play online against a random opponent. This will be achieved by requesting to play a random game. The game will then be created when another user makes such a request.
        i. Users will be able to play against random opponents.
4. Users will be able to play chess locally against each other with users using the same computer to play against each other.
    a. A file containing all ongoing offline games will be stored. This will store up to three current games. When a file is clicked on, the game is resumed.
        i. Users will be able to play up to three offline games at a time over multiple session.
    b. Users will be able to play offline and customise the time controls. Each offline game will be given a game id.
        i. When creating a game, users can choose which time control they would like to use. This time control will then be used for the game. This includes the option for no time restrictions.
5. Users will be able to play against a computer of 5 levels of varying difficulty, which should range from easy to challenging.
    a. I will use a depth limited alpha-beta pruned minimax algorithm to deduce which moves should be made. The varying difficulty will come from use of heuristics of different complexities in combination with different search depths.
        i. I will play each of the computer algorithms and rank them in order from easiest to most difficult. Then I will have the computers play each other. If my ranking is the same as the ranking obtained from playing the computers against each other, I will have clearly succeeded in creating five distinct difficulty settings.
6. When playing against the computer or locally, the user should be able to customise timer settings and decide who is what colour.
    a. Create a visual menu that allows the user to select from a range of a time controls, including unlimited and five or ten minutes. Users will also be able to decide what level of computer to play against and whether the computer is white, black or random. These settings will be stored, and a new game will be created.

        i. A user should be able to play local games using various different time controls and against a computer of varying difficulty. This can be verified by testing how long it takes for a user to lose due to running out of time.

7. The system will have a West Cross Chess Club section where Lewis James will have the ability to add, edit and delete member details.
    a. The server will store details of all club members in a table that is in the third normal form. Lewis James will be able to retrieve and view this information. He can then edit user details, which will be updated on the server. He can also remove users from the club or add users to the club. This will be achieved by updating the file on the central server.
        i. The club leader will be able to view all member details and edit these. The leader will also be able to remove or add members to the club. This can be shown to have been successful by viewing club details after changes have been processed.
8. Users will be able to request to join the club, a request which the leader can either approve or reject.
    a. Users who are not a member of the club will click on the club section and have the option to send a request to join the server. If they choose to send the request, the request will be sent to the server. The server will then store this request. The leader, currently Lewis James, will then have the option to allow these members to join the club or to reject the request.
        i. Users will be able to request to join the club and Lewis James will be able to see these requests. Lewis James will then either approve or deny these requests. This can be verified through testing by sending a request and seeing if it shows up for the leader. Then, the leader will accept the request. If the user who requested to be a member is now part of the club, the objective will have been fulfilled.
9. The West Cross Chess Club section will generate the fixtures given the name of the people in the chess club. This will result in users playing each other exactly twice.
    a. The computer will use the list of current members to generate a fixtures list for the season.
        i. If the system generates fixtures in a way that each user plays each other exactly twice, the algorithm will have succeeded.
    b. The user will be able to view their fixtures in the club section. They will also be able to see the results of any games.
        i. The user will view their fixture in the club section. The club leader will be able to set the result of each game.
    c. The system will collect all results and use them to update the league table standings. The league table, which is stored by the server, can be viewed by players.
        i. Once results have been set by the leader, the league table should be updated to display the new scores and standing.
10. The system will allow the club leader to generate a new season's worth of fixtures at any time.
    a. The leader will be able to press a button that generates all fixtures from scratch using the list of chess club members stored in the system.
        i. When the leader presses a button, the server should generate the fixture list from scratch.

11. Lewis James will have the option to make another user the club leader if he no longer wishes to be in charge.
    a. The system will store the user id of the club leader. If Lewis James wishes to make somebody else club leader, he will need to know their username. The system will then set this users id as that of the club leader. This will allow the Lewis James to make somebody else the club leader.
        i. The current club leader will be able to transfer the role of club leader to another valid user. If they do choose to transfer the role, the user who they have transferred the role to will be able to perform all the task that only the club leader can perform (such as updating the club's contact details).
12. The chess game will feature a two-dimensional chess board which allows users to move by clicking on a piece and then choosing from all available moves. The interface should be visually pleasing and clear and should highlight available moves and which piece has currently been selected.
    a. To create a GUI using built in features such as JFrame to display a visually pleasing chess board. The view will be a top-down view of a two-dimensional board and will be similar to chess.com's board. The control scheme must be very usable and intuitive.
        i. The board must feature a two-dimensional top-down view and be visually pleasing. To test this, I will ask five members whether the board is visually pleasing and looks somewhat similar to typical two-dimensional top-down boards (such as Lichess' board). If all five members agree, I will have succeeded. I will also ask the members whether the control scheme is usable and intuitive also.
    b. When a piece is selected, the piece is highlighted, and the system calculates all possible moves for the piece to make. These possible moves will be highlighted.
        i. To test this, I will select each piece across multiple games and use my knowledge of chess to determine whether all legal moves have been identified.
13. The menu system should be clean and visually clear, containing a small range of options which allow the user to perform the tasks they wish to.
    a. I will use the built in NetBeans drag and drop GUI builder to create a clear and visually pleasing menu. The menu will be simple and contain a small range of buttons that allow the user to navigate to the desired section of the system.
        i. Navigation should be intuitive and easy, even for novice users. Text and buttons should be readable and large. To measure this, I will ask five members whether they find navigation easy and whether they find the menu clear. If all five agree, I will have succeeded.
14. The system will have a constantly running server application which handles online play and user log in. The server will store all member details, as well as West Cross Chess Club details such as the table as well as a text file containing all ongoing games. The system will communicate with the server to facilitate online play.
    a. The server application will be constantly running and will handle all elements of the club that are online. This includes the creation of accounts, user log-in, online play, storing chess club details, generating fixtures, updating the league table, storing the chess club leader and scheduling fixtures for the chess club.
        i. If players are able to log-in and perform online tasks such as creating an account or playing against a friend online, the objective will have been met.

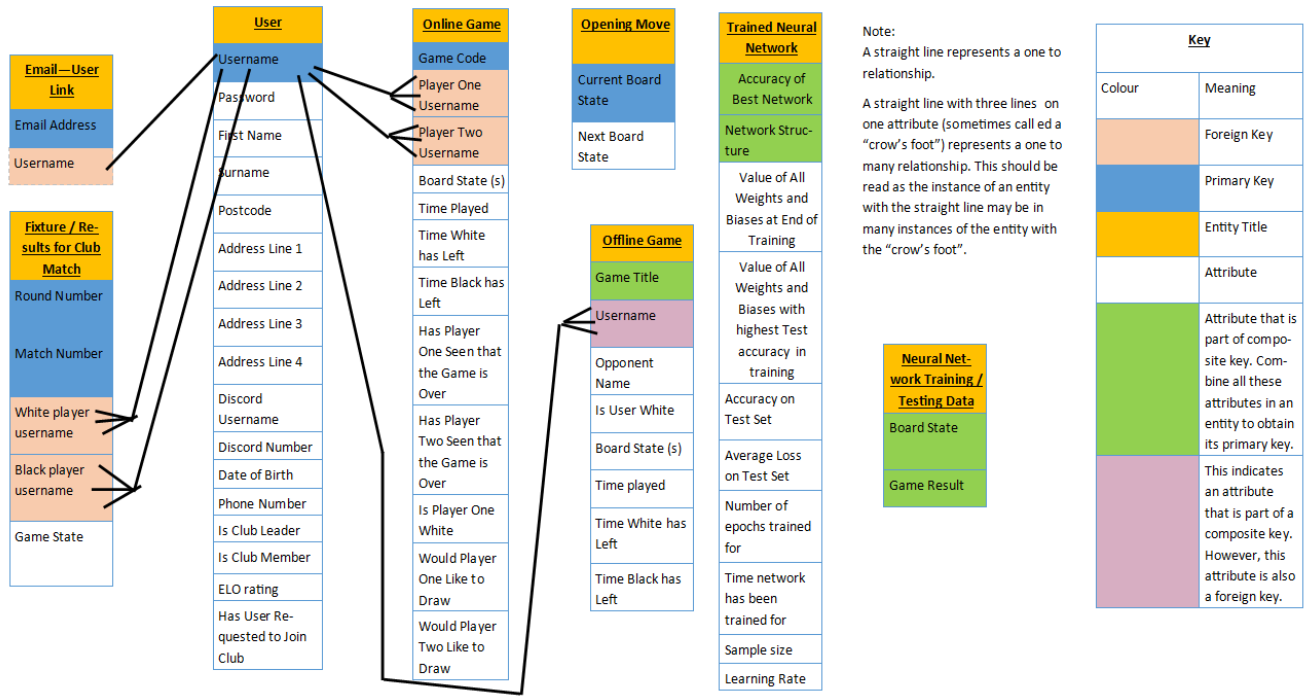
15. The system will feature contact information for the club and the leader (currently Lewis James) will have the ability to update this information.
    a. There will be a section in the club tab, which will be accessible to all users connected to the internet (including non-club members), which displays the email address and phone number of the club. These details will be stored on the server and the club leader will have the opportunity to update these details.
        i. All users should be able to view the club's contact details and Lewis James should be able to change these contact details.
16. The system will rank players via their online play results using a ratings system known as ELO.
    a. After each online game, the server will receive the result and calculate the new ELO score for each player following a set of formulae defined earlier in this document. The ELO rating of a player will be stored by the system and displayed alongside username when playing an opponent.
        i. After a game against an opponent, I will calculate what each user's new ELO score should be and compare it to what the system determined they should be. If these scores equate, the system will have succeeded.
17. When playing online, the system will display both users' username and ranking.
    a. When matching against an opponent, the server will store both users' name and ELO rating. These will be displayed to the other user when playing online. These (and all) calculations should be correct and accurate.
        i. When playing online, users should be able to see both their username and ELO rating as well as their opponent's rating and username.
18. GENERAL OBJECTIVES AND CRITERIA
    a. The system should be easy to maintain. I will achieve this through use of extensive documentation (e.g. comments in code to explain the purpose of methods). Bugs should be easy to identify and fix, something which I will achieve by designing the system in a modular fashion.
        i. A competent programmer should be able to recreate the system given its documentation.
        ii. My system will be modular, making use of many independent methods. To test whether my system is easy to debug, I aim for users to encounter no more than three minor bugs in a session and no major bugs in an average session.
    b. The system should fully meet the requirements of the specification.
        i. I will ask Lewis James whether the system performs as he intends it to once I finish it.
    c. Validation should be employed for all appropriate fields of manual data entry.
        i. I will aim to catch at least 90% of invalid inputs.
    d. All calculations performed should be accurate and correct
        i. Calculations such as ELO rating should be accurate and all relevant success criteria relating to ELO calculation should be achieved. All other calculations should be correct, such as the updating of the league table.
    e. The system should be visually clear and easy to navigate. It should be easy to use, even to those unfamiliar with technology.
        i. I will ask chess club members whether they found the system intuitive and easy to use, and if they all agree, I will have succeeded.

Software Development

My code base for the software development is tripartite. I have a project title “PhilipM CS5 Software Development User System”. This code is designed for the system users and is the biggest part of the code base. This code allows the user to do everything agreed on in my design (and subsequent post-prototype refinement of design). I also have a project title “PhilipM CS5 Software Development Server System”. This project is used to run the server that powers the online elements of my project. For example, this server allows users to create accounts and play chess online against other users. Finally, I have a third project titled “PhilipM CS5 Software Development Neural Network train”. This project contains the code and data used to train a Neural Network to play chess. This neural network model was then exported to my user system – albeit without the code to show how it was trained.

Data Structures Used

Here is the Entity Relationship Diagram (ERD) which represents the data my system stored. Note: all of my data was stored in the third normal form (as evidenced below).



User (Username, Password, First Name, Surname, Address Line 1, Address Line 2, Address Line 3, Address Line 4, Discord Username, Discord Number, Data of Birth, Phone Number, Is Club Leader, Is Club Member, ELO rating, Has User Requested to Join Club)

Email – User Link (Email Address, Username)

Fixtures / Results for Club Match (Round Number, Match Number, White Player Username, Black Player Username, Game State)

Online Game (Game Code, Player One Username, Player Two Username, Board State(s), Time Played, Time White has Left, Time Black has Left, Has Player One Seen that the Game is Over, Has

Player Two Seen that the Game is Over, Is Player One White, Would Player One Like to Draw, Would Player Two Like to Draw)

Opening Move (Current Board State, Next Board State)

Offline Game (Game Title, Username , Opponent Name, Is User White, Board State (s), Time Played, Time White has Left, Time Black has Left)

Trained Neural Network (Accuracy of Best Network, Network Structure, Value of All Weights and Biases at End of Training, Value of All Weights and Biases with highest Test accuracy in training, Accuracy on Test Set, Number of epochs trained for, Time network has been trained for, Sample Size, Learning Rate)

Neural Network Training / Testing Data (Board State, Game Result)

This design can be verified by the source code attached, as well as the text files which actually store this data.

Here are some screenshots to evidence the fact that the data structures described are employed.

Here is a screenshot from the server system folder, showing the filenames of various files.

Name	Status	Date modified	Type	Size
build		11/03/2021 16:06	File folder	
dist		11/03/2021 17:52	File folder	
nbproject		26/11/2020 13:34	File folder	
src		26/11/2020 13:34	File folder	
test		22/12/2020 14:28	File folder	
build.xml		26/11/2020 13:34	XML Document	4 KB
emailClubContact.txt		07/03/2021 20:37	Text Document	1 KB
emailLinkFile.txt		11/03/2021 12:17	Text Document	1 KB
fixtures.txt		11/03/2021 13:57	Text Document	1 KB
manifest.mf		26/11/2020 13:34	MF File	1 KB
onlineGames.txt		11/03/2021 14:48	Text Document	1 KB
PhilipM_CS5_Software_Development_Server_Sys...		11/03/2021 16:06	Executable Jar File	104 KB
phoneNumber.txt		07/03/2021 20:37	Text Document	1 KB
serverSettings.txt		10/03/2021 14:37	Text Document	1 KB
userDetails.txt		11/03/2021 12:17	Text Document	1 KB

The file “fixtures.txt” is used to store the entity **Fixtures / Results for Club Match**.

Here is the fixtures file. It is important to note that the format in the ERD is correct, but that the actual data is encoded in a slightly abstract way to make it easier for the computer to understand.

fixtures.txt - Notepad

File Edit Format View Help

```
4
luke;109phil;109@;109phil;109luke;109@;109philip;109phil;109@;109philip;109luke;109philip
@;109philip;109philip;109luke;109philip;109phil;109luke;109@;109luke;109phil;109phil;109@
-2;109-1;109-2;109-1;109-1;109-2;109-1;109-2;109-2;109-1;109-1;109-2
aasifoakofako325
```

The file “emailLinkFile.txt” represents **Email – User Link**.

This is stored in a format that is much easier for humans to understand.

emailLinkFile.txt - Notepad

File Edit Format View Help

```
lukesarfas@icloud.com;l09luke
phil@aol.com;l09phil
phil3@aol.com;l09phil3
philip@aol.com;l09philip
sdsdas@icloud.com;l09ds
```

The file “userDetails.txt” represents **User**. (Note “;lo9” is used as a separation character).

userDetails.txt - Notepad

File Edit Format View Help

```
ds;l09lukes123;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09false;l09false;l091000;l09false
luke;l09lukes123;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09false;l09true;l09988;l09false
phil;l09passw;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09false;l09false;l091000;l09false
phil3;l09passw;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09false;l09false;l091000;l09false
philip;l09passw;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09;l09true;l09true;l091012;l09false
```

The file “onlineGames.txt” represents **Online Game**.

onlineGames.txt - Notepad

File Edit Format View Help

```
000001;l091uke;l09philip;l09rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1fensplit*rnbqkbnr/pppppppp/8/8/5P2/8/PPPPP1PP/RNBQKBNR b KQkq f3 0 1;l091;l09-100;l09-100;l09false;l09false;l09false;l09false;l09-1;l09false;l09false
000002;l091uke;l09;l09rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1;l090;l09600;l09600;l09false;l09false;l09false;l09true;l09-1;l09false;l09false
```

The remaining text files store static data (i.e., only one instance of something). Hence, they are obviously not included in the ERD.

Here is a screenshot from the user system folder.

Name	Status	Date modified	Type	Size
Assets		19/02/2021 09:57	File folder	
build		11/03/2021 16:03	File folder	
dist		12/03/2021 10:17	File folder	
javadoc		11/03/2021 17:45	File folder	
nbproject		26/11/2020 13:34	File folder	
offlinegames		10/03/2021 19:23	File folder	
src		26/11/2020 13:34	File folder	
test		27/12/2020 00:49	File folder	
bestNet.txt		10/03/2021 09:35	Text Document	6,623 KB
boardSetting.txt		10/03/2021 19:20	Text Document	1 KB
build.xml		26/11/2020 13:34	XML Document	4 KB
GNU.txt		02/02/2021 17:33	Text Document	35 KB
haveTermsAndConditionsBeenAccepted.txt		11/03/2021 14:48	Text Document	1 KB
manifest.mf		26/11/2020 13:34	MF File	1 KB
openingMoves.txt		10/03/2021 12:10	Text Document	1,479 KB
PhilipM_CS5_Software_Development_User_Syste...		11/03/2021 16:03	Executable Jar File	855 KB
portNumber.txt		11/03/2021 12:16	Text Document	1 KB
serverAddress.txt		11/03/2021 12:16	Text Document	1 KB
stockfish_20090216_x64.exe		02/02/2021 16:11	Application	21,710 KB
Stockfish-sf_12.zip		02/02/2021 17:39	Compressed (zipped)...	220 KB

“openMoves.txt” represents **Opening Move**.

openingMoves.txt - Notepad

File Edit Format View Help

```
1n1q1rk1/rbpp2pp/1p2pn2/p4p2/1bPP4/3N1NP1/PP2PPBP/R1BQ1RK1 b - - 5 10,1n1q1rk1/rbppb1pp/1p2pn2/p4p2/2PP4/3N1NP1/PP2PPBP/R1BQ1RK1 w - - 6 11
1n1q1rk1/rbpp2pp/1p2pn2/p4p2/1bPP4/3N2P1/PP1NPPBP/R1BQ1RK1 w - - 4 10,1n1q1rk1/rbpp2pp/1p2pn2/p4p2/1bPP4/3N1NP1/PP2PPBP/R1BQ1RK1 b - - 5 10
1nbq1rk1/r1pp2pp/1p2pn2/p3Np2/1bPP4/6P1/PP1NPPBP/R1BQ1RK1 w - - 2 9,1nbq1rk1/r1pp2pp/1p2pn2/p4p2/1bPP4/3N2P1/PP1NPPBP/R1BQ1RK1 b - - 3 9
1nbq1rk1/r1pp2pp/1p2pn2/p4p2/1bPP4/3N2P1/PP1NPPBP/R1BQ1RK1 b - - 3 9,1n1q1rk1/rbpp2pp/1p2pn2/p4p2/1bPP4/3N2P1/PP1NPPBP/R1BQ1RK1 w - - 4 10
```

The folder “offlinegames” stores the offline games. Here is a screenshot from that folder.

Name	Status	Date modified	Type	Size
GUEST_ID9239923.txt	✓	11/03/2021 11:21	Text Document	3 KB
philip.txt	✓	10/03/2021 19:23	Text Document	0 KB

My data is stored in an interesting structure with each text file actually storing the game data for a given user.

GUEST_ID9239923.txt - Notepad

File Edit Format View Help

```
Game1;109Level Four Computer;109false;109rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1fensplit***rnbqkbnr/pppppppp/8/8/5P2/8/PPPPP1PP/RNBQKBNR b KQkq f3 0 1 1fensplit*rnbqkbnr/ppp1pppp/8/3p4/5P2/8/PPPPP1PP/RNBQKBNR w KQkq d6
PPP3PP/R1BQK2R b KQkq - 3 7fensplit*rn1qk1nr/p2b1ppp/1p2p3/2pp4/5P2/2NBPN2/PPP3PP/R1BQK2R w KQkq - 0 8;10939;109-100;109-100
Game2;109Level Four Computer;109true;109rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1fensplit**rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR b KQkq e3 0 1fensplit*rnbqkb1r/pppppppp/5n2/8/4P3/8/PPPP1PPP/RNBQKBNR w KQkq - 1 2fe
Game3;109Level One Computer;109false;109rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1fensplit*rnbqkbnr/pppppppp/8/8/5P2/8/PPPPP1PP/RNBQKBNR b KQkq f3 0 1 1fensplit*rnbqkbnr/ppp1pppp/8/3p4/5P2/8/PPPPP1PP/RNBQKBNR w KQkq d6 0 2
```

This allows for a given game title to be used by multiple different users.

As mentioned previously, the remaining text files store static data or data with only one record.

Here is a screenshot of my neural network folder.

Name	Status	Date modified	Type	Size
best net 74	✓	10/03/2021 10:47	File folder	
bestNetOld	✓	10/03/2021 10:52	File folder	
build	☁	11/03/2021 21:33	File folder	
data for comparison network	✓	27/01/2021 14:34	File folder	
data raw DONT CHANGE	✓	20/01/2021 15:15	File folder	
dist	☁	11/03/2021 21:34	File folder	
nbproject	✓	20/01/2021 14:03	File folder	
src	✓	11/03/2021 18:17	File folder	
Stockfish data attempt	✓	09/03/2021 17:13	File folder	
test	☁	20/01/2021 14:04	File folder	
Training Data Win, loss and draw long net input	✓	15/02/2021 11:40	File folder	
bestNet.txt	✓	10/03/2021 09:35	Text Document	6,623 KB
build.xml	✓	20/01/2021 14:03	XML Document	4 KB
drawstates.txt	✓	15/02/2021 11:51	Text Document	9,143 KB
losestates.txt	✓	15/02/2021 11:51	Text Document	7,207 KB
manifest.mf	✓	20/01/2021 14:03	MF File	1 KB
networkInfo.txt	✓	10/03/2021 09:35	Text Document	44,772 KB
PhilipM_CS5_Software_Development_Neural_Ne...	✓	11/03/2021 21:33	Executable Jar File	53 KB
stockfish_20090216_x64.exe	✓	02/02/2021 16:11	Application	21,710 KB
testData.txt	✓	29/01/2021 12:54	Text Document	304,505 KB
trainData.txt	✓	29/01/2021 12:53	Text Document	3,420,942 KB
winstates.txt	✓	15/02/2021 11:51	Text Document	9,541 KB

The “testData.txt” and “trainData.txt” represent **Neural Network Training / Testing Data**. These files are merged into one entity as they both are stored in exactly the same way (test data is simply a randomly chosen subset of the training data).

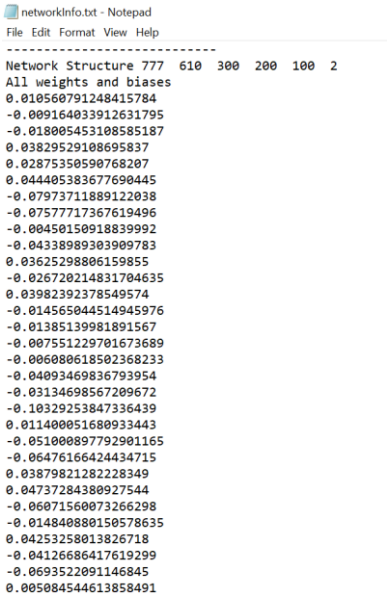
The data is not easily readable by a human, but it stores both the game result and board state.

testData.txt - Notepad

File Edit Format View Help

```
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Here is the text file that stores Neural Networks generated during training. “networkInfo.txt” represents **Trained Neural Network**.



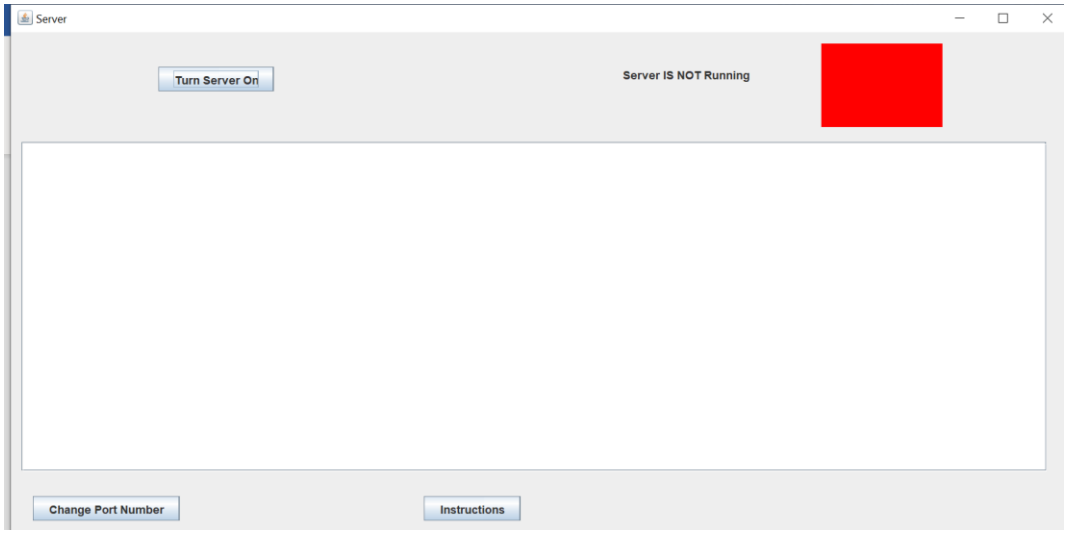
```
networkInfo.txt - Notepad
File  Edit  Format  View  Help
----------------------------
Network Structure 777  610  300  200  100  2
All weights and biases
0.010560791248415784
-0.009164033912631795
-0.018005453108585187
0.03829529108695837
0.02875350590768207
0.044405383677690445
-0.07973711889122038
-0.07577717367619496
-0.00450150918839992
-0.04338989303909783
0.03625298806159855
-0.026720214831704635
0.03982392378549574
-0.014565044514945976
-0.01385139981891567
-0.007551229701673689
-0.006080618502368233
-0.04093469836793954
-0.03134698567209672
-0.10329253847336439
0.011400051680933443
-0.051000897792901165
-0.06476166424434715
0.03879821282228349
0.04737284380927544
-0.06071560073266298
-0.014840880150578635
0.04253258013826718
-0.04126686417619299
-0.0693522091146845
0.005084544613858491
```

All other files in the folder are static or store data of only one record (and hence are irrelevant).

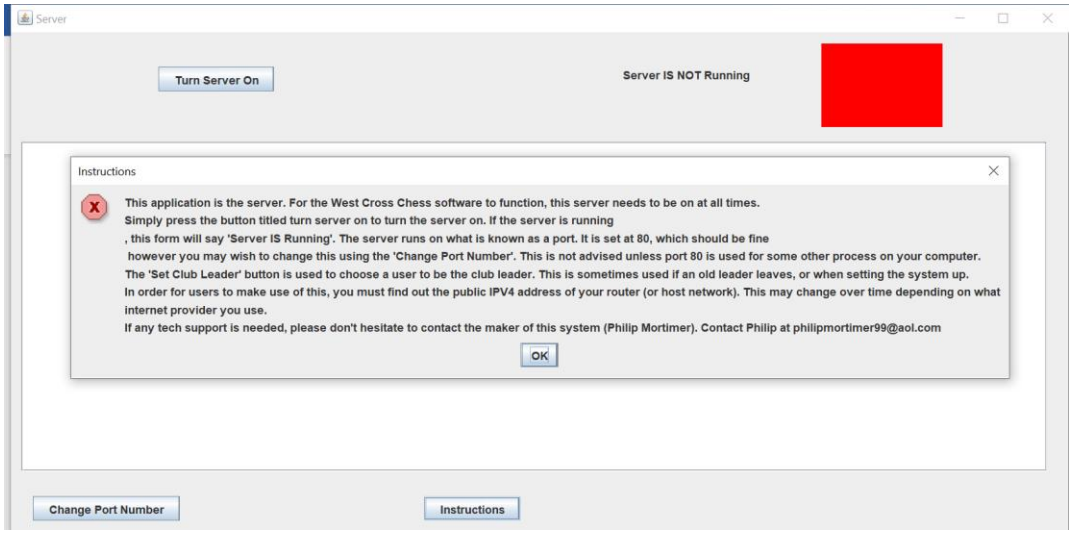
<u>User Interface</u>

My Neural Network has no user interface. This is because the project is only used by to train an AI to play chess. This AI is then transferred to my user system.

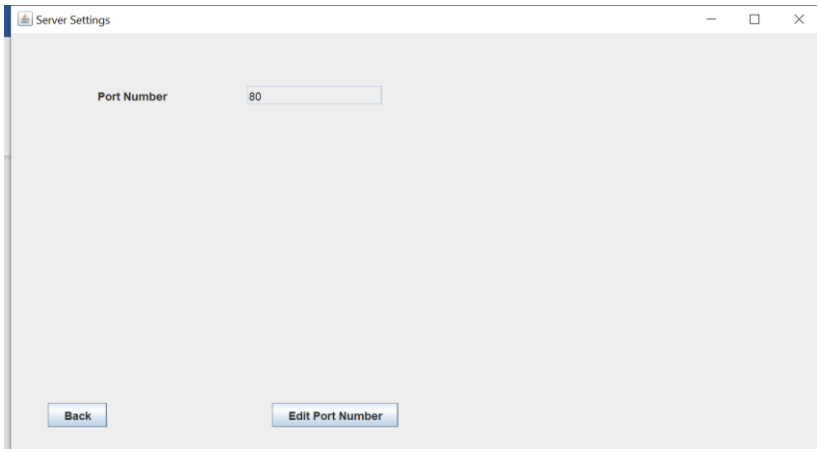
My Server system consists of two forms. The main form is used to turn the server on and off and see whether the server is running.



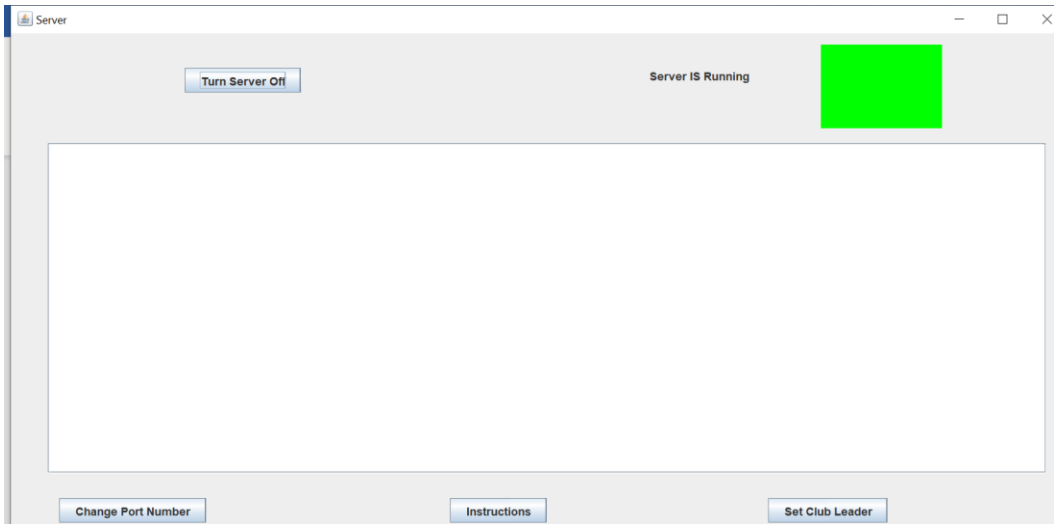
The button in the top left-hand corner of the screen is used to turn the server on. A label clearly displays whether the server is running or not. To make this visually clear, there is also a big red block to indicate the server status. There is a text box that takes up most of the form. This form displays any errors that have occurred, which gives the user a clear and obvious place to see and read errors. The instructions button gives the user some details on how to operate the server.



If the user wishes to edit the port number of the server, they are taken to a simple form that allows them to change the port number.



This is what the main form looks like when the server is running:

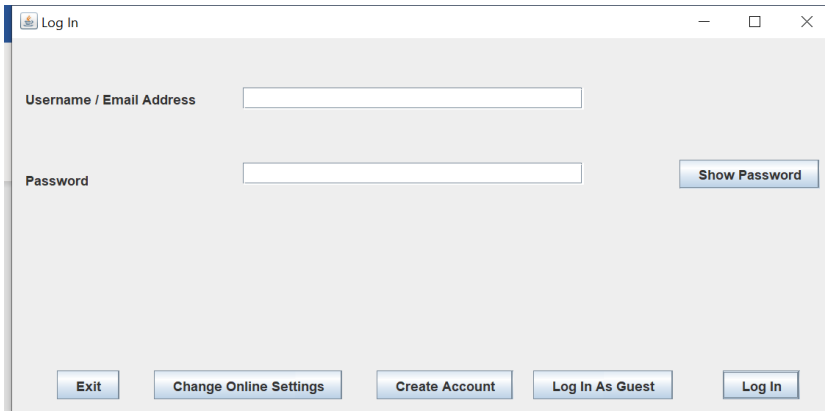


This form also allows the user to appoint a club leader from the server. This is designed to make setting the system up easy. I feel that the form is very easy and clear, with the most important aspect being the fact that the server status is displayed clearly. This form is able to much more

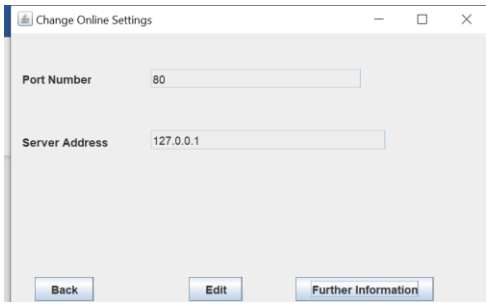
minimalist as my client is rather technologically experienced. I have also promised to go on a video call with him to take him through how to use the server which allows the form to be so minimalist.

However, the form offers great visual clarity and is very simple to use and hence successful.

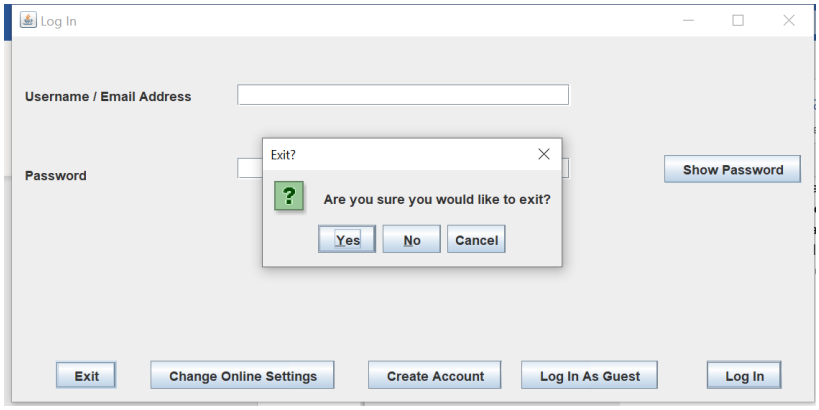
For my user system, I will be aiming my system at a group of people who may not be very technologically competent and who I won't be able to video call to clarify how the system works. Hence, making the system easy to use and hard to break is very important. Therefore, simple and clean visuals are vital to the system's success.



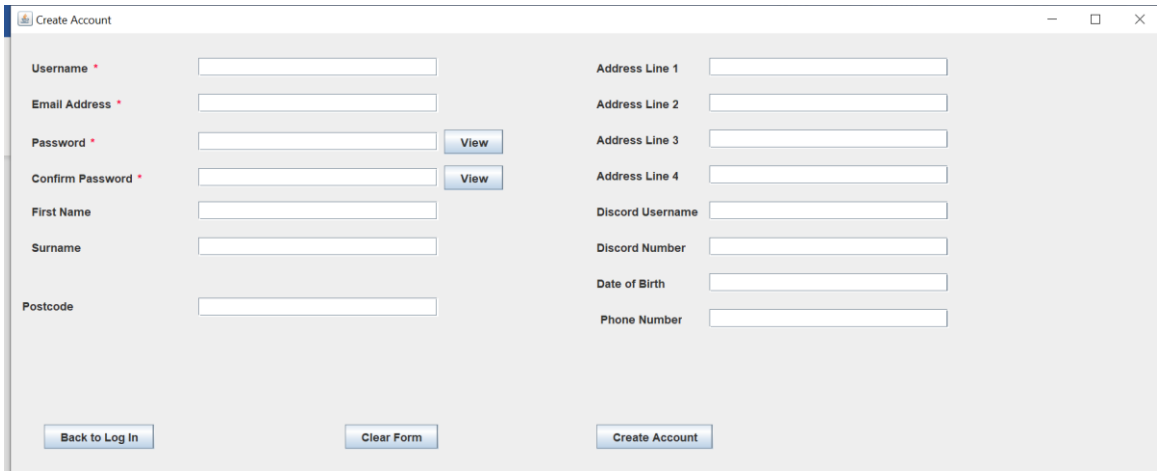
My log in form is a simple form that prompts the suer to enter a username and password. It also has a few other clearly labelled buttons that allow the user to perform various actions. Given that users may not be too confident at typing, I have included the option to display their password. I have also chosen to have a single text field to input either the username or email address. This far clearer than having a text field for both a username and an email address (which could wrongly lead users to think that they need to fill in both fields).



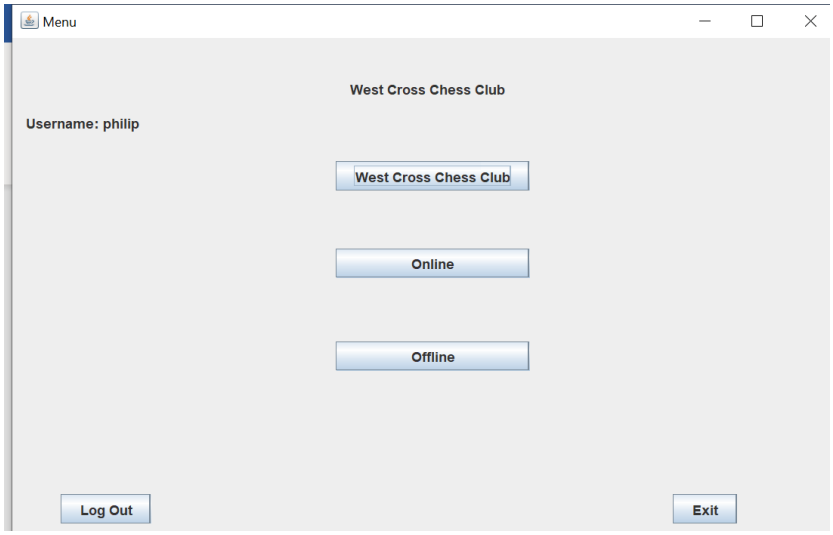
There is a simple form provided to allow the user to change the server address. The further information button essentially tells the user to speak to the West Cross Chess Club leader – who will know the correct server address details. The address will be automatically filled out to the correct address at the time of code distribution. The club leader has a static ip address, so it is unlikely that the user will have to change the network details often / ever.



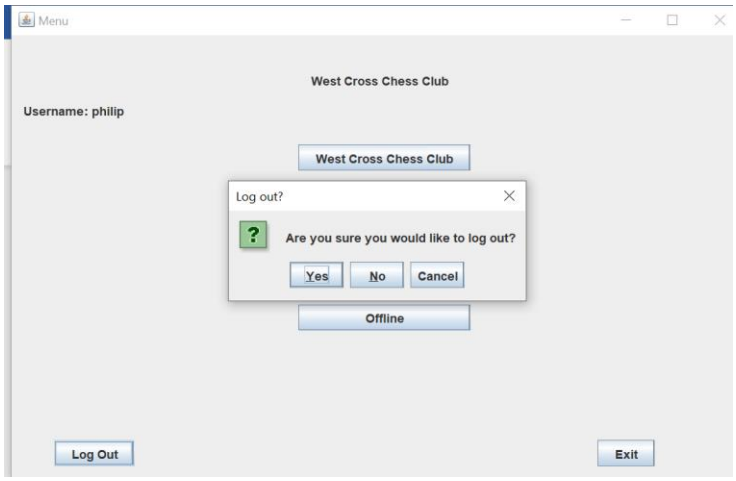
When performing important actions, such as exiting the system, users are always asked for confirmation. This is done to minimise the chance of the users accidentally doing something they did not wish to do. In my investigation, some users said that they often used to accidentally exit the chess system.



To create an account the user is taken to a form which displays the details they are asked to fill in. These details are very clear and easy to fill in. A focus traversal policy is in place for more advanced users which allows them to navigate between components in the correct order to create an account quickly. A red star is used to indicate required fields – a simple yet effective way to communicate this to user. Once more, passwords are masked to hide them from other people viewing a screen. However, they can be unmasked if the user is having trouble entering the data. The field is so important that the user is asked to enter their password twice. This is done to reduce the likelihood of mistyping their password (as a mistyped password would likely lead to them not being able to log in to their account again). Users are prompted to enter their email addresses as man felt that usernames are quite tricky to remember. A button is present which clears all fields in the form. This may be useful for users who wish to start the form from scratch.

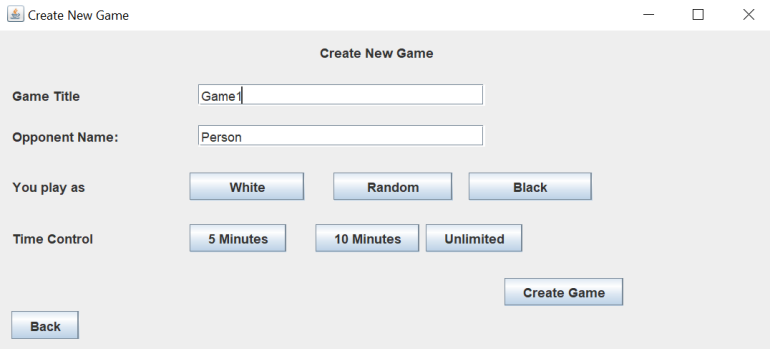


Users are taken to a menu with three big buttons. This simplistic design was used as many members reported that systems such as chess.com's felt cluttered and very hard to navigate due to the large number of features that were irrelevant to them. Each of the three big buttons is clearly labelled and makes navigation easy. The username is also displayed to remind the user in case they wish to use it for future log ins or in case they wish to communicate their username to a friend.



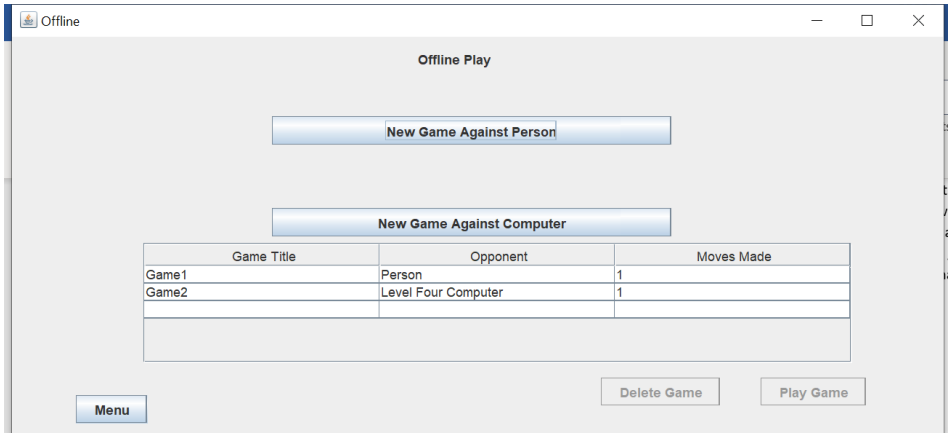
Users are asked to confirm that they wish to log out. This once more helps less technologically capable users from having to log in again due to erroneous inputs.

When the user wishes to play chess against an offline opponent, they are taken to a menu that displays all of their current ongoing offline games. It also displays two large buttons which allow the user to create new games.



When creating a game against an offline person, they are taken to this form. The form automatically generates a suggested game title that is valid. This is done for users who do not want to waste time customising game name and opponent name. A set of toggle buttons for colour of pieces and time control are displayed. Only one of each relevant button can be selected, making choosing game settings extremely easy. Users who wish to title the game and select an opponent name have the option to do so, making it more obvious to the user what a given game represents.

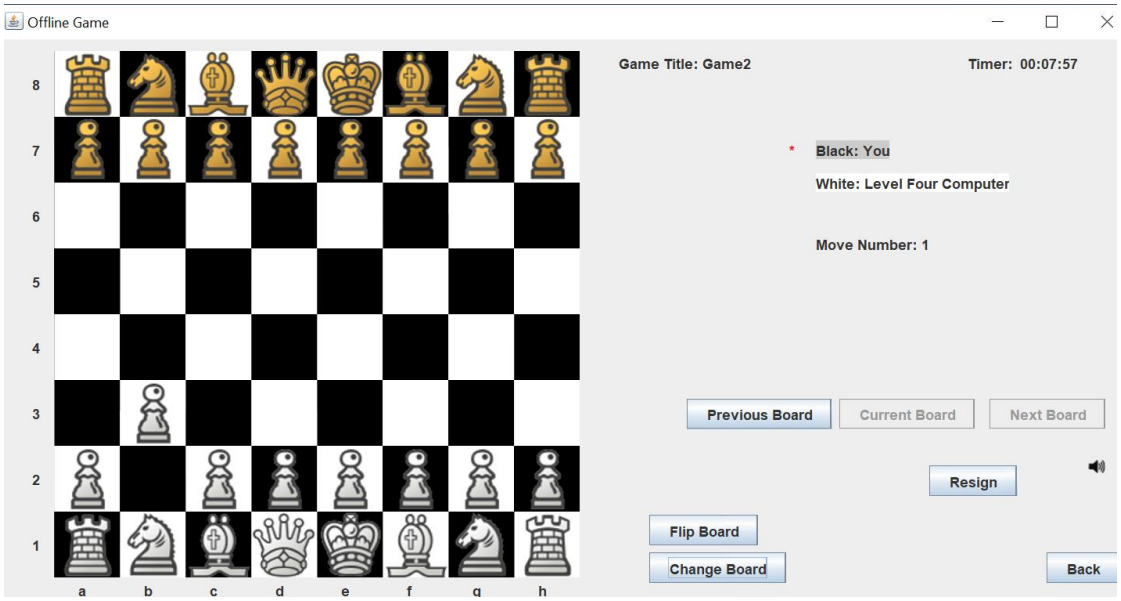
Similar to with the game against a person form, buttons are used to allow the user to set their game settings. A valid game title is also automatically generated to reduce wasted time for users who simply wish to play quickly.



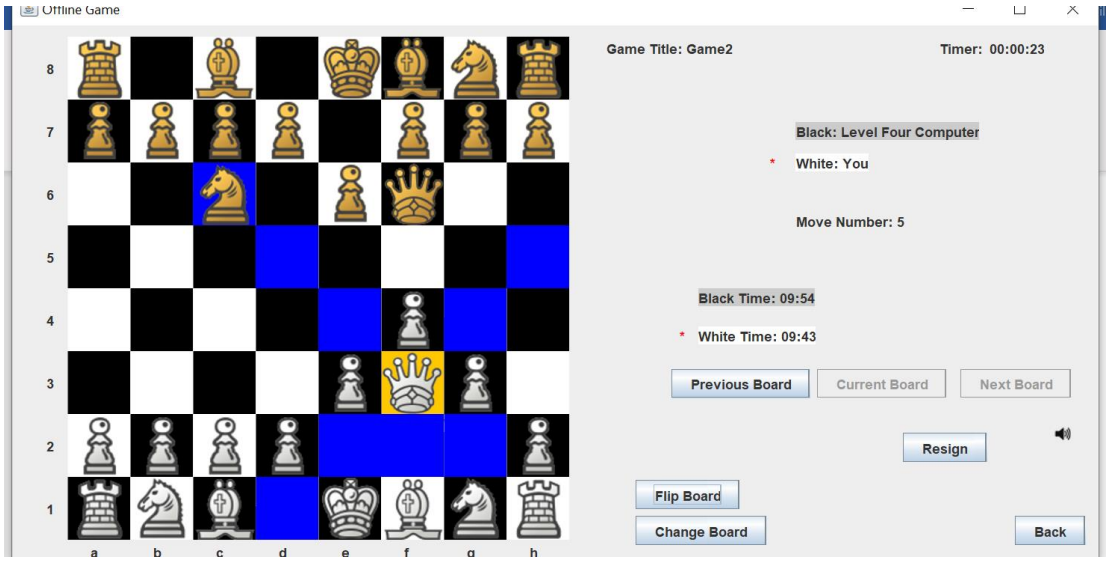
Once the games have been created, they are displayed in the table, providing clear information about the game. Users can play or delete the game by selecting the appropriate table row and pressing the play or delete buttons.

Before deleting games, a user is asked to confirm that they do wish to delete the game. This is done with the target audience in mind, who are not all highly technologically capable.

When playing, the user is taken to a form with a chess board which I feel looks very visually clear. I would say that the board looks good and that it is obvious which pieces are which. The rank and file of each square is also displayed for the user's convenience. The game title and opponent name are clearly displayed. The name of the players are coloured appropriately to underscore who is who. A red star next to the player who's turn it is to move is displayed. This is very useful as it is possible to lose focus and be unsure as to who has to play next. A timer in the top left-hand corner of the screen shows how long the game has been going on for. This timer is useful due to the information it displays and also because it tells the user that the game is going and has not crashed. The user has the option to navigate between all board states. This can be very useful when the computer has made a move, but you are not sure which piece has been moved. The user can resign the game, which simply deletes the board. A simple speaker icon is used to disable and enable sound for the game. I felt that this was preferable to a button labelled "sound" as it conveys more clearly whether the sound is on or off. Users are able to flip the chess board. This was done to make playing person versus person better as each player may want to see the board from their perspective (which mimics face-to-face chess).



The change board button changes the visual style of the board (both the colour of the squares and the images used to represent the chess pieces are altered). This was done as I had found two separate chess layouts that users liked and looked very good. Users typically preferred the style they used first. Hence, I felt that it would be a useful option to allow the user to use both board styles. Clearly, having a good-looking chess board is very important and this option will allow users to choose the board that they prefer to use. This clearly enhances the quality of the user experience.



To move a piece, the user first clicks on the piece. This piece is highlighted and all possible legal moves from that piece are highlighted. This control system is standard in chess and prevents any erroneous inputs from being made. It is also very easy and intuitive to use and leaves the user in now doubt as to how to play. This control scheme was requested by the leader of the club, who says that is very comfortable and easy to use.

Next, we will look at the online chess game menu.

The online menu is very similar to the offline menu in terms of its design philosophy. The table stores the game, and the three big buttons are used to create or join online chess matches. The top big button creates a game against a random opponent who has also requested to play a game.

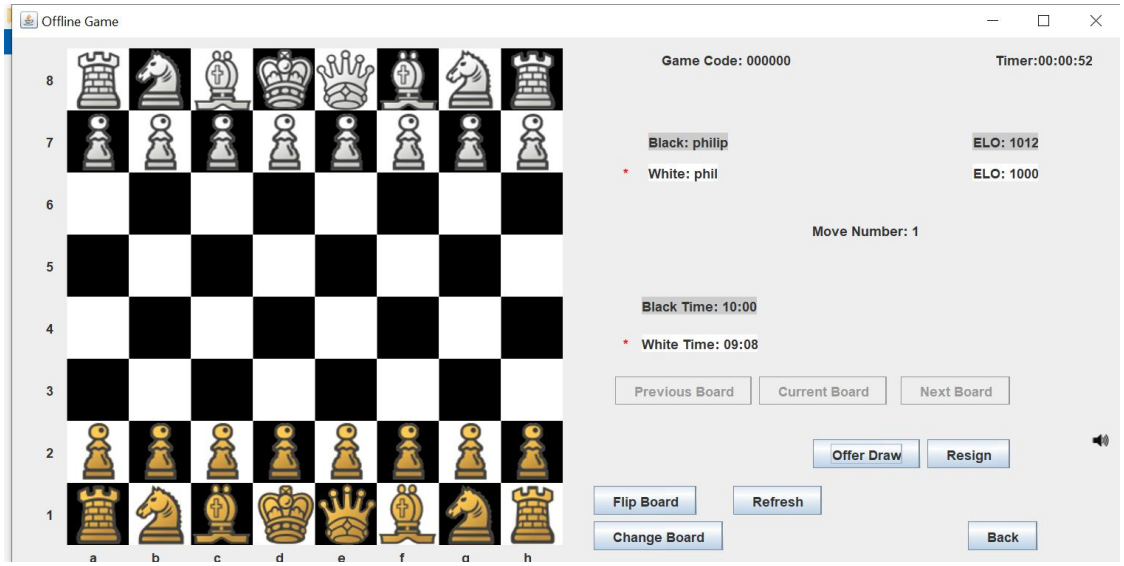
If no requests to play random games exist, a new one is created. This means that the user is waiting on somebody else to request to play a random game before the game commences. Until this occurs, the game cannot be played. The table updates every ten seconds, so the user can wait in the menu for updates to their game request which is important as users may not known enough about the system to manually refresh the table by going to the menu and back the table.

However, if another user has already requested a random game or requests a random game after the first user does, a new game is created. The colour of the game on the table changes. This is used to clearly indicate that the game is playable. This is a highly intuitive system. The system prevents users from even trying to play invalid games, which reduces the likelihood that they will find the system confusing.

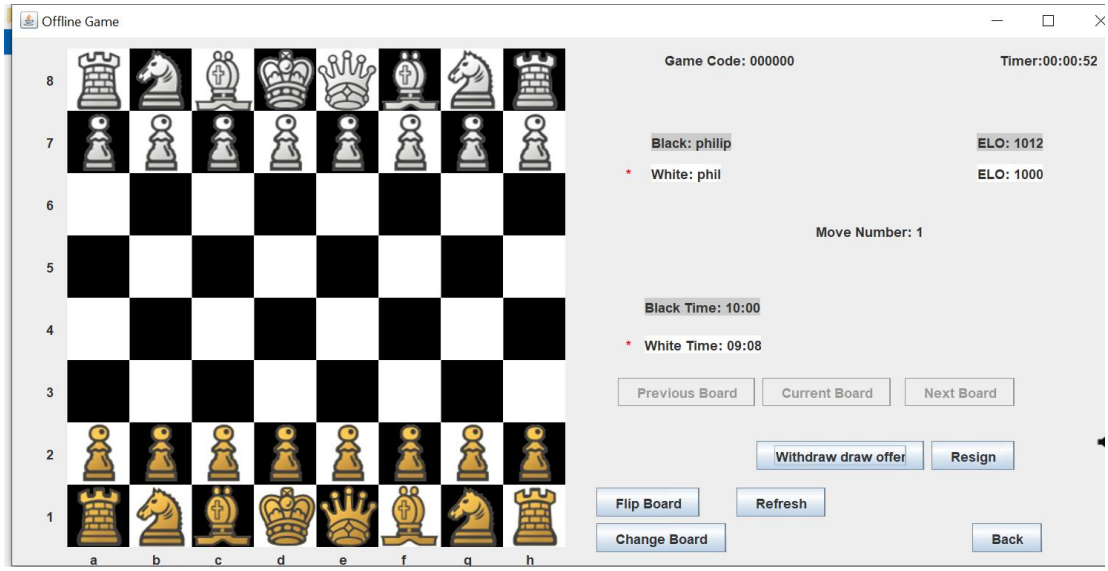
This form allows the user to create a game against a friend using their username. Validation is performed to ensure a valid username and time control setting is used.

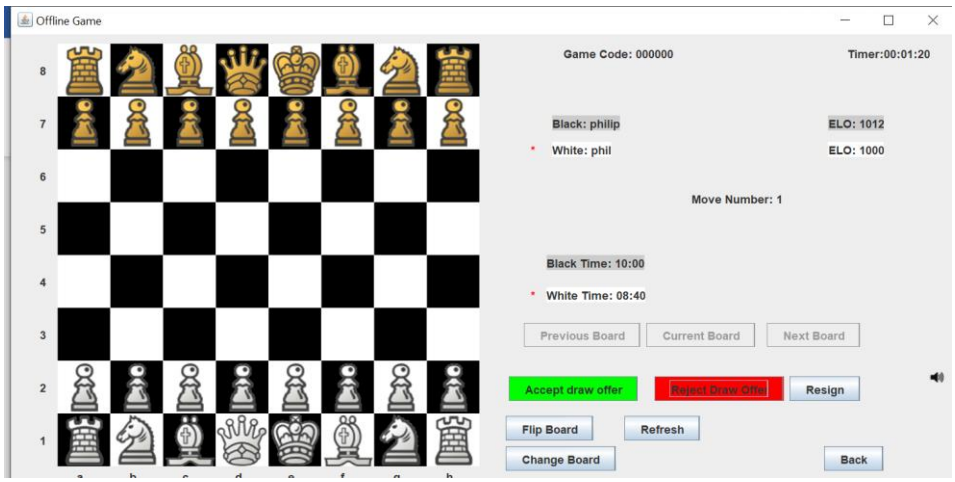
If the user wishes to create a game and share the game code instead, they can do this by pressing the big button labelled “Create Code to Share Instead”. This then disables the friend username text field, clearly showing to the user that the username will no longer be used. The user can toggle between these two options as they please. This is useful as it could be that users attempt to create a game using a friend’s username. If for whatever reason, they get the username wrong, they may find it easier to then just create a game code instead.

Users joining with a friend code are prompted to enter the code in a clear window.



The online chess board is very similar to the offline board. The ELO rating of both players is displayed. This is useful as it allows players to gauge both their own skill level and that of their opponent. If a game is being played with time controls, the time that each user has left is displayed. The game code is displayed. A refresh button is also there in case the user wishes to manually refresh the game. This is (of course) not necessary, as the game refreshes every second, but it may be useful if a user is running low on time. However, the user does now have the chance to offer a draw to an opponent. This is standard practice in chess, with players from all levels choosing to draw before stalemate is reached to save both player's time.

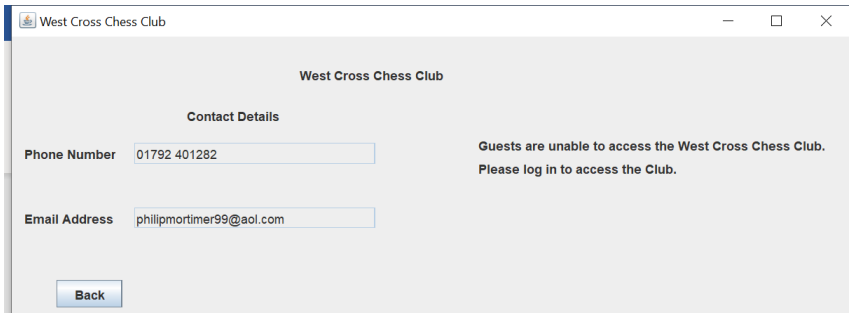




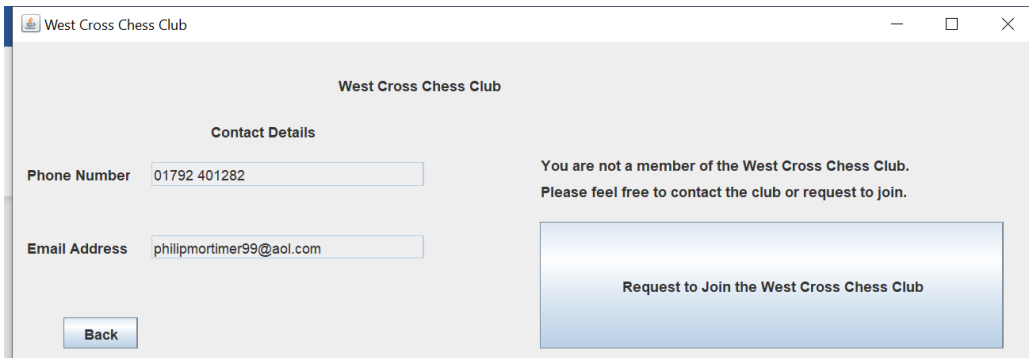
When a draw offer is made, it is made very clearly to the user being offered a draw as bright colours are used. This is very important as a user may not notice the fact that a draw has been offered if it were just a small grey button. The colour coding also helps users easily differentiated between accepting and rejecting a draw.

When any terminal state is reached, a message is displayed to the user indicating the game result. This method is very simple and easy to understand and unambiguously communicates the game result.

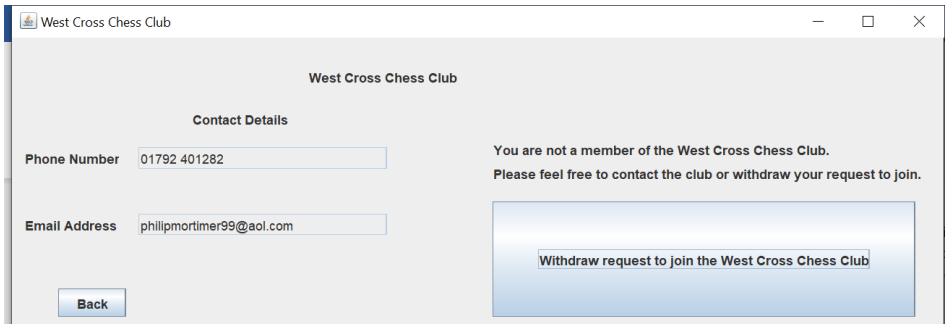
There are four user access levels in my system: guest, user, member and leader. Guests simply press in “Log in as Guest” on the log in screen. Users are people who have an account but are not members of the West Cross Chess Club. Members are people who have accounts and are members of the West Cross Chess Club but are not the club leader. There is only one leader, and they have an account and are a member of the chess club and the leader of the chess club. The West Cross Chess Club section of the system depends on which type of user is accessing it.



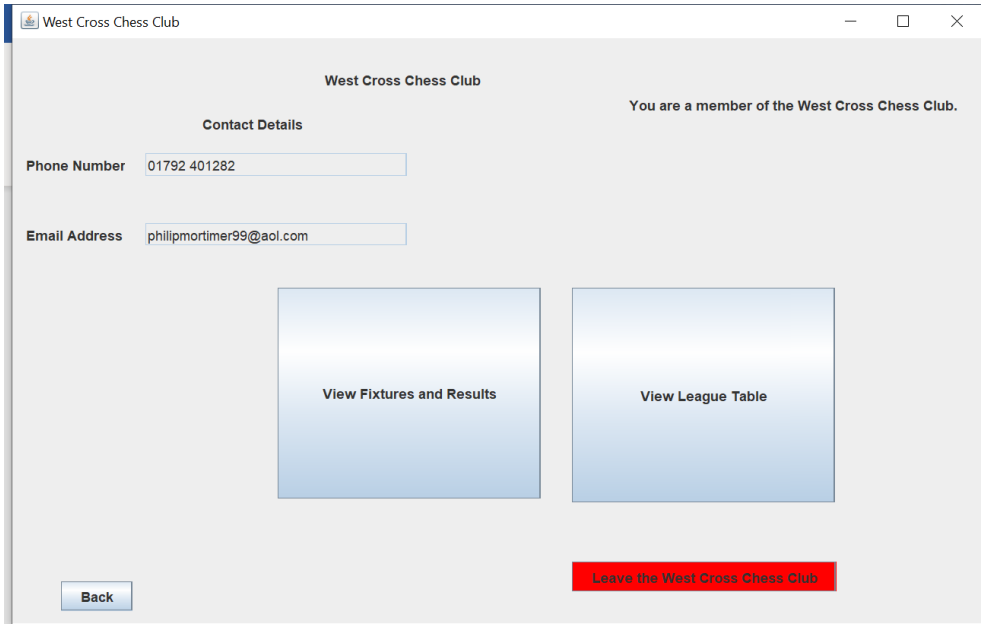
Guest users are told that they must log in to access club features, with the back button being the only input option. The club contact details are also displayed in case the guest wishes to contact the club to make an enquiry. The form is simple and tells the user clearly that the user needs to log in to access more features.



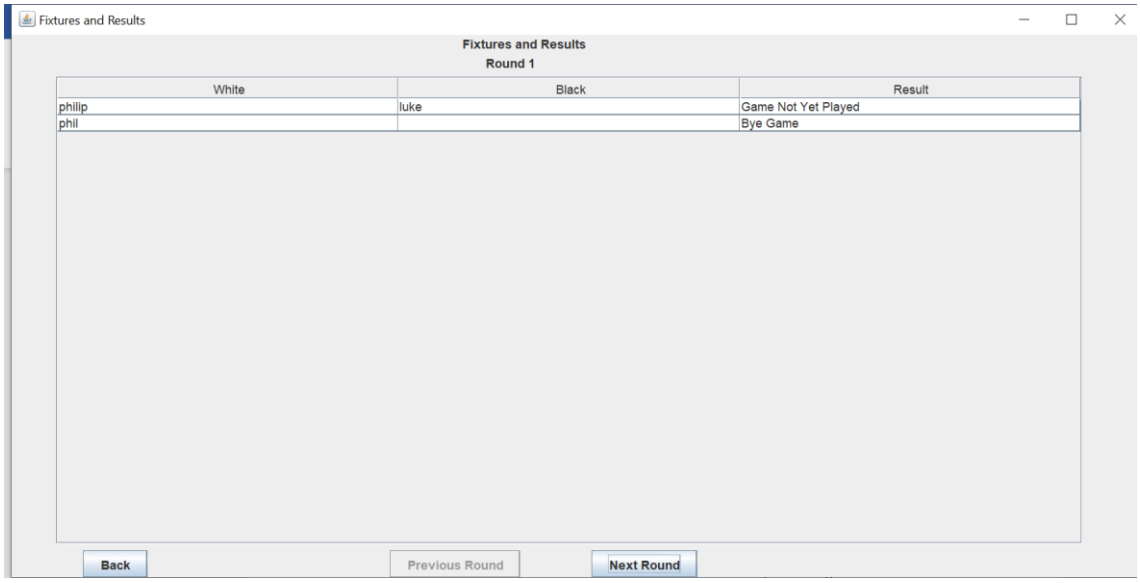
Users have a similar screen to guests, although they are able to request to join the club. This form uses a large button to allow requests to join the club. This makes it very clear that this is the only action they can perform on the form.



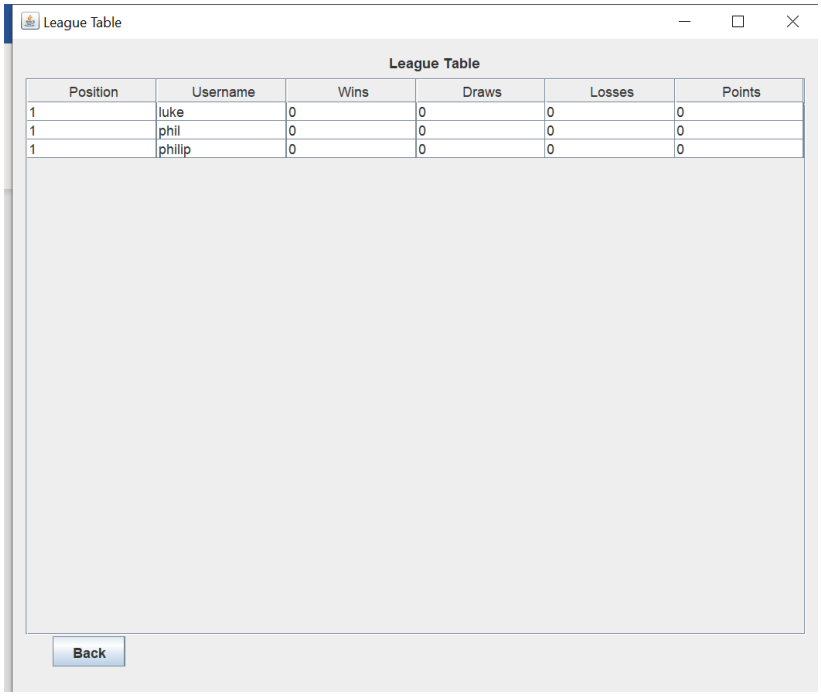
Users who have requested to join the club have a similar form, although they form now allows them to withdraw their request to join the club. This may be useful for those who pressed the button out of curiosity but do not actually wish to partake in the club.



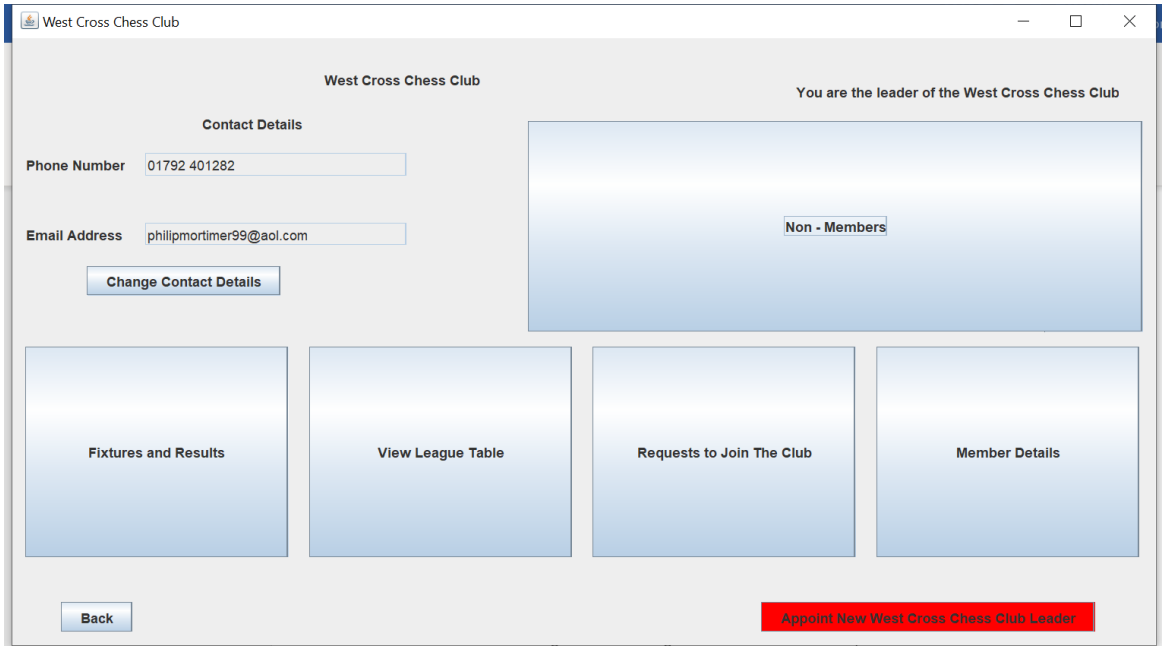
Members are taken to a form where they can view the contact details. This is useful as members will from time to time wish to contact the club leader to make various enquiries. The main actions performed by chess club members are signified by the big buttons. This is done to attract users to the write area of the screen and make the experience intuitive. The button that enabled a member to leave the club is coloured in red and asks for confirmation before proceeding. This should hopefully server to prevent anybody from accidentally quitting the club.



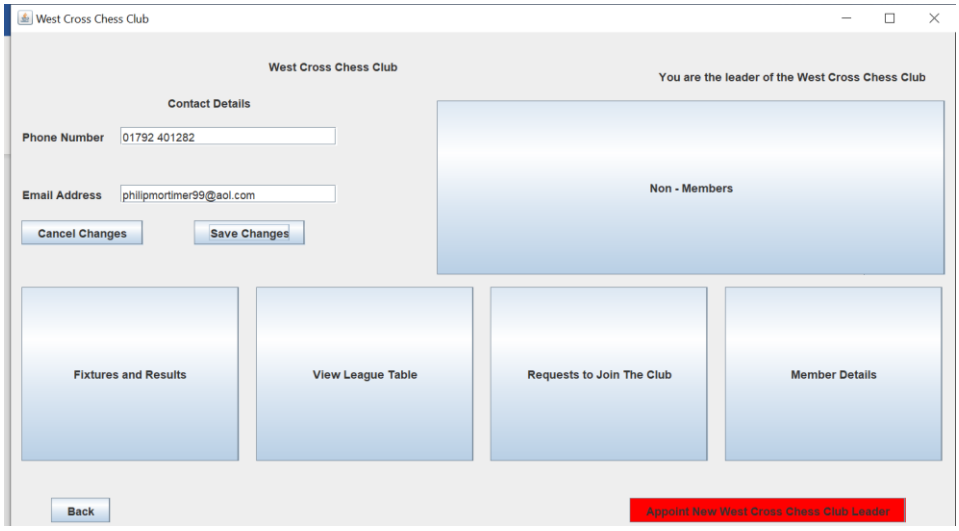
Members are taken to a form where they can see the results and fixtures for chess games. They can navigate between round by pressing the appropriate buttons at the bottom of the form. Each game is clearly labelled showing who is playing who and what the game state is.



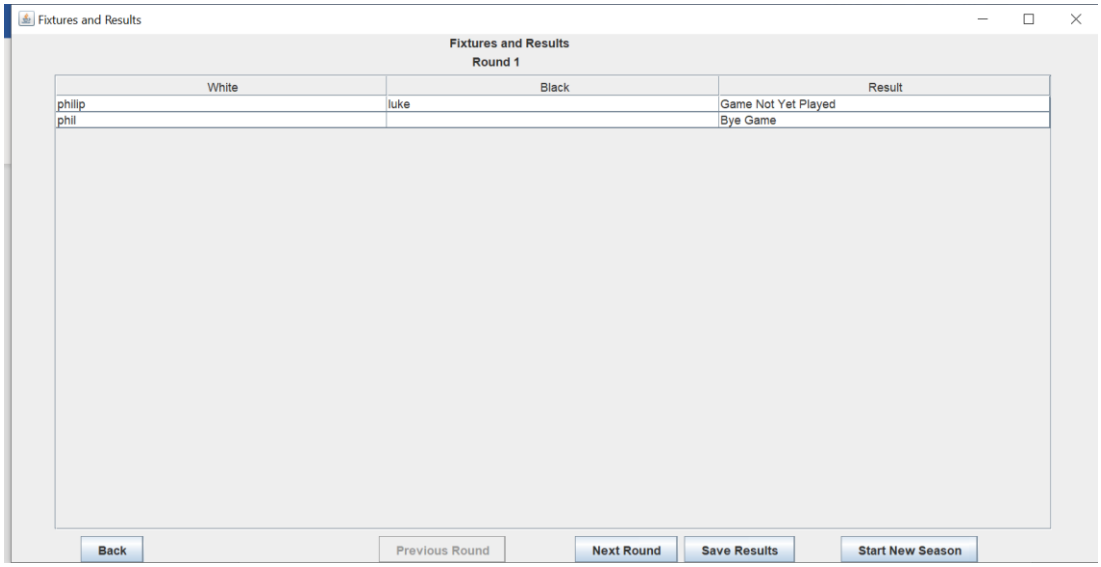
Members can see the league table that is generated from the fixtures / results. Both the fixtures and league table are clear and easy to understand, giving the members are clear view of how the season is happening.



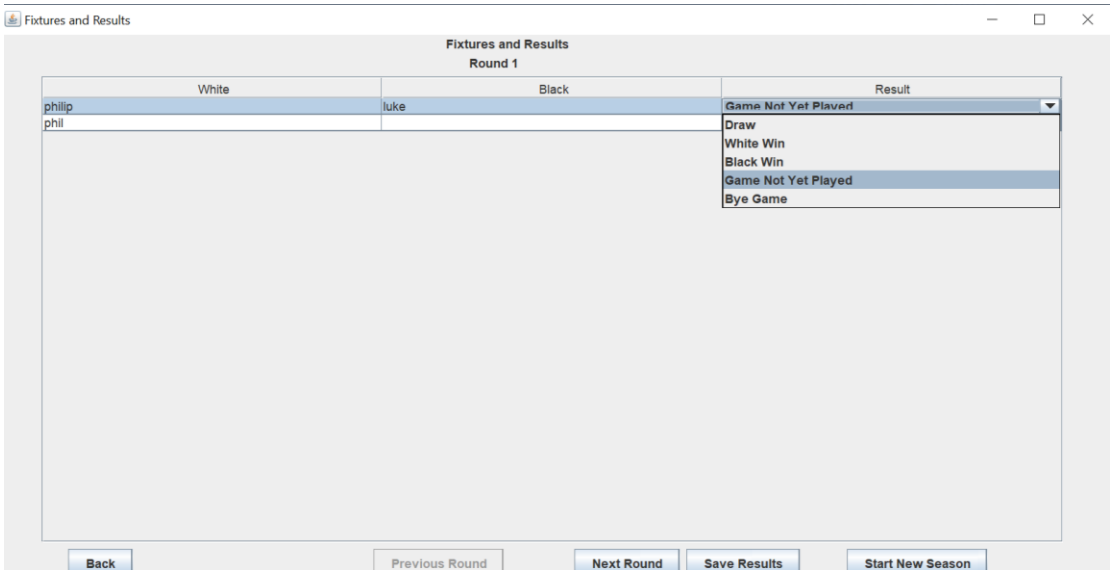
A similar design policy is employed for the leader form. All the main actions that the user will want to perform frequently are accessed through large buttons. The choice to appoint a new leader is marked in read and only occurs after the leader confirms that they wish to be replaced.



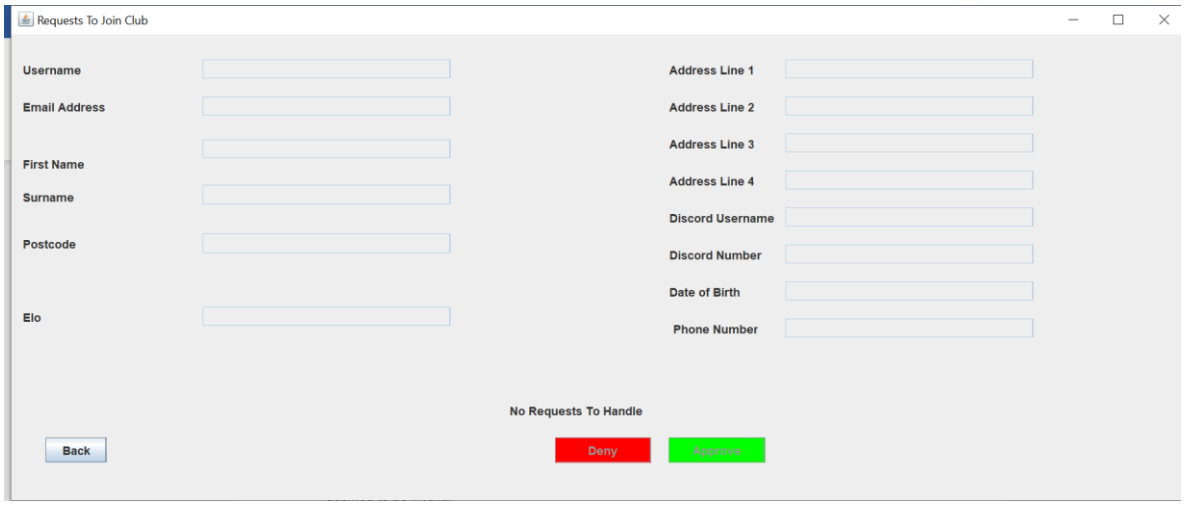
To change the contact details, the leader simply presses change contact details and types in the new values. The editing process is very easy with an option to save or cancel changes being presented to the user.



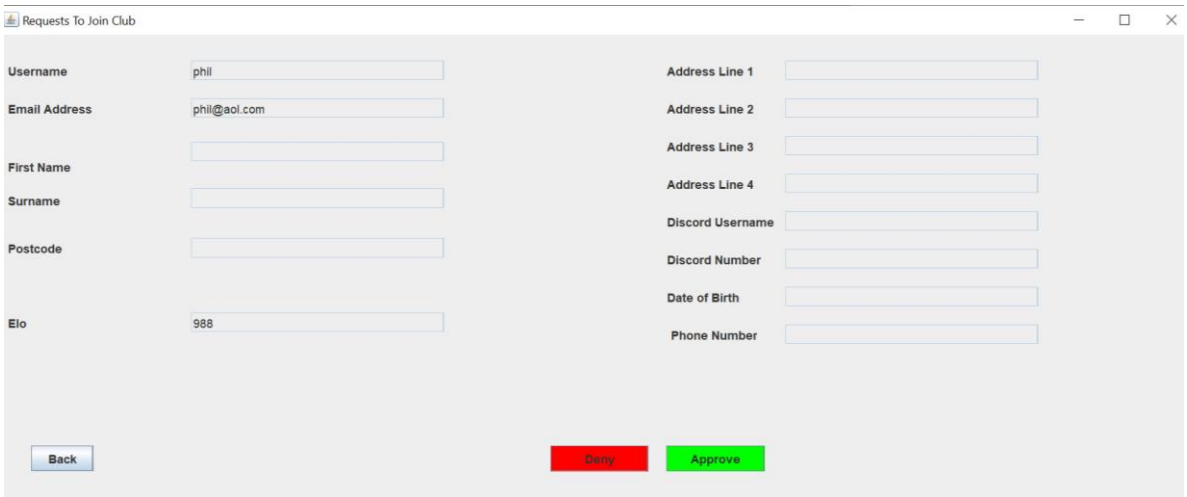
The leader sees the fixtures and results in a similar manner to a member. However, they are prevented with the option to generate all fixtures from scratch and start a completely new season.



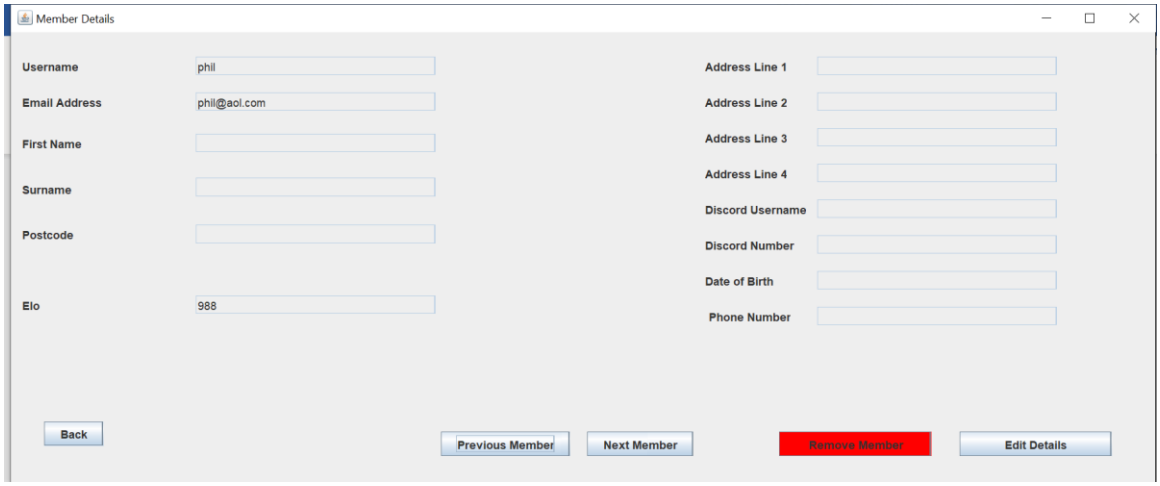
The leader edits fixture results by assigning a game state from one of the valid game states. This makes it very easy to select the correct game state and prevents erroneous input.



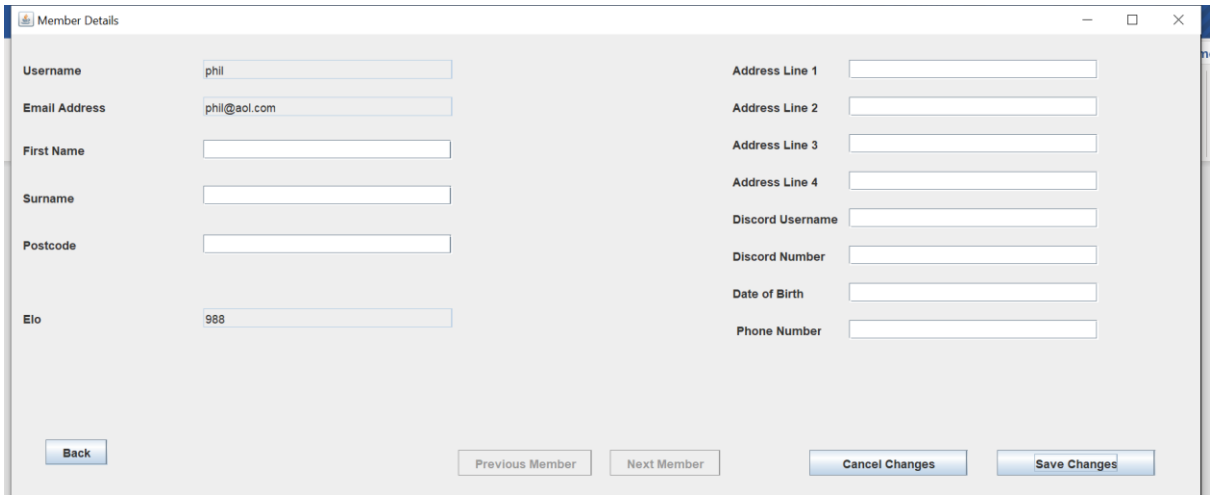
When there are no requests, the leader sees a form that clearly demonstrates this fact.



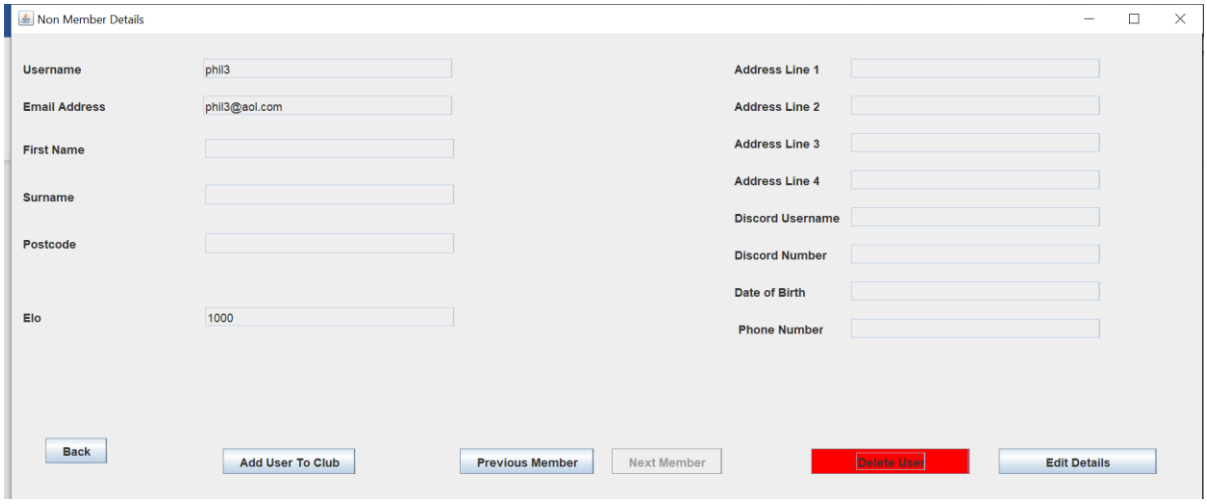
The leader is able to view all requests one by one and approve or deny them based on the user information. The accept and deny buttons are colour coordinated to aid form clarity and make it obvious and easy to accept or deny a request.



The leader is able to view the details of all club members and remove any member of the club. This button is highlighted red to ensure that it is not pressed accidentally. The leader is also asked to confirm that they would like to remove the member before they are actually removed.



When editing the details, it is made clear which fields are editable and the leader is given the choice to save or cancel the changes. They are unable to view other member details while in editing mode to prevent confusion that may occur from editing multiple users simultaneously.



The leader can also view details of users (referred to as Non-Members). The form navigation and editing process are identical. However, the leader can choose to add the user to the chess club or remove them from the system. Both of these options require confirmation before they are carried out. The delete user button is coloured red as the step is irreversible and therefore should not be made lightly.

<u>Validation Routines</u>

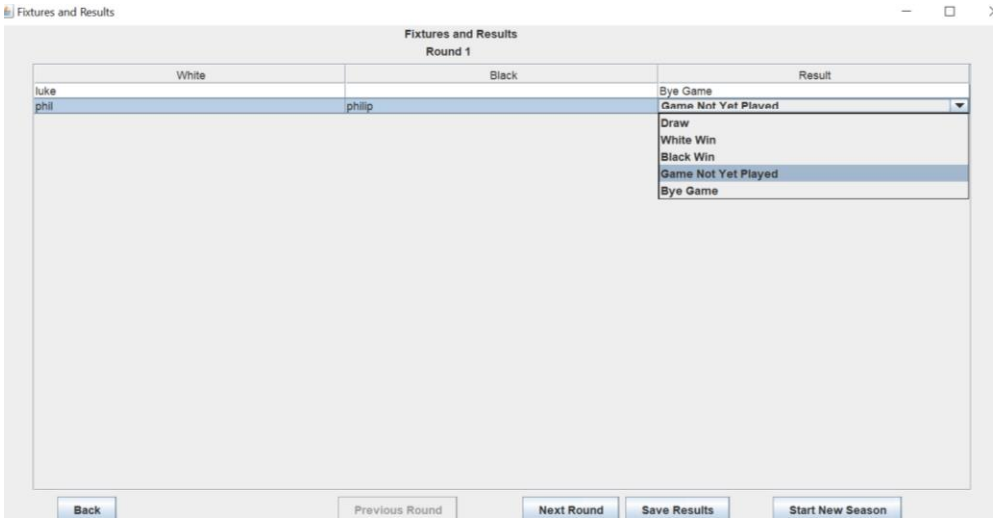
Validation is employed throughout my source code – so please view the full source code to see all validation routines. All validation routines are thoroughly annotated (as is all of the source code).

Validation routines identified in the source code will be highlighted light grey.

Despite, this I will highlight validation routines again here. In all validation cases, a detailed and specific error message is produced for invalid inputs, telling the user exactly why their input was deemed to be invalid.

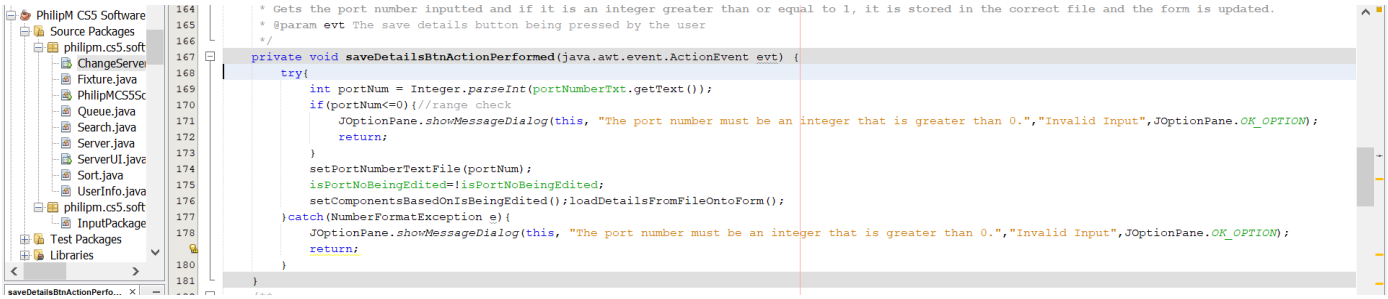
In general, my system minimises the need for validation through effective user interface design. For example, when playing chess, the user can only make valid moves.

(Notice that all valid moves are highlighted, and these are the only moves that the system accepts for moves. Hence, the user cannot make an invalid move.)



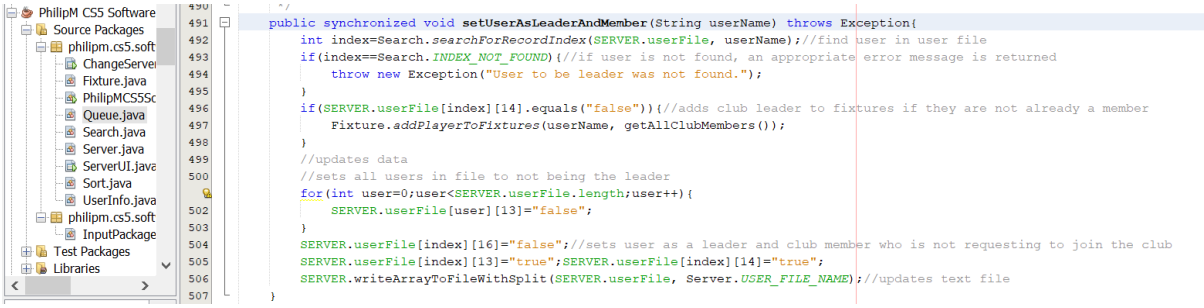
(The system only allows valid fixtures to be given valid game states. This is another example of how my UI is designed to be intuitive and minimise erroneous inputs).

When editing the port number that the server runs on, validation is performed to ensure that the port number entered is an integer greater than or equal to zero.



```
 * Gets the port number inputted and if it is an integer greater than or equal to 1, it is stored in the correct file and the form is updated.
 * @param evt The save details button being pressed by the user
 */
private void saveDetailsBtnActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        int portNum = Integer.parseInt(portNumberTxt.getText());
        if(portNum<=0){//range check
            JOptionPane.showMessageDialog(this, "The port number must be an integer that is greater than 0.","Invalid Input",JOptionPane.OK_OPTION);
            return;
        }
        setPortNumberTextFile(portNum);
        isPortNoBeingEdited=!isPortNoBeingEdited;
        setComponentsBasedOnIsBeingEdited();loadDetailsFromFileOntoForm();
    }catch(NumberFormatException e){
        JOptionPane.showMessageDialog(this, "The port number must be an integer that is greater than 0.","Invalid Input",JOptionPane.OK_OPTION);
        return;
    }
}
```

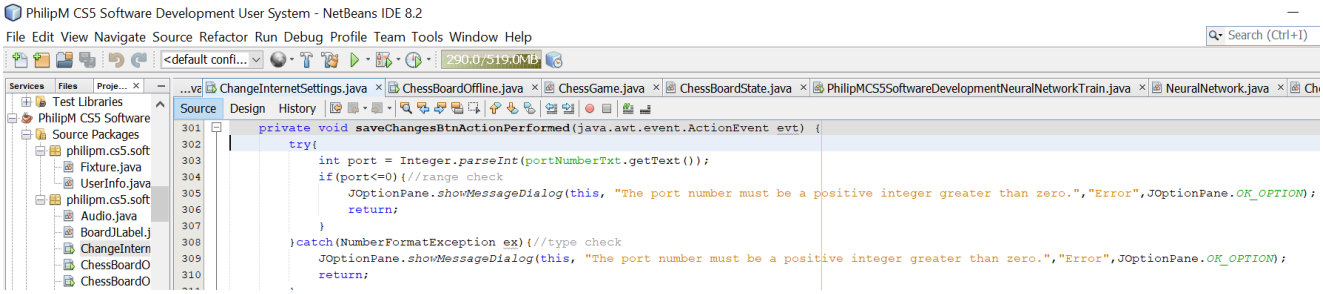
Additionally, when appointing the new West Cross Chess Club leader from the Server form, validation is performed to ensure that the username actually exists. This prevents an invalid user from being the leader. An exception is thrown if the leader chosen is invalid and they are not made leader.



```
public synchronized void setUserAsLeaderAndMember(String userName) throws Exception{
    int index=Search.searchForRecordIndex(SERVER.userFile, userName);//find user in user file
    if(index==Search.INDEX_NOT_FOUND){//if user is not found, an appropriate error message is returned
        throw new Exception("User to be leader was not found.");
    }
    if(SERVER.userFile[index][14].equals("false")){//adds club leader to fixtures if they are not already a member
        Fixture.addPlayerToFixtures(userName, getAllClubMembers());
    }
    //updates data
    //sets all users in file to not being the leader
    for(int user=0;user<SERVER.userFile.length;user++){
        SERVER.userFile[user][13]="false";
    }
    SERVER.userFile[index][16]="false";//sets user as a leader and club member who is not requesting to join the club
    SERVER.userFile[index][13]="true";SERVER.userFile[index][14]="true";
    SERVER.writeArrayToFileWithSplit(SERVER.userFile, Server.USER_FILE_NAME);//updates text file
}
```

Now, I shall look at validation in the user system.

When changing the internet settings in the user system, the port number is validated to ensure that it is an integer that is greater than or equal to 0.



When creating an account, the details entered are validated. These are validated using the validation details established in the design. Here is a reminder of those conditions.

Field Name	Data Type	Purpose	Length	Example
UserID	String	Used to uniquely identify user. Used when logging in to identify user.	No more than 15 characters	PhilipMortimer6
ClubName	String	Used to store the club that a user is part of. This is done to allow the system to determine which users are club members. Typically, this value will either be “West Cross Chess Club” or left blank. I have chosen to use a string value as opposed to a Boolean value to enable scalability of the system.	No more than 21 characters	West Cross Chess Club
Postcode	String	Used for safeguarding reasons by the club leader.	Between 6 and 8 characters (including both 6 and 8).	SA2 5RT
Email Address	String	Used as an alternative means of identifying a user. This is included as	Less than 40 characters	philipmortimer12@gmail.com

		some people struggle to remember their email address. The club leader also uses this information to contact club members.		
Password	String	Used to ensure the user account is not compromised and that only the account owner is able to log-in.	More than 5 characters and less than 20.	Ex@mplePas5w0rd23
First Name	String	Used by club leader to identify who each user actually is. A user's name will be displayed in the table if they have filled the field in.	No more than 35 characters	Philip
Surname	String	Used by club leader to identify who each user actually is. A user's name will be displayed in the table if they have filled the field in.	No more than 40 characters	Mortimer
Discord Username	String	Used by the club leader to coordinate voice chat for members.	No more than 20 characters	philipmo47
Discord Number	String	Used by the club leader to coordinate voice chat for members.	Exactly 4 characters	2344
Date of birth	String	Used by the club leader for	Exactly 10 characters	03/07/2003

		safeguarding purposes		
Address Line 1	String	Used by club leader for safeguarding purposes and for coordination of in-person chess sessions.	No more than 40 characters	3 Pennard Avenue
Address Line 2	String	Used by club leader for safeguarding purposes and for coordination of in-person chess sessions.	No more than 40 characters	West Cross
Address Line 3	String	Used by club leader for safeguarding purposes and for coordination of in-person chess sessions.	No more than 40 characters	Swansea
Address Line 4	String	Used by club leader for safeguarding purposes and for coordination of in-person chess sessions.	No more than 40 characters	Carmarthenshire
Phone Number	String	Used to replicate the old system and as a means to contact club members.	No more than 20 characters	01792 401272

```
    /**
     * Attempts to create an account after performing validation on the
appropriate fields
     * @param evt The button being pressed
     */
    private void
createAccountBtnActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        //validates the account details to ensure they are valid. Once
they have been validated, the account is created
        //presence check
        if(!(Validation.isPresent(usernameTxt.getText())&&
Validation.isPresent(emailAddressTxt.getText())&&
Validation.isPresent(new
String(passwordField.getPassword()))&&Validation.isPresent(new
String(confirmPasswordField.getPassword()))))){
            JOptionPane.showMessageDialog(this, "Username, email
address, password and confirm password must not be left blank.","Invalid
input",JOptionPane.OK_OPTION);
            return;
        }
        if(Validation.containsIllegalStrings(usernameTxt.getText()+email
AddressTxt.getText()+new String(passwordField.getPassword())+new
String(confirmPasswordField.getPassword())+firstNameTxt.getText()+surnam
eTxt.getText()+postcodeTxt.getText()+addressLineOneTxt.getText()
                +addressLineTwoTxt.getText()+addressLineThreeTxt.getText
()+addressLineFourTxt.getText()+discordNumberTxt.getText()+discordUserna
meTxt.getText()+dateOfBirthTxt.getText()+phoneNumberTxt.getText())){//pe
rforms a look up check to see whether any field contains an illegal
character
            JOptionPane.showMessageDialog(this, "One or more fields
contain an illegal character sequence. Here is a list of illegal
caharacter sequences: "+Validation.getListOfIllegalStrings(),"Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        //check username inputs
        if(!Validation.isAppropriateLength(15,
usernameTxt.getText())){//checks to see if username is 15 charcters or
less - length check
            JOptionPane.showMessageDialog(this, "The username can not
exceed 15 characters in length","Invalid Input",JOptionPane.OK_OPTION);
            return;
        }
        if(usernameTxt.getText().contains("@") ||
usernameTxt.getText().contains(" ")){//format check
            JOptionPane.showMessageDialog(this, "The username can not
contain the @ symbol OR a space.","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        //checks email address
        if(!emailAddressTxt.getText().contains("@") ||
emailAddressTxt.getText().contains(" ")){
            JOptionPane.showMessageDialog(this, "The email address must
contain the @ symbol and no spaces.","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        //length check
        if(!Validation.isAppropriateLength(39,
```

```
emailAddressTxt.getText())){//email address must not be longer than 39
characters
            JOptionPane.showMessageDialog(this, "The email address must
not be longer than 39 characters.","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!new String(passwordField.getPassword()).equals(new String
(confirmPasswordField.getPassword()))){//checks that both passwords
match - double entry verification validation
            JOptionPane.showMessageDialog(this, "The two passwords do
not match.","Invalid Input",JOptionPane.OK_OPTION);
            return;
        }
        if(new String(passwordField.getPassword()).length()<5 || new
String(passwordField.getPassword()).length()>20){//length check
            JOptionPane.showMessageDialog(this, "The password must be
between 5-20 characters in length (inclusive).","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(Validation.isPresent(firstNameTxt.getText())){//only performs
validation if first name is not empty
            if(!Validation.isAppropriateLength(35,
firstNameTxt.getText())){//length check - first name can't be more than
35 characters
                JOptionPane.showMessageDialog(this, "The first name can
not be longer than 35 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.consistsOnlyOfLetters(firstNameTxt.getText())
){
                JOptionPane.showMessageDialog(this, "The first name can
only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(Validation.isPresent(surnameTxt.getText())){//only performs
validation if  surname is not empty
            if(!Validation.isAppropriateLength(40,
surnameTxt.getText())){//length check - surname can't be more than 40
characters
                JOptionPane.showMessageDialog(this, "The surname can not
be longer than 40 characters","Invalid Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.consistsOnlyOfLetters(surnameTxt.getText())){
                JOptionPane.showMessageDialog(this, "The surname can
only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                return;
```

```
            }
        }
        if(Validation.isPresent(postcodeTxt.getText())){//only performs
validation is postcode is not left empty
            if(!Validation.isValidPostcode(postcodeTxt.getText())){
                JOptionPane.showMessageDialog(this, "The postcode must
be between 6-8 characters long, contain exactly one space and have all
non-space characters capitalised","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(!Validation.isAppropriateLength(40,
addressLineOneTxt.getText())){//length check
            JOptionPane.showMessageDialog(this, "Each address line can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!Validation.isAppropriateLength(40,
addressLineTwoTxt.getText())){//length check
            JOptionPane.showMessageDialog(this, "Each address line can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!Validation.isAppropriateLength(40,
addressLineThreeTxt.getText())){//length check
            JOptionPane.showMessageDialog(this, "Each address line can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!Validation.isAppropriateLength(40,
addressLineFourTxt.getText())){//length check
            JOptionPane.showMessageDialog(this, "Each address line can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!Validation.isAppropriateLength(32,
discordUsernameTxt.getText())){
            JOptionPane.showMessageDialog(this, "The Discord username
can not be longer than 32 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(Validation.isPresent(discordNumberTxt.getText())){
            if(!Validation.isInteger(discordNumberTxt.getText())){
                JOptionPane.showMessageDialog(this, "The Discord number
must be an integer","Invalid Input",JOptionPane.OK_OPTION);
                return;
            }
```

```
            if(!Validation.isAppropriateLength(4,
discordNumberTxt.getText())){
                JOptionPane.showMessageDialog(this, "The Discord number
must be no more than 4 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.rangeCheck(0, 9999,
Integer.parseInt(discordNumberTxt.getText()))){//range check
                JOptionPane.showMessageDialog(this, "The Discord number
must be between 0000-9999","Invalid Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(Validation.isPresent(dateOfBirthTxt.getText())){
            if(!Validation.formatCheckDateOfBirth(dateOfBirthTxt.getText
())){
                JOptionPane.showMessageDialog(this, "The date of birth
must be a valid date of birth in the form dd/mm/yyyy","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(Validation.isPresent(phoneNumberTxt.getText())){
            if(!Validation.isAppropriateLength(20,
phoneNumberTxt.getText())){
                JOptionPane.showMessageDialog(this, "Phone number can
not be longer than 20 characters.","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.onlyContainsPhoneNumberCharacters(phoneNumber
Txt.getText())){
                JOptionPane.showMessageDialog(this, "Phone number can
only contain plusses, spaces and integers.","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        //lookup check to ensure that both email address and username
are not already taken
        try{
            if(ThreadClient.isUserNameAlreadyTaken(usernameTxt.getText()
)){
                JOptionPane.showMessageDialog(this, "The username is
already taken. Plesae try a different username.","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(ThreadClient.isEmailAddressAlreadyTaken(emailAddressTxt.g
etText())){
                JOptionPane.showMessageDialog(this, "The email is
already taken. Plesae try a different email address.","Invalid
```

```
Input",JOptionPane.OK_OPTION);
                return;
            }
        }catch(Exception e){
            JOptionPane.showMessageDialog(this, "An error occured:
"+e.getMessage()+". This is almost certainly becuase something is wrong
with your internet connection or the server.","Network
Error",JOptionPane.OK_OPTION);
            return;
        }
        try{
            //creates the account once the validation has been performed
            ThreadClient.createAccount(new
String[]{usernameTxt.getText(),new
String(passwordField.getPassword()),firstNameTxt.getText(),surnameTxt.ge
tText(),postcodeTxt.getText(),addressLineOneTxt.getText(),addressLineTwo
Txt.getText(),addressLineThreeTxt.getText(),addressLineFourTxt.getText()
,discordUsernameTxt.getText(),discordNumberTxt.getText(),dateOfBirthTxt.
getText(),phoneNumberTxt.getText(),"false","false","1000","false"},email
AddressTxt.getText());
            JOptionPane.showMessageDialog(this,"Account succesfully
created.","Success", JOptionPane.OK_OPTION);
            LogIn log = new LogIn();//takes the user to the log in
screen once the account has been made
            log.setVisible(true);this.dispose();
            return;
        }catch(Exception e){
            JOptionPane.showMessageDialog(this, "An error occured:
"+e.getMessage()+". This is almost certainly becuase something is wrong
with your internet connection or the server.","Network
Error",JOptionPane.OK_OPTION);
            return;
        }
    }
```

When creating an online chess game against a friend, validation is performed to ensure that the time control settings have been chosen and that the username is valid and able to play more games.

```
        //creates a new game after performing relevant validation
        //gets time control to use
        int timeLeft=-100;
        if(fiveMinutesBtn.isSelected()){
            timeLeft=5*60;
        }else if(tenMinutesBtn.isSelected()){
            timeLeft=600;
        }else if(unlimitedBtn.isSelected()==false){
            JOptionPane.showMessageDialog(this,"Please select a time
control.","Error",JOptionPane.OK_OPTION);
            return;
```

```
        }
        String userName="";//stores the username of the opponent
        if(useCodeOrUsernameBtn.getText().equals("Create Code to Share
Instead")){
            userName=friendUserNameTxt.getText();
            if(!Validation.isPresent(userName) ||
userName.equals(session.getUsername())||
userName.equals(UserInfo.GUEST_ID)){//ensures that the username can't be
empty or be that of the current user
                JOptionPane.showMessageDialog(this,"Please input a valid
username.","Error",JOptionPane.OK_OPTION);
                return;
            }
            try{
                if(ThreadClient.isUserNameAlreadyTaken(userName)==false)
{//checks to see if username is taken
                    JOptionPane.showMessageDialog(this,"User not found.
Please enter  a valid username.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                if(ThreadClient.getOnlineGameData(userName)[2][0].equals
("")==false){//checks to see if friend is already playing three games
                    JOptionPane.showMessageDialog(this,"User selected is
currently playing three games. Users can not play more than three
games.","Error",JOptionPane.OK_OPTION);
                    return;
                }
            }catch(Exception e){
                JOptionPane.showMessageDialog(this, "There has been an
error when communicating with the
server.","Error",JOptionPane.OK_OPTION);
                return;
            }
        }
```

When creating an offline game against a computer opponent, validation is performed to ensure that the user has selected appropriate time settings and given the game a valid title (i.e., one that they have not used on another offline game).

```
        // creates a new game after checking that game settings have
actually been selected;
        //gets selected difficulty setting
        String computerDifficulty="";
        if(difficultyOneBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_ONE_COMP;
        }else if(difficultyTwoBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_TWO_COMP;
        }else if(difficultyThreeBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_THREE_COMP;
```

```
        }else if(difficultyFourBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_FOUR_COMP;
        }else if(difficultyFiveBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_FIVE_COMP;
        }else{
            //error message if no difficulty setting selected
            JOptionPane.showMessageDialog(this,"Please select a
difficulty setting.","Error",JOptionPane.OK_OPTION);
            return;
        }
        String gameTitle=gameTitleTxt.getText();//stores the game title
        if(isGameTitleValid(session,
gameTitle)==false||Validation.isPresent(gameTitle)==false){
            JOptionPane.showMessageDialog(this,"Please choose a valid,
unique game title.","Error",JOptionPane.OK_OPTION);
            return;
        }
        boolean isUserWhite=false;//stores which colour the user will
play as
        if(whiteBtn.isSelected()){
            isUserWhite=true;
        }else if(blackBtn.isSelected()){
            isUserWhite=false;
        }else if(randomBtn.isSelected()){
            //randomly assings user a colour
            if(Math.random()>=0.5){
                isUserWhite=true;
            }else{
                isUserWhite=false;
            }
        }else{
            //error message if no colour chosen
            JOptionPane.showMessageDialog(this,"Please select a colour
to play as.","Error",JOptionPane.OK_OPTION);
            return;
        }
        int timeControl=-100;
        if(unlimitedTimeBtn.isSelected()){
            timeControl=-100;
        }else if(fiveMinutesBtn.isSelected()){
            timeControl=5*60;
        }else if(tenMinsBtn.isSelected()){
            timeControl=10*60;
        }else{
            JOptionPane.showMessageDialog(this,"Please select a time
control setting.","Error",JOptionPane.OK_OPTION);
            return;
        }
```

```
    /**
     * Checks to see if the game title chosen is valid (i.e. unique and
not containing any illegal characters)
     * @param ses The session information
     * @param title The game title
     * @return True if valid, otherwise false
     */
    public static boolean isGameTitleValid(SessionInfo ses,String
title){
        if(Validation.containsIllegalStrings(title)){//checks that the
title does not contain illegal characters
            return false;
        }
        //searches to ensure that game title is unique
        try{
            FileReader read = new
FileReader(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+ses.getUserna
me()+".txt");
            BufferedReader buffRead = new BufferedReader(read);
            String lineRead;
            while((lineRead=buffRead.readLine())!=null){
                if(lineRead.split(UserInfo.SPLIT_ITEM)[0].equals(title))
{
                    return false;
                }
            }buffRead.close();read.close();
        }catch(IOException e){
            JOptionPane.showMessageDialog(null,"An unexpected error
occured "+e);
        }
        return true;
    }
```

When creating an offline game against a person, validation is performed to ensure that the game title is valid and that the settings have actually been selected.

```
        // creates a new game after checking that game settings have
actually been selected;
        //gets opponent's name
        String opponentName=opponentNameTxt.getText();
        //ensures that opponent name is not a computer id
        if(opponentName.equals(OfflinePlay.LEVEL_ONE_COMP)||opponentName
.equals(OfflinePlay.LEVEL_TWO_COMP)||opponentName.equals(OfflinePlay.LEV
EL_THREE_COMP)||opponentName.equals(OfflinePlay.LEVEL_FOUR_COMP)||oppone
ntName.equals(OfflinePlay.LEVEL_FIVE_COMP)||Validation.containsIllegalSt
rings(opponentName)){
            JOptionPane.showMessageDialog(this,"Invalid opponent name.
Please change the opponent name.","Error",JOptionPane.OK_OPTION);
            return;
```

```
        }
        boolean isUserWhite=false;//stores which colour the user will
play as
        if(whiteBtn.isSelected()){
            isUserWhite=true;
        }else if(blackBtn.isSelected()){
            isUserWhite=false;
        }else if(randomBtn.isSelected()){
            //randomly assings user a colour
            if(Math.random()>=0.5){
                isUserWhite=true;
            }else{
                isUserWhite=false;
            }
        }else{
            //error message if no colour chosen
            JOptionPane.showMessageDialog(this,"Please select a colour
to play as.","Error",JOptionPane.OK_OPTION);
            return;
        }
        String gameTitle=gameTitleTxt.getText();
        if(CreateOfflineGame.isGameTitleValid(session,
gameTitle)==false||Validation.isPresent(gameTitle)==false){
            JOptionPane.showMessageDialog(this,"Please select a valid
and unique title.","Error",JOptionPane.OK_OPTION);
            return;
        }
        int timeLeft=-100;
        if(unlimitedBtn.isSelected()){
            timeLeft=-100;
        }else if(fiveMinutesBtn.isSelected()){
            timeLeft=5*60;
        }else if(tenMinutesBtn.isSelected()){
            timeLeft =10*60;
        }else{
            JOptionPane.showMessageDialog(this,"Please select a time
control setting.","Error",JOptionPane.OK_OPTION);
            return;
        }
```

When logging in, validation is performed in order to check whether an invalid log in is being attempted. Note: this validation is not as strict as the validation for account creation. This is justified as the message that will return for a failed log with an invalid username is “The username was not found.”. Hence, there is no need to extensively validate the field as the end user results will be identical.

```
        //logs user in via server
        //validates input somewhat before querying server
        if(usernameOrEmailAddressTxt.getText().equals(UserInfo.GUEST_ID)
```

```
||
usernameOrEmailAddressTxt.getText().equals(UserInfo.GUEST_ID_EMAIL)){//b
anned inputs reserved for guest ids
            JOptionPane.showMessageDialog(this,"You can't log in with
this ID.","Error",JOptionPane.OK_OPTION);
            return;
        }
        if(usernameOrEmailAddressTxt.getText().contains("@")){//for
email address log in
            try {
                //checks to see if email address exists
                if(ThreadClient.isEmailAddressAlreadyTaken(usernameOrEma
ilAddressTxt.getText())==false){
                    JOptionPane.showMessageDialog(this, "The email
address was not found.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                //attempts to log user in
                UserInfo user =
ThreadClient.logInWithEmailAddress(usernameOrEmailAddressTxt.getText(),
new String(passwordTxt.getPassword()));
                if(user.getIsPasswordCorrect()==false){//if password
does not match email
                    JOptionPane.showMessageDialog(this, "The password is
incorrect.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                //takes user to menu if log in was succesful
                Menu menu = new Menu(new SessionInfo(user.getUserName(),
user.getIsClubMember(),
user.getIsClubLeader()));menu.setVisible(true);this.dispose();return;//t
akes user to menu with appropriate session information
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(this, "A network error has
occured. This may be due to your network settings or the server.
"+ex,"Error",JOptionPane.OK_OPTION);
                return;
            }
        }else{//for username
            try{
                //attempts to log user in with username
                UserInfo user =
ThreadClient.logInWithUserName(usernameOrEmailAddressTxt.getText(), new
String(passwordTxt.getPassword()));//gets user info from server
                if(user.getUserFound()==false){//displays appropraite
message if the username was not found
                    JOptionPane.showMessageDialog(this, "The username
was not found.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                if(user.getIsPasswordCorrect()==false){//displays
appropraite message if the password entered was wrong
                    JOptionPane.showMessageDialog(this, "The password is
```

```
incorrect.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                Menu menu = new Menu(new SessionInfo(user.getUserName(),
user.getIsClubMember(),
user.getIsClubLeader()));menu.setVisible(true);this.dispose();return;//t
akes user to menu with appropriate session information
                //takes user to menu having created session info object
to store relevant data
            }catch(Exception e){
                JOptionPane.showMessageDialog(this, "A network error has
occured. This may be due to your network settings or the server.
"+e,"Error",JOptionPane.OK_OPTION);
                return;
            }
        }
```

When the club leader edits user details, validation is also performed to ensure that the edited details are valid. This validation routine is the same as the validation performed on user details when creating an account.

```
            //checks to see that changes are valid
            //presence check
            if(Validation.containsIllegalStrings(usernameTxt.getText()+e
mailAddressTxt.getText()+firstNameTxt.getText()+surnameTxt.getText()+pos
tcodeTxt.getText()+addressLineOneTxt.getText()
                    +addressLineTwoTxt.getText()+addressLineThreeTxt.get
Text()+addressLineFourTxt.getText()+discordNumberTxt.getText()+discordUs
ernameTxt.getText()+dateOfBirthTxt.getText()+phoneNumberTxt.getText())){
//performs a look up check to see whether any field contains an illegal
character
                JOptionPane.showMessageDialog(this, "One or more fields
contain an illegal character sequence. Here is a list of illegal
caharacter sequences: "+Validation.getListOfIllegalStrings(),"Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(Validation.isPresent(firstNameTxt.getText())){//only
performs validation if first name is not empty
                if(!Validation.isAppropriateLength(35,
firstNameTxt.getText())){//length check - first name can't be more than
35 characters
                    JOptionPane.showMessageDialog(this, "The first name
can not be longer than 35 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.consistsOnlyOfLetters(firstNameTxt.getTex
t())){
```

```
                    JOptionPane.showMessageDialog(this, "The first name
can only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(surnameTxt.getText())){//only
performs validation if  surname is not empty
                if(!Validation.isAppropriateLength(40,
surnameTxt.getText())){//length check - surname can't be more than 40
characters
                    JOptionPane.showMessageDialog(this, "The surname can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.consistsOnlyOfLetters(surnameTxt.getText(
)){
                    JOptionPane.showMessageDialog(this, "The surname can
only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(postcodeTxt.getText())){//only
performs validation is postcode is not left empty
                if(!Validation.isValidPostcode(postcodeTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The postcode
must be between 6-8 characters long, contain exactly one space and have
all non-space characters capitalised","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(!Validation.isAppropriateLength(40,
addressLineOneTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineTwoTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineThreeTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
```

```
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineFourTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(32,
discordUsernameTxt.getText())){
                JOptionPane.showMessageDialog(this, "The Discord
username can not be longer than 32 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(Validation.isPresent(discordNumberTxt.getText())){
                if(!Validation.isInteger(discordNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The Discord
number must be an integer","Invalid Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.isAppropriateLength(4,
discordNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The Discord
number must be no more than 4 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.rangeCheck(0, 9999,
Integer.parseInt(discordNumberTxt.getText()))){//range check
                    JOptionPane.showMessageDialog(this, "The Discord
number must be between 0000-9999","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(dateOfBirthTxt.getText())){
                if(!Validation.formatCheckDateOfBirth(dateOfBirthTxt.get
Text())){
                    JOptionPane.showMessageDialog(this, "The date of
birth must be a valid date of birth in the form dd/mm/yyyy","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(phoneNumberTxt.getText())){
                if(!Validation.isAppropriateLength(20,
phoneNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "Phone number
can not be longer than 20 characters.","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
```

```
                }
                if(!Validation.onlyContainsPhoneNumberCharacters(phoneNu
mberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "Phone number
can only contain plusses, spaces and integers.","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
```

Similarly, when editing non-member details, the exact same validation is performed.

```
            //checks to see that changes are valid
            //presence check
            if(Validation.containsIllegalStrings(usernameTxt.getText()+e
mailAddressTxt.getText()+firstNameTxt.getText()+surnameTxt.getText()+pos
tcodeTxt.getText()+addressLineOneTxt.getText()
                    +addressLineTwoTxt.getText()+addressLineThreeTxt.get
Text()+addressLineFourTxt.getText()+discordNumberTxt.getText()+discordUs
ernameTxt.getText()+dateOfBirthTxt.getText()+phoneNumberTxt.getText())){
//performs a look up check to see whether any field contains an illegal
character
                JOptionPane.showMessageDialog(this, "One or more fields
contain an illegal character sequence. Here is a list of illegal
caharacter sequences: "+Validation.getListOfIllegalStrings(),"Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(Validation.isPresent(firstNameTxt.getText())){//only
performs validation if first name is not empty
                if(!Validation.isAppropriateLength(35,
firstNameTxt.getText())){//length check - first name can't be more than
35 characters
                    JOptionPane.showMessageDialog(this, "The first name
can not be longer than 35 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.consistsOnlyOfLetters(firstNameTxt.getTex
t())){
                    JOptionPane.showMessageDialog(this, "The first name
can only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(surnameTxt.getText())){//only
performs validation if  surname is not empty
                if(!Validation.isAppropriateLength(40,
surnameTxt.getText())){//length check - surname can't be more than 40
```

```
characters
                    JOptionPane.showMessageDialog(this, "The surname can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.consistsOnlyOfLetters(surnameTxt.getText(
)){
                    JOptionPane.showMessageDialog(this, "The surname can
only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(postcodeTxt.getText())){//only
performs validation is postcode is not left empty
                if(!Validation.isValidPostcode(postcodeTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The postcode
must be between 6-8 characters long, contain exactly one space and have
all non-space characters capitalised","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(!Validation.isAppropriateLength(40,
addressLineOneTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineTwoTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineThreeTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineFourTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(32,
```

```
discordUsernameTxt.getText())){
                JOptionPane.showMessageDialog(this, "The Discord
username can not be longer than 32 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(Validation.isPresent(discordNumberTxt.getText())){
                if(!Validation.isInteger(discordNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The Discord
number must be an integer","Invalid Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.isAppropriateLength(4,
discordNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The Discord
number must be no more than 4 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.rangeCheck(0, 9999,
Integer.parseInt(discordNumberTxt.getText())){//range check
                    JOptionPane.showMessageDialog(this, "The Discord
number must be between 0000-9999","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(dateOfBirthTxt.getText())){
                if(!Validation.formatCheckDateOfBirth(dateOfBirthTxt.get
Text())){
                    JOptionPane.showMessageDialog(this, "The date of
birth must be a valid date of birth in the form dd/mm/yyyy","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(phoneNumberTxt.getText())){
                if(!Validation.isAppropriateLength(20,
phoneNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "Phone number
can not be longer than 20 characters.","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.onlyContainsPhoneNumberCharacters(phoneNu
mberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "Phone number
can only contain plusses, spaces and integers.","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
```

In the offline menu, validation is performed to ensure that only valid games are played or deleted. Valid games are classed as games that have been created (and hence are playable).

```
private void deleteGameBtnActionPerformed(java.awt.event.ActionEvent evt) {
    //deletes game stored if it is a valid game to delete
    if(offlineGamesTbl.getSelectedRow()==-1 || gameData[offlineGamesTbl.getSelectedRow()][0]==null){//checks to make sure a game is actually selected
        JOptionPane.showMessageDialog(this, "Please select a valid game to delete.","No Game Selected",JOptionPane.OK_OPTION);
        return;
    }
```

```
    //takes user to selected game, if it is a valid game
    if(offlineGamesTbl.getSelectedRow()==-1 || gameData[offlineGamesTbl.getSelectedRow()][0]==null){//checks to make sure a game is actually selected
        JOptionPane.showMessageDialog(this, "Please select a valid game to play.","No Game Selected",JOptionPane.OK_OPTION);
        return;
    }
```

When the user attempts to create any new online game, validation is performed in the online menu to ensure that they are able to create a new game. They are only able to create a new game if they have two or fewer ongoing online games.

```
//checks to ensure that the user is in two or fewer games currently
try{
    Integer.parseInt(String.valueOf(onlineGamesTbl.getValueAt(2, 0)));
    JOptionPane.showMessageDialog(this, "You can only play up to three online games at any one time.","Error",JOptionPane.OK_OPTION);
    return;
}catch(NumberFormatException e){
}
```

```
//takes user to form to create game against a friend
if(onlineGamesTbl.getValueAt(2,0).equals("")==false){
    JOptionPane.showMessageDialog(this, "You can only have three games at any one time.","Error",JOptionPane.OK_OPTION);
    return;
}
```

```
// allows user to input game code and play against friend
if(onlineGamesTbl.getValueAt(2,0).equals("")==false){//ensures that user does not join more than three games
    JOptionPane.showMessageDialog(this, "You can only have three games at any one time.","Error",JOptionPane.OK_OPTION);
    return;
}
```

Additionally, when attempting to join a game using a game code, validation is performed to ensure that the code is an integer and a valid code.

```
        // allows user to input game code and play against friend
        if(onlineGamesTbl.getValueAt(2,0).equals("")==false){//ensures
that user does not join more than three games
            JOptionPane.showMessageDialog(this, "You can only have three
games at any one time.","Error",JOptionPane.OK_OPTION);
            return;
        }
        String code =JOptionPane.showInputDialog(this,"Please enter the
game code provided by a friend:","Enter
code",JOptionPane.OK_CANCEL_OPTION);
        if(code==null){//if cancel is pressed, nothing happens
            return;
        }
        try{
            int codeInt=Integer.parseInt(code);//tries converting code
to an integer
            if(codeInt<0){
                //type and range check to ensure that the game code is
possibly valid
                JOptionPane.showMessageDialog(this,"The game code must
be a valid integer greater than 0","Invalid
code",JOptionPane.OK_OPTION);
                return;
            }
```

```
            try{
                boolean succ =
ThreadClient.joinGameUsingFriendCode(codeInt,
session.getUsername());//attempts to join the game
                if(succ==false){
                    JOptionPane.showMessageDialog(this,"Unable to join
game. The code entered is invalid or can't be used to join a
game","Error",JOptionPane.OK_OPTION);
                    updateTable();
                    return;
                }
                updateTable();//updates the table
            }catch(Exception e){
                 JOptionPane.showMessageDialog(this,"Error when
communicating with the server. Please check your network settings. Error
details: "+e.getMessage(),"Error",JOptionPane.OK_OPTION);
                 updateTable();
                 return;
            }
        }catch(NumberFormatException e){
            JOptionPane.showMessageDialog(this,"The game code must be a
valid integer greater than 0","Invalid code",JOptionPane.OK_OPTION);
            return;
        }
```

Validation is also performed when trying to play and delete an online game to ensure that the game can be deleted.

```
       // takes the user to the selected game
        if(onlineGamesTbl.getSelectedRow()<0 ||
onlineGamesTbl.getSelectedRow()>2){//ensures that a valid game is
selected
            JOptionPane.showMessageDialog(this, "Invalid game
selected.","Invalid Game",JOptionPane.OK_OPTION);
            return;
        }
        //checks to ensure game selected is playable
        if(onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow(),
0).equals("") ||
onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow(),
1).equals("")){
            JOptionPane.showMessageDialog(this, "Invalid game
selected.","Invalid Game",JOptionPane.OK_OPTION);
            return;
        }
        //gets the most recent version of the data from the server
        String []data=null;
        try{
            data=ThreadClient.getGameData(Integer.parseInt(String.valueO
```

```
f(onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow(),0))));//gets
updated version of game record before joining
        }catch(Exception e){
            JOptionPane.showMessageDialog(this, "Error when
communicating with the server. Error details: "+e,"Invalid
Game",JOptionPane.OK_OPTION);
            return;
        }
        //checks to see if game is already over - if it is, an
appropriate message is displayed and the game is deleted
        boolean isUserPlayerOne=data[1].equals(session.getUsername());
        if(isUserPlayerOne && data[8].equals("false") &&
data[9].equals("true")){
            if(data[11].equals("1")){
                JOptionPane.showMessageDialog(this,"You won this
game.","Win",JOptionPane.OK_OPTION);
            }else if(data[11].equals("0.5")){
                JOptionPane.showMessageDialog(this,"This game was a
draw.","Draw",JOptionPane.OK_OPTION);
            }else{
                JOptionPane.showMessageDialog(this,"You lost this
game.","Loss",JOptionPane.OK_OPTION);
            }
            data[8]="true";
            try{
                ThreadClient.updateOnlineGameRecord(data);
            }catch(Exception e){
                JOptionPane.showMessageDialog(this, "Error
when  communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
                updateTable();
                return;
            }
            updateTable();
            return;
        }else if(!isUserPlayerOne && data[9].equals("false") &&
data[8].equals("true")){
            if(data[11].equals("1")){
                JOptionPane.showMessageDialog(this,"You lost this
game.","Loss",JOptionPane.OK_OPTION);
            }else if(data[11].equals("0.5")){
                JOptionPane.showMessageDialog(this,"This game was a
draw.","Draw",JOptionPane.OK_OPTION);
            }else{
                JOptionPane.showMessageDialog(this,"You won this
game.","Win",JOptionPane.OK_OPTION);
            }
            data[9]="true";
            try{
                ThreadClient.updateOnlineGameRecord(data);
            }catch(Exception e){
                JOptionPane.showMessageDialog(this, "Error
when  communicating with the server. Error details:
```

```
"+e,"Error",JOptionPane.OK_OPTION);
                updateTable();
                return;
            }
            updateTable();
            return;
        }
        //games is playable, hence user is taken to it
        //creates the chess game object
        String fenData[]=data[3].split(OfflinePlay.SPLIT_FEN_STRING);
        ChessBoardState[]statesSoFar=new
ChessBoardState[fenData.length];
        for(int board=0;board<fenData.length;board++){
            statesSoFar[board]=new ChessBoardState(fenData[board]);
        }
        //takes the user to play the game
        ChessGame game = new ChessGame(statesSoFar, false);
         ChessBoardOnline   onlineGame = new
ChessBoardOnline(session,game,data);onlineGame.setVisible(true);this.dis
pose();
```

```
        //attempts to delete a game that can be deleted.
        //only games where the opponent has net yet joined can be
deleted
        if(onlineGamesTbl.getSelectedRow()>=0 &&
onlineGamesTbl.getSelectedRow()<=2){
            if(onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow()
,1).equals("")==false){
                JOptionPane.showMessageDialog(this,"You can only delete
games that an opponent has not yet joined. Please go into the game and
resign if you wish to remove the game.","Unable to
Delete",JOptionPane.OK_OPTION);
                return;
            }
            //asks for confirmation before deleting games
            int confirm=JOptionPane.showConfirmDialog(this,"Are you sure
you would like to delete this
game?","Delete?",JOptionPane.YES_NO_CANCEL_OPTION);
            if(confirm!=JOptionPane.YES_OPTION){
                return;
            }
            try{
                ThreadClient.deleteOnlineGame(Integer.parseInt(String.va
lueOf(onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow(),0))),
0.5);//deletes game
                updateTable();//updates the tab;e
            }catch(Exception  e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server. Please check your network settings. Error
```

```
details: "+e.getMessage(),"Error",JOptionPane.OK_OPTION);
                updateTable();
                return;
            }
        }else{
            JOptionPane.showMessageDialog(this, "Please select a valid
game to delete","Error",JOptionPane.OK_OPTION);
            return;
        }
```

My Validation class consists of static methods which are used for a lot of my system's validation.

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import philipm.cs5.software.development.server.system.UserInfo;

/**
 * This class performs various validation routines
 * @author mortimer
 */
public class Validation {
    private static final
String[]ILLEGAL_CHAR_SEQUENCES={UserInfo.SPLIT_ITEM,OfflinePlay.SPLIT_FE
N_STRING, UserInfo.GUEST_ID,UserInfo.GUEST_ID_EMAIL};//stores all
illegeal characters sequences
    /**
     * Checks to see whether input string is empty or not. Returns false
if length of String is 0 or a NullPointerException occurs (e.g. for
newly intialised String arrays java).
     * Otherwise true is returned. Note: this presence check views a
String with a (or any) single space(s) in it as
     * a non-empty input and returns true for it.
     * @param stringToCheck the string to be checked
     * @return returns false if the String is empty (i.e. has a length
of 0, using the .length() command or triggers a NullPointerException
caused by String arrays that have been initialised but have no value
assigned to them) and true if the string is not empty
     */
    public static boolean isPresent(String stringToCheck){//presence
check
        try{
            if(stringToCheck.length()==0){//checks to see if number of
characters are zero e.g.:""
                return false;
```

```
            }else{
                return true;
            }
        }catch(java.lang.NullPointerException e){//nullpointer exception
triggered for newly initialised elements in a string array
            return false;
        }
    }
    /**
     * Checks to see if String is of desired length. If String is less
than or equal to desired length, true is returned. Otherwise, false is
returned.
     * @param maxLengthAllowed the maximum length of the string
     * @param stringToCheck the string to perform the length check on
     * @return returns true if the length of the string is less than or
equal to the max length allowed and false if not.
     */
    public static boolean isAppropriateLength(int
maxLengthAllowed,String stringToCheck){//length check
        return !(stringToCheck.length()>maxLengthAllowed);
    }
    /**
     * A simple format check to see whether a string is in the form
dd/mm/yyyy.It checks to see that there are integers and dashes in the
correct places.
     * It also checks to see if the month field is greater than twelve
and if the day field is greater than 31 (although invalid inputs can
occur, for example if the day is 31 and the month is february. If a
valid date is not entered, false is returned.
     * @param stringToCheck the string to check
     * @return true if the inputted string is considered a valid date in
the form dd/mm/yyyy.
     */
    public static boolean formatCheckDateOfBirth(String
stringToCheck){//format check
        return((stringToCheck.matches("([0-9]{2})/([0-9]{2})/([0-
9]{4})"))&&(Integer.parseInt(String.valueOf(stringToCheck.charAt(3))+Str
ing.valueOf(stringToCheck.charAt(4)))<=12)&&(Integer.parseInt(String.val
ueOf(stringToCheck.charAt(0))+String.valueOf(stringToCheck.charAt(1)))<=
31));
    }
    /**
     * Tries converting the input string into an integer using the
Integer.parseInt(). If a NumberFormatException is thrown, it is not an
integer and false is returned, else true is returned.
     * @param stringToCheck the string to check
     * @return true is returned if the string is an integer, false is
returned if the string is not an integer
     */
    public static boolean isInteger(String stringToCheck){//type check
        try{
            int val = Integer.parseInt(stringToCheck);
            return true;
```

```
        }catch(NumberFormatException e){
            return false;
        }
    }
    /**
     * Checks to see if the integer to check is with in the range
specified. True is returned if in the range, else false
     * @param lowerLimitInclusice this is the lower integer limit for
range. Checks to see if integerToCheck is greater than OR equal to this
number
     * @param upperLimitInclusive this is the upper integer limit for
range. Checks to see if integerToCheck is less than OR equal to this
number
     * @param integerToCheck the integer number to see if it is with in
the specified range
     * @return True is returned if integerToCheck is greater than or
equal to lowerLimitInclusive AND if integerToCheck is less than or equal
to upperLimitInclusive. Otherwise, false is returned
     */
    public static boolean rangeCheck(int lowerLimitInclusice,int
upperLimitInclusive,int integerToCheck){
        return (integerToCheck>=lowerLimitInclusice &&
integerToCheck<=upperLimitInclusive);
    }
    /**
     * Performs a linear search through all acceptable inputs comparing
it to the String input. If a match is found, true is returned, else
false is returned.
     * @param acceptableInputs A String array containing all acceptable
inputs
     * @param stringToCheck The String to check against the acceptable
inputs
     * @return True is returned if stringToCheck matches any item in the
array, otherwise false is returned.
     */
    public static boolean lookUpCheck(String[]acceptableInputs,String
stringToCheck){//lookup check
        for(int
indexInAcceptableInputs=0;indexInAcceptableInputs<acceptableInputs.lengt
h;indexInAcceptableInputs++){
            if(stringToCheck.equals(acceptableInputs[indexInAcceptableIn
puts])){
                return true;
            }
        }
        return false;
    }
    /**
     * Checks to see if the length of the string is equal to the desired
length.
     * @param correctLength The desired length of the string
     * @param stringToCheck the string to have its length checked
     * @return True is returned if the length of the string equals
```

```
correctLength, else false.
     */
    public static boolean isCorrectLength(int correctLength,String
stringToCheck){
        return (stringToCheck.length()==correctLength);
    }
    /**
     * Checks whether an integer is greater than or equal to another
integer
     * @param numberThatIntegerShouldBeGreaterThanOrEqual The value that
numberToCheck should be greater than or equal to in order to return true
     * @param numberToCheck The number that is being checked to see if
it is greater than or equal to
numberThatIntegerShouldBeGreaterThanOrEqual
     * @return True is returned if numberToCheck is greater than or
equal to numberThatIntegerShouldBeGreaterThanOrEqual, otherwise false is
returned
     */
    public static boolean isgreaterThanOrEqual(int
numberThatIntegerShouldBeGreaterThanOrEqual,int numberToCheck){
        return(numberToCheck>=numberThatIntegerShouldBeGreaterThanOrEqua
l);
    }
    /**
     * Checks the string to see if it contains any of strings that are
not allowed. For example, if commas are used to separate items, commas
may be banned.
     * @param itemToCheck The string to check
     * @return true is returned if it contains any illegal characters /
strings. Otherwise, false is returned.
     */
    public static boolean containsIllegalStrings(String itemToCheck){
        for(int i=0;i<ILLEGAL_CHAR_SEQUENCES.length;i++){
            if(itemToCheck.contains(ILLEGAL_CHAR_SEQUENCES[i])){
                return true;
            }
        }
        return false;
    }
    /**
     * Returns all illegal character sequences as a string
     * @return The illegal characters sequence as a presentable string
     */
    public static String getListOfIllegalStrings(){
        String ill="";
        for(int i=0;i<ILLEGAL_CHAR_SEQUENCES.length;i++){
            ill=ill+ILLEGAL_CHAR_SEQUENCES[i];
        }
        return ill;
    }
    /**
     * Checks to see whether the string only contains characters from
the English alphabet.
```

```
     * @param stringToCheck The string to be checked
     * @return True is returned if the string consists only of letters
in the English alphabet. Otherwise, false is returned.
     */
    public static boolean consistsOnlyOfLetters(String stringToCheck){
        return stringToCheck.matches("[a-zA-Z]+");
    }
    /**
     * Checks to see if the postcode is valid.
     * @param postCode The postcode as a string
     * @return True is returned if the postcode is between 6 and 8
characters, contains exactly one space and all characters other than a
space character are capitalised.
     */
    public static boolean isValidPostcode(String postCode){
        if(postCode.length()>8 || postCode.length()<6){//length checks
            return false;
        }
        int noOfSpaces=0;//counts number of spaces
        for(int c=0;c<postCode.length();c++){
            if(postCode.charAt(c)==' '){
                if(noOfSpaces==1){
                    return false;
                }
                noOfSpaces++;
            }else{
                if(!Character.isUpperCase(postCode.charAt(c)) &&
Character.isDigit(postCode.charAt(c))==false){
                    return false;
                }
            }
        }
        if(noOfSpaces==0){
            return false;
        }
        return true;
    }
    /**
     * Checks if the string only contains +, spaces and numbers.
     * @param phoneNumber The phone number
     * @return True is returned if the string consists only of plusses,
spaces and digits. Otehrwise, false is returned.
     */
    public static boolean onlyContainsPhoneNumberCharacters(String
phoneNumber){
        for(int c=0;c<phoneNumber.length();c++){
            if(phoneNumber.charAt(c)!=' '&&
phoneNumber.charAt(c)!='+'&&!Character.isDigit(phoneNumber.charAt(c))){
                return false;
            }
        }
        return true;
    }
```

```
}
```

When the current club leader selects a new club leader, validation is performed to ensure that the username of the new leader is a valid username.

```
        // appoints a new chess club leader
        //asks for comfirmation before making the appointment
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you want to make somebody else the leader? This will mean that you will
no longer be the leader.", "Proceed?",
JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            //appoints somebody else as the chess club leader
            String newLeader = JOptionPane.showInputDialog(this,"Please
input the username of the new leader:");
            if(newLeader==null){//if the user pressses cancel on the
input dialog boc, nothing should occur
                return;
            }
            try{
                //performs validation
                if(ThreadClient.isUserNameAlreadyTaken(newLeader)==false
){//checks to see that the username chosen is actually valid
                    JOptionPane.showMessageDialog(this,"This user does
not exist. Please input a real user.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                if(newLeader.equals(session.getUsername())){//if the new
leader is just the same as the current leader, nothing needs to occur
                    return;
                }
                ThreadClient.setUserAsClubLeader(newLeader);//sets the
new leader
                session.setIsClubLeader(false);//updates the session
data
                JOptionPane.showMessageDialog(this,"You are no longer
the club leader.","Success",JOptionPane.OK_OPTION);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();//takes the user back
to the menu
                return;
            }catch(Exception e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server. "+e,"Error",JOptionPane.OK_OPTION);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
                return;
            }
```

```
        }else{
            return;
        }
```

Source Code

In this section, I will include the source code to my solution. The solution was coded using Java and the NetBeans 8.2 IDE. The project files will be included with the submission, so code may also be viewed using that. Code can also be viewed in each of the relevant NetBeans project folders as ".html" files. In the server folder, simply view the folder called "Server code". Similarly, one can view "User code" and "Neural code". I recommend viewing the source code in the ".html" files or the actual NetBeans IDE. The rest of this software development document is just a copy of the source code.

The source code is fully annotated, allowing for easy maintenance and expansion of the code by a competent developer. Additionally, I have generated the Javadoc's for each project. This allows for easy and effective viewing of my annotated listings. These can be found in each project under the folder title "javadoc".

For my "PhilipM CS5 Software Development User System", the first method call is to the main method of PhilipMCS5SoftwareDevelopmentUserSystem.java". For my "PhilipM CS5 Software Development Server System", the first method call is to the main method of PhilipMCS5SoftwareDevelopmentServerSystem.java". For my "PhilipM CS5 Software Development Neural Network train", the first method call is to the main method of PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain.java".

Users will run the code by executing the ".jar" file in the folder.

Source Code – User System

First, I will be including the source code for "PhilipM CS5 Software Development User System". This project has two packages: "philip.m.cs5.software.development.user.system" and "philip.m.cs5.software.development.server.system". Here is the code for the "philip.m.cs5.software.development.server.system" package in this project.

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
```

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

/**
 * This static class is used to interact with the fixture and results
stored.
 * @author mortimer
 */
public class Fixture {
    //stores various constants used thorughout the fixture code- such as
values that represent a win or draw
    public static final String FILE_NAME="fixtures.txt";
    public static final String SPLIT=UserInfo.SPLIT_ITEM;
    public static final String GAME_NOT_YET_PLAYED="-1";
    private static final String UNABLE_TO_CREATE_GAME="cantCreateGame";
    public static final String BYE_GAME_STATE="-2";//indicates that the
game is a bye. Does not necesarily mean that game is scheduled against a
bye opponent
    public static final String BY_NAME="@";
    private static final String NO_PLAYERS_TO_ADD="aasifoakofako325";
    public static final String TOO_FEW_PLAYERS_ARRAY="Unable to load
array too few players";
    public static final String GAME_WHITE_WIN="whiteW";
    public static final String GAME_BLACK_WIN="blackW";
    public static final String GAME_DRAW="draww";
    /**
     * Creates a new set of fixtures from scratch with the status of not
having been played
     * @param players The name of all players. They must all be unique.
     * @throws Exception An exception if the fixtures can't be generated
due to player size
     */
    public static void createdNewFixturesFromStart(String[]players)
throws Exception{
        if(players.length==0 || players.length==1){
            //stores the fact that there are too few players to play a
game
            try{
                FileWriter w = new FileWriter(FILE_NAME,false);
                BufferedWriter buffW = new BufferedWriter(w);
                buffW.write(UNABLE_TO_CREATE_GAME);
                buffW.flush();w.flush();
                buffW.close();w.close();
                throw new Exception("Unable to create game as too few
players are present");
            }catch(IOException e){
                System.out.println(e);
            }
        }
        if(players.length % 2 == 1){
            String newNames[]=new String[players.length+1];//creates a
```

```
new array and adds a bye to the list (as only an even number of players
can exist in a head-to-head league.
            //copies old array to new one
            for(int i=0;i<players.length;i++){
                newNames[i]=players[i];
            }
            newNames[players.length]=BY_NAME;
            players=newNames;
        }
        //randomly swaps names in list
        String buffer;int index;
        Random rnd = new Random();
        for(int i=0;i<players.length;i++){
            index=rnd.nextInt(players.length);
            buffer = players[index];
            players[index]=players[i];
            players[i]=buffer;
        }
        String[][][]fixtures=new String[2*(players.length  -
1)][3][players.length/2];//stores all fixtures.
        //the 3d array is indexed fixtures[z][y][x]. z represents the
round number, y represents some attribute in the round.
        //For example, y = 0 represents all white players. y=1
represents black players. y=2 represents the game state.
        //x represents the match element in the round. E.g.
fixtures[0][0][0] return the player playing as white in the nought round
and in the nought fixture
        //sets all games to have not yet been played
        for(int round=0;round<fixtures.length;round++){
            for(int game=0;game<players.length/2;game++){
                fixtures[round][2][game]=GAME_NOT_YET_PLAYED;
            }
        }
        //generates fixtures for first round
        for(int y=0;y<2;y++){
            for(int x=0;x<players.length/2;x++){
                fixtures[0][y][x]=players[y*(players.length/2) +x];
            }
        }
        //loads first half of fixtures
        for(int round=1;round<fixtures.length/2;round++){//fills in the
rest of the round
            //pushes all of top to the right apart from first element,
which is fixed
            fixtures[round][0][0]=fixtures[0][0][0];
            for(int x=1;x<players.length/2 -1;x++){
                fixtures[round][0][x+1]=fixtures[round-1][0][x];
            }
            //adds in buffer at appropriate spot
            fixtures[round][0][1]=fixtures[round-1][1][0];
            //pushes all of bottom to the left
            for(int x=0;x<players.length/2 -1;x++){
                fixtures[round][1][x]=fixtures[round-1][1][x+1];
```

```
            }
            fixtures[round][1][players.length/2 -1]=fixtures[round-
1][0][players.length/2 -1];
        }
        //second half of fixtures is the same as the first half just
swapping them around
        for(int round=fixtures.length/2;round<fixtures.length;round++){
            for(int x=0;x<players.length/2;x++){
                fixtures[round][0][x]=fixtures[round-
(fixtures.length/2)][1][x];
                fixtures[round][1][x]=fixtures[round-
(fixtures.length/2)][0][x];
            }
        }
        //shuffles the games in each round randomly and then shuffles
the rounds randomly
        String[][]bufferArr;
        for(int round=0;round<fixtures.length;round++){
            //shuffles games within round randomly
            for(int game=0;game<players.length/2;game++){
                index=rnd.nextInt(players.length/2);
                buffer=fixtures[round][0][game];
                fixtures[round][0][game]=fixtures[round][0][index];
                fixtures[round][0][index]=buffer;
                buffer=fixtures[round][1][game];
                fixtures[round][1][game]=fixtures[round][1][index];
                fixtures[round][1][index]=buffer;
            }
            //shuffles the rounds randomly
            index=rnd.nextInt(fixtures.length);
            bufferArr=fixtures[index];
            fixtures[index]=fixtures[round];
            fixtures[round]=bufferArr;
        }
        for(int round=0;round<fixtures.length;round++){
            for(int game=0;game<players.length/2;game++){
                if(fixtures[round][0][game].equals(BY_NAME) ||
fixtures[round][1][game].equals(BY_NAME)){//stores games which are byes
                    fixtures[round][2][game]=BYE_GAME_STATE;
                }
            }

        }
        //writes generated fixtures to a file, overwriting anything that
was previously there, in a form that the computer can intepret
        writeToFile(fixtures, new String[0]);
    }
    /**
     * Writes the fixtures list to the file.
     * @param fixtures The fixtures array to write
     * @param nameOfPlayersAddedLate The players added halfway through
the season
     * @throws IOException An IOException may occur when writing to file
```

```
     */
    public static void writeToFile(String fixtures[][][], String[]
nameOfPlayersAddedLate) throws IOException{
            int playerLength=fixtures[0][0].length*2;
            FileWriter write = new FileWriter(FILE_NAME,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write(String.valueOf(playerLength));buffWrite.newL
ine();
            //writes fixtures for white
            for(int round=0;round<fixtures.length;round++){
                for(int game=0;game<playerLength/2;game++){
                    if(round==0 && game==0){
                        buffWrite.write(fixtures[round][0][game]);
                    }else{
                        buffWrite.write(SPLIT+fixtures[round][0][game]);
                    }
                }
            }
            buffWrite.newLine();
            //writes fixtures for black
            for(int round=0;round<fixtures.length;round++){
                for(int game=0;game<playerLength/2;game++){
                    if(round==0 && game==0){
                        buffWrite.write(fixtures[round][1][game]);
                    }else{
                        buffWrite.write(SPLIT+fixtures[round][1][game]);
                    }
                }
            }
            buffWrite.newLine();
            //writes game states to file
            for(int round=0;round<fixtures.length;round++){
                for(int game=0;game<playerLength/2;game++){
                     if(round==0 && game==0){
                        buffWrite.write(fixtures[round][2][game]);
                    }else{
                        buffWrite.write(SPLIT+fixtures[round][2][game]);
                    }
                }
            }
            //writes queue of players to add to file
            buffWrite.newLine();
            if(nameOfPlayersAddedLate.length==0){
                buffWrite.write(NO_PLAYERS_TO_ADD);
            }else{
                String lineWrite=nameOfPlayersAddedLate[0];
                for(String name: nameOfPlayersAddedLate){
                    lineWrite = lineWrite+SPLIT+name;
                }
                buffWrite.write(lineWrite);
            }
            buffWrite.flush();write.flush();
            buffWrite.close();write.close();
```

```
    }
    /**
     * Gets an array containing the people who wish to play in the
league
     * @return The array
     * @throws IOException An exception when reading from the text file
     */
    public static String[]getListOfPeopleToAddedDuringSeason()throws
IOException{
        FileReader read = new FileReader(FILE_NAME);
        BufferedReader buffRead = new BufferedReader(read);
        buffRead.readLine();buffRead.readLine();buffRead.readLine();buff
Read.readLine();
        String line = buffRead.readLine();
        buffRead.close();read.close();
        if(NO_PLAYERS_TO_ADD.equals(line)){
            return new String[0];
        }
        return line.split(SPLIT);
    }
    /**
     * Loads the fixtures array from the text file.
     * @return The array of fixtures
     * @throws java.lang.Exception An exception if the array can't be
loaded due to the number of players being too small.
     */
    public static String[][][]loadFixturesArray() throws Exception{
        try{
            FileReader read = new FileReader(FILE_NAME);
            BufferedReader buffRead = new BufferedReader(read);
            String lin=buffRead.readLine();
            if(lin.equals(UNABLE_TO_CREATE_GAME)){//if there are too few
players and hence no fixture has been created, an exception is thrown
                buffRead.close();read.close();
                throw new Exception(TOO_FEW_PLAYERS_ARRAY);
            }
            int lengthOfFile = Integer.parseInt(lin);
            String[][][]fixtures = new String[2*(lengthOfFile -
1)][3][lengthOfFile/2];//stores the fixtures / results
            String line[]=buffRead.readLine().split(SPLIT);int
lineIndex=0;//stores the white players
            for(int round=0;round<fixtures.length;round++){//loops
through file updating all white players
                for(int game=0;game<lengthOfFile/2;game++){
                    fixtures[round][0][game]=line[lineIndex++];
                }
            }
            line=buffRead.readLine().split(SPLIT);lineIndex=0;//stores
all black players
            for(int round=0;round<fixtures.length;round++){//loops
through array to store all black players
                for(int game=0;game<lengthOfFile/2;game++){
                    fixtures[round][1][game]=line[lineIndex++];
```

```
                }
            }
            line=buffRead.readLine().split(SPLIT);lineIndex=0;//stores
the game states of each game
            for(int round=0;round<fixtures.length;round++){
                for(int game=0;game<lengthOfFile/2;game++){
                    fixtures[round][2][game]=line[lineIndex++];
                }
            }
            buffRead.close();read.close();
            return fixtures;
        }catch(IOException e){
            System.out.println("Error "+e);
            throw new Exception("Error "+e);
        }
    }
    /**
     * Adds a player who is not part of the fixtures already to the
league. If no games have been played, the fixtures will be generated
from scratch. Otherwise, they cannot actually play until a new season
begins.
     * @param playerName The player name
     * @param allClubMembers An array of all club member names
(INCLUDING PLAYER TO BE ADDED)
     * @throws Exception An exception may occur when reading and writing
to text files
     */
    public static void addPlayerToFixtures(String playerName, String[]
allClubMembers) throws Exception{
        /*
        If no games have been played yet, then just regernate fixtures
from scratch. If the data is stored as too few players
        then generate fixtures from scratch also.

        Else, player is added to list of players who cannot play until
next season.
        */
        try{
            String[][][]fixtures = loadFixturesArray();
            //checks to see if any games have been played
            boolean hasMatchBeenPlayed=false;
            for(int round=0;round<fixtures.length;round++){
                for(int match=0;match<fixtures[0][0].length;match++){
                    if((fixtures[round][2][match].equals(BYE_GAME_STATE)
|| fixtures[round][2][match].equals(GAME_NOT_YET_PLAYED))==false){
                        hasMatchBeenPlayed=true;
                        break;
                    }
                }
                if(hasMatchBeenPlayed){
                    break;
                }
            }
```

```
            if(hasMatchBeenPlayed){
                //adds player to a list of people added halfway through
season
                String[]playersAdded=getListOfPeopleToAddedDuringSeason(
);
                String playersNew[]=new String[playersAdded.length+1];
                for(int i=0;i<playersAdded.length;i++){
                    playersNew[i]=playersAdded[i];
                }
                playersNew[playersAdded.length]=playerName;
                writeToFile(fixtures, playersNew);
                return;
            }else{
                //generates the fixtures from scratch as no games have
been played
                createdNewFixturesFromStart(allClubMembers);
                return;
            }
        }catch(Exception e){
            //catches exceptions. An exception may be thrown due to file
handling. If this is the case, this is a genuine error and needs to be
thrown.
            //The error could simply be that fixtures cannot be loaded
as there are too few players to generate any fixtures. In this case,
fixtures are generated again.
            if(TOO_FEW_PLAYERS_ARRAY.equals(e.getMessage())){
                createdNewFixturesFromStart(allClubMembers);
                return;
            }else{
                throw e;
            }
        }

    }
    /**
     * Checks to see if the player is part of the league (or waiting to
be) and removes them if they are. If they appear in the fixtures, those
games will be marked as bye games.
     * @param playerName The name of the player
     * @throws Exception An exception may occur when reading to and from
a text file
     */
    public static void removePlayerFromFixtures(String playerName)throws
Exception{
        try{
            String fixtures[][][]=loadFixturesArray();
            //checks to see if player is in fixture. If they are, all
their matches are set to bye games (even the ones already played)
            boolean playerFound=false;
            for(int round=0;round<fixtures.length;round++){
                for(int match=0;match<fixtures[0][0].length;match++){
                    if(fixtures[round][0][match].equals(playerName)){
                        fixtures[round][0][match]=BY_NAME;
```

```
                        fixtures[round][2][match]=BYE_GAME_STATE;
                        playerFound=true;
                    }else
if(fixtures[round][1][match].equals(playerName)){
                        fixtures[round][1][match]=BY_NAME;
                        fixtures[round][2][match]=BYE_GAME_STATE;
                        playerFound=true;
                    }
                }
            }
            if(playerFound){
                //writes the new fixtures to file
                writeToFile(fixtures,
getListOfPeopleToAddedDuringSeason());
                return;
            }else{
                //checks to see if player was added mid way through
season.
                int indexOfPlayer=-1;
                String[]people=getListOfPeopleToAddedDuringSeason();
                for(int i=0;i<people.length;i++){
                    if(people[i].equals(playerName)){
                        indexOfPlayer=i;
                        break;
                    }
                }
                if(indexOfPlayer==-1){
                    return ;//no action is needed
                }else{
                    //removes player from list of people waiting
                    String peopleNew[]=new String[people.length-1];
                    for(int i=0;i<indexOfPlayer;i++){
                        peopleNew[i]=people[i];
                    }
                    for(int i=indexOfPlayer+1;i<people.length;i++){
                        peopleNew[i-1]=people[i];
                    }
                    //writes updated file
                    writeToFile(fixtures, peopleNew);
                    return;
                }
            }
        }catch(Exception e){
            //an exception may occur if there are too few players to
create fixtures. In this case, no action is needed.
            if(TOO_FEW_PLAYERS_ARRAY.equals(e.getMessage())){
                return;
            }else{
                throw e;
            }
        }
```

```
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

import java.io.Serializable;

/**
 * This is a class that creates a UserInfo Object. This object is
Serializable, allowing to be sent across a stream.
 * @author mortimer
 */
public class UserInfo implements Serializable{
    public static final int WIDTH_OF_USER_DETAILS_FILE=17;//stores the
width of the user file
    private String[]userDetails;//stores the user record
    private boolean isPasswordCorrect=false;//stores whether the
password is correct
    private boolean userFound;//stores whether the user with a given
name was found in the user file
    public static final int WIDTH_OF_EMAIL_LINK_FILE=2;//stores the
width of the email link file
    public static  transient final String
GUEST_ID="GUEST_ID9239923";//stores the guest id
    public static transient final String
GUEST_ID_EMAIL="GuestEmail9932d3@l.com";//stores the guest email address
    private boolean emailAddressFound=true;//stores whether the email
address was found
    public static final String SPLIT_ITEM=";lO9";//used to split data
items
    /*structure of user text FILE:
    Username, password, first name, surname, postcode, addressline one,
address line two, address line three, address line four, discord
username, discord number, date of birth, phone number, isClubLeader,
isClubMember, ELO, hasUserRequestedToJoinClub
    */
    /**
     * Creates a new User with the record found and the password entered
     * @param record The user file record
     * @param userEnteredPasssword The password the user entered to
access the account
     */
    public UserInfo(String[]record,String userEnteredPasssword){
```

```
        userFound = record[0]!=null;//updates whether the user was found
        userDetails = record;//stores the record
        if(userFound){
            //checks to see if the password is correct
            isPasswordCorrect=userEnteredPasssword.equals(record[1]);
        }
    }
    /**
     * Creates a new UserInfo object that indicates that the chosen
email address was not found.
     * @return The UserInfoObject
     */
    public static UserInfo CreateUserInfoEmailNotFound(){
        return new UserInfo();
    }
    /**
     * Called privately to create a user info when email address is not
found
     */
    private UserInfo(){
        emailAddressFound=false;
    }
    /**
     * Gets whether the email address was found
     * @return whether the email address was found
     */
    public boolean getEmailAddressFound(){
        return emailAddressFound;
    }
    /**
     * Gets the password associated with the user
     * @return Returns the password
     */
    public String getPassword(){
        return userDetails[1];
    }
    /**
     * Gets whether the user was found
     * @return whether the user was found
     */
    public boolean getUserFound(){
        return userFound;
    }
    /**
     * Gets whether the password entered was correct
     * @return whether the password entered was correct
     */
    public boolean getIsPasswordCorrect(){
        return isPasswordCorrect;
    }
    /**
     * Returns the user name stored.
     * @return The username
```

```
     */
    public String getUserName(){
        return userDetails[0];
    }
    /**
     * Returns whether the user is a club member.
     * @return Whether the user is  a club member
     */
    public boolean getIsClubMember(){
        return userDetails[14].equals("true");
    }
    /**
     * Returns whether the user is the club leader or not
     * @return Whether the user is club leader or not.
     */
    public boolean getIsClubLeader(){
        return userDetails[13].equals("true");
    }
    /**
     * Returns the ELO rating of the user
     * @return The ELO rating
     */
    public int getElo(){
        return Integer.parseInt(userDetails[15]);
    }
}
```

The next code also comes from the user system but this times it's from the "philipm.cs5.software.development.user.system" package.

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.File;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;


/**
 * Plays desired sounds
 * @author mortimer
 */
public class Audio{
```

```
    /**
     * Plays the specified sound file. The sound method is threaded
     * @param PATH The file name
     */
    public static synchronized void playSound(final String PATH) {
       //creates new thread that plays the desired sound clip
      new Thread(new Runnable() {
          /**
           * Plays the chosen audio clip
           */
        @Override
        public void run() {
          try {
            Clip clip = AudioSystem.getClip();
            AudioInputStream inputStream =
AudioSystem.getAudioInputStream(
              new File(PATH));
            clip.open(inputStream);
            clip.start();
          } catch (Exception e) {

          }
        }
      }).start();
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import javax.swing.JLabel;

/**
 * Custom JLabel that renders to a  higher quality
 * @author mortimer
 */
public class BoardJLabel extends JLabel {
    /**
     * Creates a new BoardJLabel.
     */
    public BoardJLabel(){
        super();//creates a standard JLabel
```

```
    }
    /**
     * Paints the component using the highest possible quality rendering
settings.
     * @param g The graphics
     */
    @Override
    public void paintComponent(Graphics g) {
        //creates a custom paint component that ensures high quality
rendering options are used when rendering the JLabel
        Graphics2D g2d = (Graphics2D) g;
                g2d.setRenderingHint(RenderingHints.KEY_ALPHA_INTERPOLAT
ION, RenderingHints.VALUE_ALPHA_INTERPOLATION_QUALITY);
                g2d.setRenderingHint(RenderingHints.KEY_COLOR_RENDERING,
RenderingHints.VALUE_COLOR_RENDER_QUALITY);
                g2d.setRenderingHint(RenderingHints.KEY_DITHERING,
RenderingHints.VALUE_DITHER_ENABLE);
                g2d.setRenderingHint(RenderingHints.KEY_FRACTIONALMETRIC
S, RenderingHints.VALUE_FRACTIONALMETRICS_ON);
                g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BICUBIC);
                g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2d.setRenderingHint(RenderingHints.KEY_RENDERING,
RenderingHints.VALUE_RENDER_QUALITY);
        super.paintComponent(g2d);
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;

/**
 * Creates a new form which allows the user to change the Internet
settings (i.e. the server address and port number).
 * @author mortimer
 */
```

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

    /**
     * Creates new form ChangeInternetSettings
     */
    private boolean isInEditMode=false;//stores whether the values are
currently being edited
    private static final String DEFAULT_PORT_NUMBER="80";//stores
default port number
    private static final String
DEFUALT_SERVER_ADDRESS="86.129.93.149";//stores defualt address of
server
    private String bufferPortNumber;//stores the most recent port number
    private String bufferServerAddress;//stores the most recent server
address
    private static final String PORT_NUMBER_FILE_NAME="portNumber.txt";
    private static final String
SERVER_ADDRESS_FILE_NAME="serverAddress.txt";
    public ChangeInternetSettings() {
        //initilaises components and updates form based on stored
address and port number
        initComponents();
        updateFormBasedOnWhetherInEditingMode();
        loadPortNumAndServerAddressOnForm();
        bufferPortNumber=portNumberTxt.getText();bufferServerAddress=ser
verAddressTxt.getText();
    }
    /**
     * Loads the port number and server address onto the form
     */
    private void loadPortNumAndServerAddressOnForm(){
        portNumberTxt.setText(getPortNumberString());
        serverAddressTxt.setText(getServerAddressNonStatic());
    }
    /**
     * Gets the port number as a string
     * @return returns the port number stored in the text file. An empty
string is returned if an error occurs.
     */
    public String getPortNumberString(){
        try{
            FileReader read = new
FileReader(PORT_NUMBER_FILE_NAME);//reads the port number from the file
            BufferedReader buffRead = new BufferedReader(read);
            String ret = buffRead.readLine();
            buffRead.close();read.close();
            return ret;
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An error occured with
"+PORT_NUMBER_FILE_NAME, "Error", JOptionPane.OK_OPTION);
        }
        return "";
    }
```

```
    /**
     * Gets the port number
     * @return The port number as an integer. If an error occurs, an
error message is displayed and -1 is returned.
     */
    public static int getPortNumber(){
        try{
            FileReader read = new
FileReader(PORT_NUMBER_FILE_NAME);//reads the port number from the file
            BufferedReader buffRead = new BufferedReader(read);
            String portStr=buffRead.readLine();
            buffRead.close();read.close();
            try{
                int port = Integer.parseInt(portStr);
                return port;
            }catch(NumberFormatException ex){
                JOptionPane.showMessageDialog(null, "The port number
stored is not an integer. Please fix this by going into settings.",
"Error", JOptionPane.OK_OPTION);
                return -1;
            }
        }catch(IOException e){
            JOptionPane.showMessageDialog(null, "An error occured with
"+PORT_NUMBER_FILE_NAME+". Error details: "+e, "Error",
JOptionPane.OK_OPTION);
        }
        return -1;
    }
    /**
     * Gets the server address
     * @return The server address. If an error occurs, an appropriate
message is displayed and an empty string is returned.
     */
    public static String getServerAddress(){
        try{
            FileReader read = new
FileReader(SERVER_ADDRESS_FILE_NAME);//reads the server address from the
file
            BufferedReader buffRead = new BufferedReader(read);
            String ret = buffRead.readLine();
            buffRead.close();read.close();
            return ret;
        }catch(IOException e){
            JOptionPane.showMessageDialog(null, "An error occured with
"+SERVER_ADDRESS_FILE_NAME+". Error details: "+e, "Error",
JOptionPane.OK_OPTION);
            return "";
        }
    }
    /**
     * Gets the server address
     * @return The server address. If an error occurs, an appropriate
message is displayed and an empty string is returned.
```

```
     */
    private String getServerAddressNonStatic(){
        try{
            FileReader read = new
FileReader(SERVER_ADDRESS_FILE_NAME);//reads the server address from the
file
            BufferedReader buffRead = new BufferedReader(read);
            String ret = buffRead.readLine();
            buffRead.close();read.close();
            return ret;
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An error occured with
"+SERVER_ADDRESS_FILE_NAME+". Error details: "+e, "Error",
JOptionPane.OK_OPTION);
            return "";
        }
    }
    /**
     * Alters relevant attributes of various buttons and text fields
based on whether the system is currently in editing mode
     */
    private void updateFormBasedOnWhetherInEditingMode(){
        editBtn.setEnabled(!isInEditMode);editBtn.setVisible(!isInEditMo
de);
        cancelBtn.setEnabled(isInEditMode);cancelBtn.setVisible(isInEdit
Mode);
        saveChangesBtn.setEnabled(isInEditMode);saveChangesBtn.setVisibl
e(isInEditMode);
        revertToDefaultBtn.setEnabled(isInEditMode);revertToDefaultBtn.s
etVisible(isInEditMode);
        portNumberTxt.setEditable(isInEditMode);serverAddressTxt.setEdit
able(isInEditMode);
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        portNumberTxt = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        serverAddressTxt = new javax.swing.JTextField();
        backBtn = new javax.swing.JButton();
        cancelBtn = new javax.swing.JButton();
        saveChangesBtn = new javax.swing.JButton();
```

```
        editBtn = new javax.swing.JButton();
        revertToDefaultBtn = new javax.swing.JButton();
        moreInfoBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Change Online Settings");

        jLabel1.setText("Port Number");

        portNumberTxt.setEditable(false);

        jLabel2.setText("Server Address");

        serverAddressTxt.setEditable(false);

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        cancelBtn.setText("Cancel Changes");
        cancelBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                cancelBtnActionPerformed(evt);
            }
        });

        saveChangesBtn.setText("Save Changes");
        saveChangesBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                saveChangesBtnActionPerformed(evt);
            }
        });

        editBtn.setText("Edit");
        editBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                editBtnActionPerformed(evt);
            }
        });

        revertToDefaultBtn.setText("Revert to Default Values");
        revertToDefaultBtn.addActionListener(new
java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                revertToDefaultBtnActionPerformed(evt);
            }
        });

        moreInfoBtn.setText("Further Information");
        moreInfoBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                moreInfoBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel1)
                            .addComponent(jLabel2)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(14, 14, 14)
                                .addComponent(backBtn)))
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(46, 46, 46)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(portNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 224,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(serverAddressTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 250,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(87, 87, 87)
                                .addComponent(editBtn)
                                .addGap(56, 56, 56)
                                .addComponent(moreInfoBtn))))
```

```
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(14, 14, 14)
                        .addComponent(cancelBtn)
                        .addGap(29, 29, 29)
                        .addComponent(saveChangesBtn)
                        .addGap(18, 18, 18)
                        .addComponent(revertToDefaultBtn)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(38, 38, 38)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(portNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(45, 45, 45)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel2)
                    .addComponent(serverAddressTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 86, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(cancelBtn)
                    .addComponent(saveChangesBtn)
                    .addComponent(revertToDefaultBtn))
                .addGap(23, 23, 23)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(backBtn)
                    .addComponent(editBtn)
                    .addComponent(moreInfoBtn))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>
    /**
     * changes the editing mode of the form and updates the form
accordingly
     * @param evt The edit button being pressed
     */
    private void editBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //updates editing mode and form accordingly
        bufferPortNumber=portNumberTxt.getText();bufferServerAddress=ser
verAddressTxt.getText();
        isInEditMode=!isInEditMode;
        updateFormBasedOnWhetherInEditingMode();
    }
    /**
     * Sets the port number and server address text fields to their
default values
     * @param evt The revert button being pressed
     */
    private void
revertToDefaultBtnActionPerformed(java.awt.event.ActionEvent evt)
{
       //reverts the server to the hard coded default values
        portNumberTxt.setText(DEFAULT_PORT_NUMBER);
        serverAddressTxt.setText(DEFUALT_SERVER_ADDRESS);
    }
    /**
     * Displays further information about what port number and server
address mean and how a user may go about fixing issues involving these.
     * @param evt The more information button being pressed
     */
    private void moreInfoBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //displays some info on what the port number and server address
actually mean.
        JOptionPane.showMessageDialog(this, "The port number and server
address are what allow users to access online features. Please speak to
the server owner to find the correct details. Normally, this will be the
chess club leader. ","Further Information",JOptionPane.OK_OPTION);
    }
/**
 * Saves the inputted changes after appropriate validation.
 * @param evt The save button being pressed
```

```
 */
    private void
saveChangesBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        try{
            int port = Integer.parseInt(portNumberTxt.getText());
            if(port<=0){//range check
                JOptionPane.showMessageDialog(this, "The port number
must be a positive integer greater than
zero.","Error",JOptionPane.OK_OPTION);
                return;
            }
        }catch(NumberFormatException ex){//type check
            JOptionPane.showMessageDialog(this, "The port number must be
a positive integer greater than zero.","Error",JOptionPane.OK_OPTION);
            return;
        }
        try{
            FileWriter writePort = new
FileWriter(PORT_NUMBER_FILE_NAME,false);//writes port number and server
address to their files
            BufferedWriter buffWritePort = new
BufferedWriter(writePort);
            buffWritePort.write(portNumberTxt.getText());
            buffWritePort.flush();writePort.flush();
            buffWritePort.close();writePort.close();
            FileWriter writeServ = new
FileWriter(SERVER_ADDRESS_FILE_NAME,false);
            BufferedWriter buffWriteServ = new
BufferedWriter(writeServ);
            buffWriteServ.write(serverAddressTxt.getText());
            buffWriteServ.flush();writeServ.flush();
            buffWriteServ.close();writeServ.close();
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An unexpected error
occured with either "+SERVER_ADDRESS_FILE_NAME+" or
"+PORT_NUMBER_FILE_NAME+". Error Details
"+e,"Error",JOptionPane.OK_OPTION);
        }
        isInEditMode=!isInEditMode;updateFormBasedOnWhetherInEditingMode
();//changes editing mode and updates form accordingly
    }
    /**
     * Sets the text fields to the values they were set at before
editing mode was activated
     * @param evt The cancel button being pressed
     */
    private void cancelBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //exits editing mode and reverts the values on the form to what
they were before editing began
        isInEditMode=!isInEditMode;updateFormBasedOnWhetherInEditingMode
();
```

```
        portNumberTxt.setText(bufferPortNumber);serverAddressTxt.setText
(bufferServerAddress);
    }

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user back to the log in page, reminding them that
they may wish to save any changes made first.
        if(isInEditMode){
            int confirm= JOptionPane.showConfirmDialog(this, "Are you
sure you would like to go back. Your unsaved changes will be lost if you
go back now. Are you sure you would like to
continue?","Continue",JOptionPane.OK_OPTION);
            if(confirm==JOptionPane.YES_OPTION){
                LogIn log = new LogIn();
                log.setVisible(true);
                this.dispose();
            }else{
                return;
            }
        }else{
            LogIn log = new LogIn();
            log.setVisible(true);
            this.dispose();
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ChangeInternetSettings.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ChangeInternetSettings.cl
```

```
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ChangeInternetSettings.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ChangeInternetSettings.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ChangeInternetSettings().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JButton cancelBtn;
    private javax.swing.JButton editBtn;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton moreInfoBtn;
    private javax.swing.JTextField portNumberTxt;
    private javax.swing.JButton revertToDefaultBtn;
    private javax.swing.JButton saveChangesBtn;
    private javax.swing.JTextField serverAddressTxt;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.Color;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
```

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Random;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.Timer;
import philipm.cs5.software.development.server.system.UserInfo;

/**
 * Creates a form that displays and handles the offline chess game.
 * @author mortimer
 */
public class ChessBoardOffline extends javax.swing.JFrame {
    private boolean soundIsOn=true;//stores whether sound is on
    SessionInfo session;//stores the current session information
    private boolean isUserWhite;//stores whether the logged in user is
playing as white
    private String opponentId;//stores the name of the opponent
    private ChessGame game;//this object stores the game state and
handle the game logic
    public int currentIndex;//stores the current board index
    private int computerLevel;//stores the computer level, this is -1 if
its against a person
    private JLabel boardVisual[][]=new JLabel[8][8];
    private final static String [] IMAGES_NAME = {"Black King","Black
Queen","Black Rook","Black Bishop","Black Knight","Black Pawn","White
King","White Queen","White Rook","White Bishop","White Knight","White
Pawn"};//stores the file names of the images of the pieces
    private boolean isDefaultLayout=true;//stores which GUI layout the
user is using
    private static ImageIcon imagePiece[] = new ImageIcon[12];//stores
images of the the pieces
    private Timer timer;//timer that updates the clock
    private boolean isWhiteAtBottom;//stores whether white should be
displayed at bottom of board
    private int secondsWhiteHas;
    private int secondsBlackHas;
    private String gameId;//stores the game title
    private int currentTime;//stores total time spen
    private Thread compMove=null;//thread that gets computer move
    private int yCoordSquareClicked=-1;private int xCoordSquareClicked=-
1;//stores the coordinates of the square previously clicked. If no such
square exsists, they are equal to -1 and -1
    private ArrayList<Integer>legalSquaresToMoveTo = new
ArrayList<>();//stores legal squares to move to from current square
    private boolean accountForPlayerTime;//stores whether the time each
player has spent needs to be tracked
    private ImageIcon muteIcon;private ImageIcon soundOnIcon;//stores
icons for sound setting
    private static final String MUTE_PICTURE="mute.png";//stores file
```

```
name for mute icon
    private static final String SOUND_ON_PICTURE="soundOn.png";//stores
file name for sound on image
    private static final String
BOARD_SETTING="boardSetting.txt";//stores file name for board settings
    /**
     * Creates new form ChessBoardOffline
     */
    private ChessBoardOffline() {
        initComponents();
        //game id,opponentID, isUserWhite, boardState(s),time game has
taken, time white has left, time black has left
    }
    /**
     * Updates the variable isDefault layout based on text file contents
     */
    private void updateLayoutVariableBasedOnTextFile(){
        //gets the board configuration to use
        try{
            FileReader read = new FileReader(BOARD_SETTING);
            BufferedReader buffRead = new BufferedReader(read);
            isDefaultLayout=buffRead.readLine().equals("true");
            buffRead.close();read.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }
    /**
     * Creates a new chess board.
     * @param session The session information
     * @param opponentId The type of opponent. This is used to display
their name and to determine whether the computer must generate moves
     * @param chess The chess game to represent
     * @param isUserWhite Stores whether the user is white. This is used
to determine when computer moves are needed. If this variable is true,
the board will be displayed so that white is on the bottom.
     * @param currentTimeGame The number of seconds the game has been
played for
     * @param secondsWhiteHasLeft The number of seconds white has left.
If this is unlimited, set this to -100
     * @param secondsBlackHasLeft The number of seconds black has left.
If this is unlimited, set this to -100
     * @param gameId The game id
     */
    public ChessBoardOffline(SessionInfo session, String opponentId,
ChessGame chess, boolean isUserWhite,int currentTimeGame,int
secondsWhiteHasLeft, int secondsBlackHasLeft, String gameId){
        updateLayoutVariableBasedOnTextFile();//sets variable that
controls the look of the chess board
        //sets certain variable from the parameter to object variables
        this.session =
session;this.opponentId=opponentId;this.game=chess;this.isUserWhite=isUs
erWhite;
```

```
        this.secondsWhiteHas=secondsWhiteHasLeft;this.secondsBlackHas=se
condsBlackHasLeft;
        this.gameId=gameId;
        computerLevel=-1;//-1 indicates that the opponent is a person
        //calculates the computer level
        if(opponentId.equals(OfflinePlay.LEVEL_ONE_COMP)){
            computerLevel=1;
        }else if(opponentId.equals(OfflinePlay.LEVEL_TWO_COMP)){
            computerLevel=2;
        }else if(opponentId.equals(OfflinePlay.LEVEL_THREE_COMP)){
            computerLevel=3;
        }else if(opponentId.equals(OfflinePlay.LEVEL_FOUR_COMP)){
            computerLevel=4;
        }else if(opponentId.equals(OfflinePlay.LEVEL_FIVE_COMP)){
            computerLevel=5;
        }
        currentTime=currentTimeGame;//stores the current time
        initComponents();
        //decides whether or not certain time components should be
displayed depedning on whether time control is being used
        if(secondsBlackHas==-100){
            blackTimeLbl.setVisible(false);blackTimeLeftLbl.setVisible(f
alse);
            whiteTiimeLbl.setVisible(false);whiteTimeLeftLbl.setVisible(
false);
        }
        gameTitleLbl.setText("Game Title: "+gameId);//sets the game
title
        currentIndex=chess.getLengthOfStatesFoFar()-1;
        isWhiteAtBottom=isUserWhite;
        initialiseBoard();
        if(secondsBlackHasLeft==-100){//stores whether time is limited
or not
            accountForPlayerTime=false;
        }else{
            accountForPlayerTime=true;
        }
        setTimers();
        timer = new Timer(1000, new ActionListener() {//timer that
updates the timer every second
            @Override
            public void actionPerformed(ActionEvent e) {
                //this timer updates the overall time and also the time
each individual player has left if relevant
                currentTime++;
                if(accountForPlayerTime){
                    if(game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT
){
                        secondsBlackHas--;
                        if(secondsBlackHas<=0){
                            //deletes game and returns to menu as player
has run out of time
                            secondsBlackHas=0;
```

```
                            timer.stop();
                            timer =null;
                            setTimers();
                            JOptionPane.showMessageDialog(null,"Black
has run out of time. White wins!","White Wins",JOptionPane.OK_OPTION);
                            deleteGameAndReturnToMenu();
                        }
                    }else{
                        secondsWhiteHas--;
                        if(secondsWhiteHas<=0){
                            //deletes game and returns to menu as player
has run out of time
                            secondsWhiteHas=0;
                            timer.stop();
                            timer=null;
                            JOptionPane.showMessageDialog(null,"White
has run out of time. Black wins!","Black Wins",JOptionPane.OK_OPTION);
                            deleteGameAndReturnToMenu();
                        }
                    }
                }
                setTimers();//updates timers
            }
        });timer.start();
        if(this.isUserWhite){//sets label illustrating who is who
            whiteLbl.setText("White: You");
            blackLbl.setText("Black: "+this.opponentId);
        }else{
            whiteLbl.setText("White: "+this.opponentId);
            blackLbl.setText("Black: You");
        }
        updateImageArrayBasedOnVaraible();//updates the look of board
based on which style the user is currently using
        updateBoard();
        updateSoundIcon();
        addSquareListeners();
        this.jPanel1.setDoubleBuffered(true);
        getComputerToMoveIfAppropriate();//gets the computer move if
this is a game against a computer
    }
    /**
     * Changes piece image array based on what visual style has been
chosen.
     */
    private void updateImageArrayBasedOnVaraible(){
        try{
            File fileOfPiece=null;//stores file where image is stored
            BufferedImage bufferedImagePiece;//stores a buffered image
of the piece
            String fileExt="";
             for(int i = 0; i<IMAGES_NAME.length;i++){
                if(isDefaultLayout){
                    fileExt="oldPieceImages";
```

```
                }else{
                    fileExt="test";
                }
                fileOfPiece = new
File("Assets"+File.separator+fileExt+File.separator+IMAGES_NAME[i]
+".png");//finds file
                bufferedImagePiece = ImageIO.read(fileOfPiece);
                CustomImageIconBoard cust = new
CustomImageIconBoard(bufferedImagePiece);
                imagePiece[i]= cust;//stores image
            }
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An unexpected error
occured when loading the image of pieces: "+e, "Error",
JOptionPane.OK_OPTION);
        }
    }
    /**
     * Updates the sound icon based on whether sound is currently on or
off
     */
    private void updateSoundIcon(){
        if(soundIsOn){
            speakerLbl.setIcon(soundOnIcon);
        }else{
            speakerLbl.setIcon(muteIcon);
        }
    }
    /**
     * Updates various components of board to reflect currently selected
board state
     */
    public void updateBoard(){
        ChessBoardState state = game.getState(currentIndex);//gets state
specified
        viewPreviousBoard.setEnabled(true);nextBoardBtn.setEnabled(true)
;viewPresentBoard.setEnabled(true);
        moveNumberLbl.setText("Move Number:
"+String.valueOf(state.FULL_MOVE_NUMBER));
        if(currentIndex==0){//sets it so that no previous states can be
accessed
            viewPreviousBoard.setEnabled(false);
        }
        if(currentIndex==game.getLengthOfStatesFoFar()-1){
            nextBoardBtn.setEnabled(false);//sets it so that view next
board is not enabled
            viewPresentBoard.setEnabled(false);
        }
        //adds visual mark to display who's turn it is to play
        if(state.IS_BLACK_TURN_TO_PLAY_NEXT){
            isBlackTurnLbl.setText("*");isWhiteTurnLbl.setText("");
        }else{
            isBlackTurnLbl.setText("");isWhiteTurnLbl.setText("*");
```

```
        }
        if(game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
            blackTimeLbl.setText("*");whiteTiimeLbl.setText("");
        }else{
            blackTimeLbl.setText("");whiteTiimeLbl.setText("*");
        }
        int[][]board=state.getCopyOfBoard();//gets a copy of the game
board
        if(isWhiteAtBottom==false){//checks to see whether board needs
to be flipped
            //rotates board by 180 degrees
            rotateMatrix(board);
        }
        //disables resign button if game is played against computer and
its the computers turn
        resignBtn.setEnabled(true);
        if(computerLevel!=-1){
            if((isUserWhite&&game.getCurrentState().IS_BLACK_TURN_TO_PLA
Y_NEXT)||(!isUserWhite&&!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NE
XT)){
                resignBtn.setEnabled(false);
            }
        }
        yCoordSquareClicked=-1;xCoordSquareClicked=-1;
        legalSquaresToMoveTo=new ArrayList<>();
        //loads the board and the corresponding images
        int yKing=-1;int xKing=-1;
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                //sets square to the appropriate colour
                if(y%2==x%2){
                    if(isDefaultLayout){
                       boardVisual[y][x].setBackground(Color.white);
                    }else{
                        boardVisual[y][x].setBackground(Color.decode("#F
1D9C0"));
                    }
                }else{
                    if(isDefaultLayout){
                        boardVisual[y][x].setBackground(Color.black);
                    }else{
                        boardVisual[y][x].setBackground(Color.decode("#A
97A65"));
                    }
                }
                //sets the icons of the chess board depending on what
piece is currently on the square
                switch (board[y][x]) {
                    case ChessBoardState.EMPTY_VALUE:
                        boardVisual[y][x].setIcon(null);
                        break;
                    case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
```

```
                        boardVisual[y][x].setIcon(imagePiece[11]);
                        break;
                    case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[5]);
                        break;
                    case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[8]);
                        break;
                    case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[2]);
                        break;
                    case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[10]);
                        break;
                    case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[4]);
                        break;
                    case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[9]);
                        break;
                    case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[3]);
                        break;
                    case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[7]);
                        break;
                    case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[1]);
                        break;
                    case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[6]);
                        if(!state.IS_BLACK_TURN_TO_PLAY_NEXT){
                            yKing=y;xKing=x;
                        }
                        break;
                    default:
                        //black king
                        boardVisual[y][x].setIcon(imagePiece[0]);
                        if(state.IS_BLACK_TURN_TO_PLAY_NEXT){
                            yKing=y;xKing=x;
                        }
                        break;
                }
```

```
            }
        }
        //if the king is in check, the square of the king is highlighted
red
        if(ChessGame.isInCheck(!state.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(state.BOARD, state.CASTLING_RIGHTS,
!state.IS_BLACK_TURN_TO_PLAY_NEXT, state.FULL_MOVE_NUMBER,
state.HALF_MOVE_CLOCK,
state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT))){//if king is in check,
its square is coloured red
            boardVisual[yKing][xKing].setBackground(Color.red);
        }
    }
    /**
     * Checks to see if a computer move is needed. If it is, the
computer makes a move and updates the board
     */
    private void getComputerToMoveIfAppropriate(){
        if(computerLevel!=-1 &&
((isUserWhite&&game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT)||(!isU
serWhite&&!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT))){
            //creates a new thread and updates the board once move has
been calculated
            GetComputerMove com = new GetComputerMove(game, this,
computerLevel);//creates a thread that processes the move
            compMove = new Thread(com);compMove.start();
        }
    }
    /**
     * Sets the values for all timers  that are appropriate
     */
    private void setTimers(){
        //sets value of overall timer
        int sec=currentTime%60;
        int hour=currentTime/60;
        int min=hour%60;
        hour=hour/60;
        String s=String.valueOf(sec);if(s.length()==1){s="0"+s;}//makes
seconds at least two digits
        String m = String.valueOf(min);if(m.length()==1){m="0"+m;}
        String h=String.valueOf(hour);if(h.length()==1){h="0"+h;}
        timerLbl.setText(h+":"+m+":"+s);//sets time in hours, minutes
and seconds format
        if(accountForPlayerTime){//sets time left for each colour if
relevant
                m=String.valueOf(secondsBlackHas/60);
                if(m.length()==1){
                    m="0"+m;
                }
                s=String.valueOf(secondsBlackHas%60);
                if(s.length()==1){
                    s="0"+s;
                }
```

```
                blackTimeLeftLbl.setText("Black Time: "+m+":"+s);
                m=String.valueOf(secondsWhiteHas/60);
                if(m.length()==1){
                    m="0"+m;
                }
                s=String.valueOf(secondsWhiteHas%60);
                if(s.length()==1){
                    s="0"+s;
                }
                whiteTimeLeftLbl.setText("White Time:
"+m+":"+s);
        }
    }
    /**
     * This rotates a chess board array by 180 degrees.
     * @param board The board (must be an 8*8 array for this rotation
method to work)
     */
    private static void rotateMatrix(int[][] board) {
        int temp;//used as a buffer
        //rotates 8*8 board
        for (int i = 0; i <= 3; i++) {
            for (int j = 0; j < 8; j++) {
                temp = board[i][j];
                board[i][j] = board[7 - i ][7- j];
                board[7-i][7- j] = temp;
            }
        }
    }
    /**
     * This methods creates the board pieces, adds appropriate action
listeners and is used to initialise the board. It also changes colour of
squares
     * depending on whether the user is white.
     */
    private void initialiseBoard(){
        //creates JLabels and loads them to array
        final int WIDTH_OF_SQUARE=56;//stores the width of the square 56
        final int HEIGHT_OF_SQUARE=56;//stores the height of the square
        final int SPACE_ABOVE_BOARD=10;//stores the space above the
board
        final int SPACE_TO_RIGHT_OF_BOARD=50;//stores the gap between
the right edge of the screen and the board
        int xCoordOfSquare=SPACE_TO_RIGHT_OF_BOARD;//stores the
coordinate of the square
        int yCoordOfSquare=SPACE_ABOVE_BOARD;//stores the y coordinate
of the square
        final String
RANK_AND_FILE[]={"a","b","c","d","e","f","g","h","1","2","3","4","5","6"
,"7","8"};//stores the values to be used by labels that indicate the
coordinates on the board
        final int HEIGHT_OF_LABEL=16;//stores the height of the label
showing the rank or file of the square
```

```
        final int DISTANCE_OF_LETTERS_BELOW=3;//stores distance between
squares and letters
        //adds colour squares to GUI
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                boardVisual[y][x]=new BoardJLabel();//creates the
BoardJLabel that will display a square of the chess board
                boardVisual[y][x].setSize(WIDTH_OF_SQUARE,
HEIGHT_OF_SQUARE);//sets size of label
                boardVisual[y][x].setLocation(xCoordOfSquare,
yCoordOfSquare);
                //sets square to the approrpiate colour
                if(y%2==x%2){
                    //boardVisual[y][x].setBackground(Color.white);
                    boardVisual[y][x].setBackground(Color.decode("#F1D9C
0"));
                }else{
                    //boardVisual[y][x].setBackground(Color.black);
                    boardVisual[y][x].setBackground(Color.decode("#A97A6
5"));
                }
                boardVisual[y][x].setOpaque(true);//allows piece's
background to be seen
                boardVisual[y][x].setDoubleBuffered(true);
                this.jPanel1.add(boardVisual[y][x]);
                if(x==7){//updates coordinates of next label
                    xCoordOfSquare=SPACE_TO_RIGHT_OF_BOARD;
                    yCoordOfSquare=yCoordOfSquare+HEIGHT_OF_SQUARE;
                }else{
                    xCoordOfSquare=xCoordOfSquare+WIDTH_OF_SQUARE;
                }
            }
        }
        //adds labels to indicate rank and file of pieces
        xCoordOfSquare=SPACE_TO_RIGHT_OF_BOARD+(WIDTH_OF_SQUARE/2)-
(HEIGHT_OF_LABEL/2);
        yCoordOfSquare=yCoordOfSquare+DISTANCE_OF_LETTERS_BELOW;
        JLabel lbl;//stores label to be added
        for(int indexForCoord=0;indexForCoord<8;indexForCoord++){//adds
letters to board which indicate file
            lbl=new JLabel(RANK_AND_FILE[indexForCoord]);
            lbl.setSize(HEIGHT_OF_LABEL, HEIGHT_OF_LABEL);
            lbl.setLocation(xCoordOfSquare,yCoordOfSquare);
            xCoordOfSquare=xCoordOfSquare+WIDTH_OF_SQUARE;
            this.jPanel1.add(lbl);
        }
        yCoordOfSquare=SPACE_ABOVE_BOARD+(HEIGHT_OF_SQUARE/2)-
(HEIGHT_OF_LABEL/2);
        xCoordOfSquare=SPACE_TO_RIGHT_OF_BOARD-
DISTANCE_OF_LETTERS_BELOW-HEIGHT_OF_LABEL;
        for(int indexForCoord=15;indexForCoord>=8;indexForCoord--){
            lbl=new JLabel(RANK_AND_FILE[indexForCoord]);
            lbl.setSize(HEIGHT_OF_LABEL, HEIGHT_OF_LABEL);
```

```
            lbl.setLocation(xCoordOfSquare,yCoordOfSquare);
            yCoordOfSquare=yCoordOfSquare+HEIGHT_OF_SQUARE;
            this.jPanel1.add(lbl);
        }
        //loads all images and stores them (i.e. of the pieces and sound
icons)
        try{
            File fileOfPiece;//stores file where image is stored
            BufferedImage bufferedImagePiece=null;//stores a buffered
image of the piece
            Image imgPiece=null;//stores the actual image of the piece
             for(int i = 0; i<IMAGES_NAME.length;i++){
                fileOfPiece = new
File("Assets"+File.separator+"oldPieceImages"+File.separator+IMAGES_NAME
[i] +".png");//finds file
                bufferedImagePiece = ImageIO.read(fileOfPiece);
                //imgPiece =
bufferedImagePiece.getScaledInstance(WIDTH_OF_SQUARE, HEIGHT_OF_SQUARE ,
Image.SCALE_SMOOTH);//loads image of piece and scales it to size
                //imageIconPiece = new ImageIcon(bufferedImagePiece);
                CustomImageIconBoard cust = new
CustomImageIconBoard(bufferedImagePiece);
                imagePiece[i]= cust;//stores image
            }
             //gets speaker icons
             fileOfPiece=new File("Assets"+File.separator
+MUTE_PICTURE);
             bufferedImagePiece=ImageIO.read(fileOfPiece);
             imgPiece=bufferedImagePiece.getScaledInstance(speakerLbl.ge
tPreferredSize().width, speakerLbl.getPreferredSize().width,
Image.SCALE_SMOOTH);
             muteIcon=new CustomImageIconBoardSpeaker(imgPiece);
             fileOfPiece=new File("Assets"+File.separator
+SOUND_ON_PICTURE);
             bufferedImagePiece=ImageIO.read(fileOfPiece);
             imgPiece=bufferedImagePiece.getScaledInstance(speakerLbl.ge
tPreferredSize().width, speakerLbl.getPreferredSize().width,
Image.SCALE_SMOOTH);
             soundOnIcon=new CustomImageIconBoardSpeaker(imgPiece);
            //closes and flushes relevant methods
            imgPiece.flush();bufferedImagePiece.flush();
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An unexpected error
occured when loading the image of pieces: "+e, "Error",
JOptionPane.OK_OPTION);
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
```

```
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        backBtn = new javax.swing.JButton();
        resignBtn = new javax.swing.JButton();
        viewPreviousBoard = new javax.swing.JButton();
        viewPresentBoard = new javax.swing.JButton();
        nextBoardBtn = new javax.swing.JButton();
        timerLbl = new javax.swing.JLabel();
        timerLbl1 = new javax.swing.JLabel();
        blackLbl = new javax.swing.JLabel();
        whiteLbl = new javax.swing.JLabel();
        isBlackTurnLbl = new javax.swing.JLabel();
        isWhiteTurnLbl = new javax.swing.JLabel();
        moveNumberLbl = new javax.swing.JLabel();
        flipBoardBtn = new javax.swing.JButton();
        blackTimeLeftLbl = new javax.swing.JLabel();
        whiteTimeLeftLbl = new javax.swing.JLabel();
        gameTitleLbl = new javax.swing.JLabel();
        blackTimeLbl = new javax.swing.JLabel();
        whiteTiimeLbl = new javax.swing.JLabel();
        speakerLbl = new BoardJLabel()
        ;
        changeBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Offline Game");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosed(java.awt.event.WindowEvent evt) {
                formWindowClosed(evt);
            }
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        backBtn.setText("Back");
        backBtn.setDoubleBuffered(true);
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        resignBtn.setText("Resign");
        resignBtn.setDoubleBuffered(true);
        resignBtn.addActionListener(new java.awt.event.ActionListener()
```

```
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                resignBtnActionPerformed(evt);
            }
        });

        viewPreviousBoard.setText("Previous Board");
        viewPreviousBoard.setDoubleBuffered(true);
        viewPreviousBoard.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewPreviousBoardActionPerformed(evt);
            }
        });

        viewPresentBoard.setText("Current Board");
        viewPresentBoard.setDoubleBuffered(true);
        viewPresentBoard.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewPresentBoardActionPerformed(evt);
            }
        });

        nextBoardBtn.setText("Next Board");
        nextBoardBtn.setDoubleBuffered(true);
        nextBoardBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                nextBoardBtnActionPerformed(evt);
            }
        });

        timerLbl.setText(" ");
        timerLbl.setDoubleBuffered(true);

        timerLbl1.setText("Timer:");
        timerLbl1.setDoubleBuffered(true);

        blackLbl.setBackground(new java.awt.Color(204, 204, 204));
        blackLbl.setText("Black:");
        blackLbl.setDoubleBuffered(true);
        blackLbl.setOpaque(true);

        whiteLbl.setBackground(new java.awt.Color(255, 255, 255));
        whiteLbl.setText("White:");
        whiteLbl.setDoubleBuffered(true);
        whiteLbl.setOpaque(true);
```

```
        isBlackTurnLbl.setForeground(new java.awt.Color(255, 0, 0));
        isBlackTurnLbl.setText("*");
        isBlackTurnLbl.setDoubleBuffered(true);

        isWhiteTurnLbl.setForeground(new java.awt.Color(255, 0, 0));
        isWhiteTurnLbl.setText("*");
        isWhiteTurnLbl.setDoubleBuffered(true);

        moveNumberLbl.setText("Move Number: ");
        moveNumberLbl.setDoubleBuffered(true);

        flipBoardBtn.setText("Flip Board");
        flipBoardBtn.setDoubleBuffered(true);
        flipBoardBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                flipBoardBtnActionPerformed(evt);
            }
        });

        blackTimeLeftLbl.setBackground(new java.awt.Color(204, 204,
204));
        blackTimeLeftLbl.setText("Black Time: ");
        blackTimeLeftLbl.setDoubleBuffered(true);
        blackTimeLeftLbl.setOpaque(true);

        whiteTimeLeftLbl.setBackground(new java.awt.Color(255, 255,
255));
        whiteTimeLeftLbl.setText("White Time:");
        whiteTimeLeftLbl.setDoubleBuffered(true);
        whiteTimeLeftLbl.setOpaque(true);

        gameTitleLbl.setText("Game Title: ");
        gameTitleLbl.setDoubleBuffered(true);

        blackTimeLbl.setForeground(new java.awt.Color(255, 0, 0));
        blackTimeLbl.setText("*");
        blackTimeLbl.setDoubleBuffered(true);

        whiteTiimeLbl.setForeground(new java.awt.Color(255, 0, 0));
        whiteTiimeLbl.setText("*");
        whiteTiimeLbl.setDoubleBuffered(true);

        speakerLbl.setPreferredSize(new java.awt.Dimension(15, 15));
        speakerLbl.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                speakerLblMouseClicked(evt);
            }
        });

        changeBtn.setText("Change Board");
        changeBtn.addActionListener(new java.awt.event.ActionListener()
```

```
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                changeBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap(557, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(isWhiteTurnLbl)
                            .addComponent(isBlackTurnLbl))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(blackLbl)
                            .addComponent(whiteLbl)
                            .addComponent(moveNumberLbl))
                        .addGap(107, 107, 107))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(blackTimeLbl)
                            .addComponent(whiteTiimeLbl))
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.UNRELATED)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(whiteTimeLeftLbl)
                            .addComponent(blackTimeLeftLbl))
                        .addGap(253, 253, 253))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.TRAILING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(resignBtn)
                                .addGap(60, 60, 60)
                                .addComponent(speakerLbl,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(viewPreviousBoard)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED)
                                .addComponent(viewPresentBoard)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.UNRELATED)
                                .addComponent(nextBoardBtn)))
                        .addGap(25, 25, 25))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
,
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING, false)
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAI
LING, jPanel1Layout.createSequentialGroup()
                            .addComponent(changeBtn)
                            .addPreferredGap(javax.swing.LayoutStyle.Com
ponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(backBtn)
                            .addContainerGap())
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAI
LING, jPanel1Layout.createSequentialGroup()
                            .addComponent(flipBoardBtn)
                            .addGap(321, 321, 321)))))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(gameTitleLbl)
                .addGap(184, 184, 184)
                .addComponent(timerLbl1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED)
                .addComponent(timerLbl)
                .addGap(49, 49, 49))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(timerLbl)
                    .addComponent(timerLbl1)
                    .addComponent(gameTitleLbl))
                .addGap(58, 58, 58)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
```

```
                    .addComponent(blackLbl)
                    .addComponent(isBlackTurnLbl))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.UNRELATED)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(whiteLbl)
                    .addComponent(isWhiteTurnLbl))
                .addGap(36, 36, 36)
                .addComponent(moveNumberLbl)
                .addGap(52, 52, 52)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(blackTimeLeftLbl)
                    .addComponent(blackTimeLbl))
                .addGap(18, 18, 18)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(whiteTimeLeftLbl)
                    .addComponent(whiteTiimeLbl))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 22, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(viewPreviousBoard)
                            .addComponent(viewPresentBoard)
                            .addComponent(nextBoardBtn))
                        .addGap(31, 31, 31)
                        .addComponent(resignBtn))
                    .addComponent(speakerLbl,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 52,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(16, 16, 16)
                .addComponent(flipBoardBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(backBtn)
                    .addComponent(changeBtn))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
```

```
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void flipBoardBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //flips the board
        isWhiteAtBottom=!isWhiteAtBottom;
        updateBoard();
    }

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //saves the game and takes the user back to the menu
        saveBoardToFile();
        OfflinePlay offline = new OfflinePlay(session);
        offline.setVisible(true);
        this.dispose();
    }

    private void
viewPreviousBoardActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes user to previous board state
        currentIndex--;
        updateBoard();
    }

    private void nextBoardBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // takes user to a more recent board state
        currentIndex++;
        updateBoard();
    }

    private void
viewPresentBoardActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes user to current board
        currentIndex=game.getLengthOfStatesFoFar()-1;
        updateBoard();
    }
```

```
    private void resignBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // gets player to resign
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to resign? Doing so means you will lose the
game.","Resign?",JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            if(game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
                JOptionPane.showMessageDialog(this, "Black resigned -
White wins!","White Wins!",JOptionPane.OK_OPTION);
            }else{
                JOptionPane.showMessageDialog(this, "White resigned -
Black wins!","Black Wins!",JOptionPane.OK_OPTION);
            }
            deleteGameAndReturnToMenu();
        }
    }

    private void formWindowClosed(java.awt.event.WindowEvent evt)
{
        if(compMove!=null){//kills thread that gets computer move if
appropriate when closing the window
            compMove.interrupt();
        }
        compMove=null;
    }

    private void formWindowClosing(java.awt.event.WindowEvent evt)
{
      //if a user closes the window by pressing the cross in the top
right corner, this method is invoked. It saves the board state
      saveBoardToFile();
      if(compMove!=null){
            compMove.interrupt();
        }
        compMove=null;
    }

    private void speakerLblMouseClicked(java.awt.event.MouseEvent evt)
{
    //updates whether the sound is on or not. It also changes the
relevant sound icon to show this.
    soundIsOn=!soundIsOn;
    updateSoundIcon();
    }

    private void changeBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // updates the variable is Default layout and stores it to a
text file. This basically changes how the pieces and squares look
        isDefaultLayout=!isDefaultLayout;
        try{
            FileWriter write = new
```

```
FileWriter(BOARD_SETTING,false);//saves the user's prefered board style
to a text file
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write(String.valueOf(isDefaultLayout));
            buffWrite.flush();write.flush();
            buffWrite.close();write.close();
        }catch(IOException e){
            System.out.println(e);
        }
        updateImageArrayBasedOnVaraible();
        updateBoard();//updates board
    }
    /**
     * Saves the current board state to the offline game text file.
     */
    private void saveBoardToFile(){
        String fileContents="";//stores the contents of the offline game
text file
        String
updatedRecord=gameId+UserInfo.SPLIT_ITEM+opponentId+UserInfo.SPLIT_ITEM+
String.valueOf(isUserWhite)+UserInfo.SPLIT_ITEM+game.getState(0).getFenS
tring();//stores updated record
        for(int i=1;i<game.getLengthOfStatesFoFar();i++){//adds all
board states to file
            updatedRecord=updatedRecord+OfflinePlay.SPLIT_FEN_STRING+gam
e.getState(i).getFenString();
        }
        updatedRecord=updatedRecord+UserInfo.SPLIT_ITEM+String.valueOf(c
urrentTime)+UserInfo.SPLIT_ITEM+String.valueOf(secondsWhiteHas)+UserInfo
.SPLIT_ITEM+String.valueOf(secondsBlackHas);
        try{
            FileReader read =new
FileReader(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+session.getUs
ername()+".txt");
            BufferedReader buffRead = new BufferedReader(read);String
lineRead;
            while((lineRead=buffRead.readLine())!=null){
                if(lineRead.split(UserInfo.SPLIT_ITEM)[0].equals(gameId)
){//if record is the same as the current game
                    lineRead=updatedRecord;
                }
                if(fileContents.equals("")){//writes updated content to
file
                    fileContents=lineRead;
                }else{
                    fileContents=fileContents+System.lineSeparator()+lin
eRead;
                }
            }buffRead.close();read.close();
            //writes the new text file
            FileWriter write = new
FileWriter(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+session.getUs
ername()+".txt",false);
```

```
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write(fileContents);
            buffWrite.flush();write.flush();
            buffWrite.close();write.close();
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An unexpected error
occured: "+e,"Error",JOptionPane.OK_OPTION);
        }
    }
    /**
     * Checks to see if current board is in a terminal state. If it is,
the game result is announced and then the system terminates
     */
    public void checkForTerminalState(){
        if(game.getAllLegalBoardsFromState(game.getCurrentState()).size(
)==0){//checks to see if there are any legal moves from the current
board state - if there are not then the game must be over
            timer.stop();//stops the timers
            if(game.getCurrentState().getIsDraw()){
                //displays that the game was a draw
                JOptionPane.showMessageDialog(this, "The game was a
draw!","Draw!",JOptionPane.OK_OPTION);
                deleteGameAndReturnToMenu();
                return;
            }
            if(ChessGame.isInCheck(!game.getCurrentState().IS_BLACK_TURN
_TO_PLAY_NEXT, new ChessBoardState(game.getCurrentState().BOARD,
game.getCurrentState().CASTLING_RIGHTS,
!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT,
game.getCurrentState().FULL_MOVE_NUMBER,
game.getCurrentState().HALF_MOVE_CLOCK,
game.getCurrentState().SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT))){
                //if the user is in check, then the outcome of the game
must be checkmate
                if(game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
                    JOptionPane.showMessageDialog(this, "White
wins!","Checkmate!",JOptionPane.OK_OPTION);
                    deleteGameAndReturnToMenu();
                    return;
                }else{
                    JOptionPane.showMessageDialog(this, "Black
wins.","Checkmate!",JOptionPane.OK_OPTION);
                    deleteGameAndReturnToMenu();
                    return;
                }
            }
            JOptionPane.showMessageDialog(this, "The game was a
draw!","Draw!",JOptionPane.OK_OPTION);
            deleteGameAndReturnToMenu();
            return;
        }
    }
    /**
```

```
     * This method deletes the game and returns to the offline play
menu.
     */
    private void deleteGameAndReturnToMenu(){
        String fileContents="";//stores contents of new file
        String lineRead;
        try{
            //gets new file contents - which is the same as the old file
contents just without the current game record
            FileReader read = new
FileReader(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+session.getUs
ername()+".txt");
            BufferedReader buffRead = new BufferedReader(read);
            while((lineRead=buffRead.readLine())!=null){
                //loops through the file and appends the line to the
file if it is not about the current game
                if(!lineRead.split(UserInfo.SPLIT_ITEM)[0].equals(gameId
)){
                    if(fileContents.equals("")){
                        fileContents=lineRead;
                    }else{
                        fileContents=fileContents+System.lineSeparator()
+lineRead;
                    }
                }
            }buffRead.close();read.close();
            FileWriter write = new
FileWriter(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+session.getUs
ername()+".txt",false);//writes the updated offline game records to the
file
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write(fileContents);
            buffWrite.flush();write.flush();
            buffWrite.close();write.close();//closes and flushes writers
        }catch(IOException e){
            JOptionPane.showMessageDialog(this,"An unexpected error
occurred. "+e,"Error",JOptionPane.OK_OPTION);
        }
        //takes user back to menu
        OfflinePlay off = new OfflinePlay(session);
        off.setVisible(true);this.dispose();return;
    }
    /**
     * Plays a sound to indicate that a move has been made if the game
is not muted
     */
    public void playMoveMadeSound(){
        if(soundIsOn){
            Random rnd = new Random();
            Audio.playSound("Assets"+File.separator+String.valueOf(rnd.n
extInt(8)+1)+"Sound.wav");
        }
    }
```

```
    /**
     * Handles when a square is clicked. This includes highlighting the
square and all available moves if appropriate.
     * @param y The y index of the square
     * @param x The x index of the square
     */
    private void squareClicked(int y,int x){
        //if opponent is a computer and its the computer's turn to play,
no squares should be highlighted
        if((computerLevel!=-
1)&&((isUserWhite&&game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT)||(
!isUserWhite&&!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT))){
            return;
        }
        if(currentIndex!=game.getLengthOfStatesFoFar()-1){//if the board
is not the current board, no action is needed
            return;
        }
        int yBoard=y;int xBoard=x;//stores the y and x coords regarding
to the actual board array stored
        if(isWhiteAtBottom==false){//accounts for a flipped board
            yBoard=7-y;xBoard=7-x;
        }
        //updateBoard();
        //checks to see if square clicked is a legal square to move to.
If no square has been clicked, the array is of length 0
        for(int move=0;move<legalSquaresToMoveTo.size();move=move+2){
            if(legalSquaresToMoveTo.get(move)==yBoard&&legalSquaresToMov
eTo.get(move+1)==xBoard){//if square choosen is a move, it updates game
object and board
                ArrayList<ChessBoardState>states=ChessGame.getAllLegalBo
ardsFromMove(yCoordSquareClicked, xCoordSquareClicked, yBoard, xBoard,
game.getCurrentState());
                if(states.size()==1){
                    game.addState(states.get(0));
                }else{//used to allow user to make choice about pawn
promotion
                    Object[] options =
{"Queen","Rook","Bishop","Knight"};//stores the possible pieces to
promote
                    int choice=JOptionPane.showOptionDialog(this,"Please
select piece to promote pawn
to.","Piece",JOptionPane.YES_NO_CANCEL_OPTION,JOptionPane.QUESTION_MESSA
GE,null,options,options[0]);
                    if(choice==-1){//if no option is seleted, the queen
is chosen
                        game.addState(states.get(0));
                    }else{
                        game.addState(states.get(choice));
                    }
                }
                //updates board following the move having been made
                currentIndex++;
```

```
                updateBoard();
                playMoveMadeSound();
                checkForTerminalState();
                getComputerToMoveIfAppropriate();
                return;
            }
        }
        //if invalid square is selected, nothing occurrs
        if((game.getCurrentState().BOARD[yBoard][xBoard]>=0&&game.getCur
rentState().IS_BLACK_TURN_TO_PLAY_NEXT)||(game.getCurrentState().BOARD[y
Board][xBoard]<=0&&!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT)){
            return;
        }
        if(yCoordSquareClicked==yBoard&&xCoordSquareClicked==xBoard){//i
f selected square is being clicked again, it will be deselected
            updateBoard();
            return;
        }
        updateBoard();
        //gets all legal moves from square pressed and highlights square
pressed as well as any legal moves
        yCoordSquareClicked=yBoard;xCoordSquareClicked=xBoard;
        boardVisual[y][x].setBackground(Color.orange);
        legalSquaresToMoveTo=ChessGame.getAllLegalMovesFromSquare(yBoard
, xBoard, game.getCurrentState());
        for(int
move=0;move<legalSquaresToMoveTo.size();move=move+2){//goes through each
legal move and highlights square
            if(isWhiteAtBottom){//accounts for flipped board
                boardVisual[legalSquaresToMoveTo.get(move)][legalSquares
ToMoveTo.get(move+1)].setBackground(Color.blue);
            }else{
                boardVisual[7-legalSquaresToMoveTo.get(move)][7-
legalSquaresToMoveTo.get(move+1)].setBackground(Color.blue);
            }
        }
    }
    /**
     * Adds listeners to each square on the board. This code is very
long and has been generated by a computer. It has been collapsed as it
takes up a large amount of space
     */
    private void addSquareListeners(){
        //<editor-fold defaultstate="collapsed" desc="Computer generated
action listeners for each square of the board">
        boardVisual[0][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,0);}});
        boardVisual[0][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,1);}});
        boardVisual[0][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
```

```
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,2);}});
        boardVisual[0][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,3);}});
        boardVisual[0][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,4);}});
        boardVisual[0][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,5);}});
        boardVisual[0][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,6);}});
        boardVisual[0][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,7);}});
        boardVisual[1][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,0);}});
        boardVisual[1][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,1);}});
        boardVisual[1][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,2);}});
        boardVisual[1][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,3);}});
        boardVisual[1][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,4);}});
        boardVisual[1][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,5);}});
        boardVisual[1][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,6);}});
        boardVisual[1][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,7);}});
        boardVisual[2][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,0);}});
        boardVisual[2][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,1);}});
        boardVisual[2][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,2);}});
        boardVisual[2][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,3);}});
        boardVisual[2][4].addMouseListener(new
```

```
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,4);}});
        boardVisual[2][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,5);}});
        boardVisual[2][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,6);}});
        boardVisual[2][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,7);}});
        boardVisual[3][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,0);}});
        boardVisual[3][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,1);}});
        boardVisual[3][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,2);}});
        boardVisual[3][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,3);}});
        boardVisual[3][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,4);}});
        boardVisual[3][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,5);}});
        boardVisual[3][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,6);}});
        boardVisual[3][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,7);}});
        boardVisual[4][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,0);}});
        boardVisual[4][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,1);}});
        boardVisual[4][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,2);}});
        boardVisual[4][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,3);}});
        boardVisual[4][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,4);}});
        boardVisual[4][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,5);}});
```

```
        boardVisual[4][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,6);}});
        boardVisual[4][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,7);}});
        boardVisual[5][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,0);}});
        boardVisual[5][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,1);}});
        boardVisual[5][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,2);}});
        boardVisual[5][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,3);}});
        boardVisual[5][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,4);}});
        boardVisual[5][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,5);}});
        boardVisual[5][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,6);}});
        boardVisual[5][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,7);}});
        boardVisual[6][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,0);}});
        boardVisual[6][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,1);}});
        boardVisual[6][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,2);}});
        boardVisual[6][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,3);}});
        boardVisual[6][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,4);}});
        boardVisual[6][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,5);}});
        boardVisual[6][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,6);}});
        boardVisual[6][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
```

```
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,7);}});
        boardVisual[7][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,0);}});
        boardVisual[7][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,1);}});
        boardVisual[7][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,2);}});
        boardVisual[7][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,3);}});
        boardVisual[7][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,4);}});
        boardVisual[7][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,5);}});
        boardVisual[7][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,6);}});
        boardVisual[7][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent
evt){squareClicked(7,7);}});
        //</editor-fold>
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ChessBoardOffline.class.g
etName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
```

```
            java.util.logging.Logger.getLogger(ChessBoardOffline.class.g
etName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ChessBoardOffline.class.g
etName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ChessBoardOffline.class.g
etName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ChessBoardOffline().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JLabel blackLbl;
    private javax.swing.JLabel blackTimeLbl;
    private javax.swing.JLabel blackTimeLeftLbl;
    private javax.swing.JButton changeBtn;
    private javax.swing.JButton flipBoardBtn;
    private javax.swing.JLabel gameTitleLbl;
    private javax.swing.JLabel isBlackTurnLbl;
    private javax.swing.JLabel isWhiteTurnLbl;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JLabel moveNumberLbl;
    private javax.swing.JButton nextBoardBtn;
    private javax.swing.JButton resignBtn;
    private javax.swing.JLabel speakerLbl;
    private javax.swing.JLabel timerLbl;
    private javax.swing.JLabel timerLbl1;
    private javax.swing.JButton viewPresentBoard;
    private javax.swing.JButton viewPreviousBoard;
    private javax.swing.JLabel whiteLbl;
    private javax.swing.JLabel whiteTiimeLbl;
    private javax.swing.JLabel whiteTimeLeftLbl;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package philipm.cs5.software.development.user.system;

import java.awt.Color;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Random;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.Timer;

/**
 * Creates new form which displays chess board for an online game of
chess.
 * @author mortimer
 */
public class ChessBoardOnline extends javax.swing.JFrame {
    private boolean soundIsOn=true;//stores whether sound is on
    SessionInfo session;//stores the current session information
    private boolean isUserWhite;//stores whether the logged in user is
playing as white
    private ChessGame game;//this object stores the game state and
handle the game logic
    public int currentIndex;//stores the current board index
    private JLabel boardVisual[][]=new JLabel[8][8];
    private final static String [] IMAGES_NAME = {"Black King","Black
Queen","Black Rook","Black Bishop","Black Knight","Black Pawn","White
King","White Queen","White Rook","White Bishop","White Knight","White
Pawn"};//stores the file names of the images of the pieces
    private static ImageIcon imagePiece[] = new ImageIcon[12];//stores
images of the the pieces
    private Timer timer;//timer that updates the clock
    private boolean isWhiteAtBottom;//stores whether white should be
displayed at bottom of board
    private int secondsWhiteHas;
    private boolean isDefaultLayout=true;
    private static final String BOARD_SETTING="boardSetting.txt";
    private int secondsBlackHas;
    private int currentTime;//stores total time spen
    private int yCoordSquareClicked=-1;private int xCoordSquareClicked=-
1;//stores the coordinates of the square previously clicked. If no such
square exsists, they are equal to -1 and -1
```

```
    private ArrayList <Integer> legalSquaresToMoveTo=new
ArrayList<>();//stores legal squares to move to from current square
    private boolean accountForPlayerTime;//stores whether the time each
player has spent needs to be tracked
    private ImageIcon muteIcon;private ImageIcon soundOnIcon;//stores
icons for sound setting
    private static final String MUTE_PICTURE="mute.png";
    private Thread getOpponentMoveThread=null;//stores the thread that
gets an opponents move
    private static final String SOUND_ON_PICTURE="soundOn.png";
    private int eloPlayerOne;//stores the elo rating of player one
    private int eloPlayerTwo;//stores the elo rating of player two
    private String[]gameData;//stores the game data
    private boolean isUserPlayerOne;//stores whether the user is player
one in the game data
    private final static Color DEFAULT_BUTTON_BACKGROUND=new
JButton().getBackground();
    private boolean killThread=false;
    /**
     * Creates new form ChessBoardOffline
     */
    private ChessBoardOnline() {
        initComponents();
    }
    /**
     * Creates a new chess board.
     * @param session The session information
     * @param chess The chess game to represent
     * @param gameData The record storing the online game
     */
    public ChessBoardOnline(SessionInfo session, ChessGame
chess,String[]gameData ){
        updateLayoutVariableBasedOnTextFile();//updates board layout
based on which layout the user prefers
        //converts various paramters to object variables
        this.gameData=gameData;
        this.session = session;this.game=chess;
        this.isUserPlayerOne=gameData[1].equals(session.getUsername());/
/determines whether the user is classed as player one
        this.secondsWhiteHas=Integer.parseInt(gameData[5]);this.secondsB
lackHas=Integer.parseInt(gameData[6]);
        currentTime=Integer.parseInt(gameData[4]);
        isUserWhite=(gameData[1].equals(session.getUsername()) &&
gameData[10].equals("true"))||
(gameData[2].equals(session.getUsername()) &&
gameData[10].equals("false"));
        initComponents();
        invisibleDrawBtn.setEnabled(false);
        invisibleDrawBtn.setVisible(false);
        //attempts to load the elo of users onto the form
        try{
            eloPlayerOne=(ThreadClient.logInWithUserName(gameData[1],
"passw").getElo());
```

```
            eloPlayerTwo=(ThreadClient.logInWithUserName(gameData[2],
"passw").getElo());
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
            //takes user back to menu
            Menu menu = new Menu(session);
            menu.setVisible(true);
            this.dispose();return;
        }
        accountForPlayerTime=true;
        if(secondsBlackHas==-100){//sets timer labels invisible if
player time is not tracked
            blackTimeLbl.setVisible(false);blackTimeLeftLbl.setVisible(f
alse);
            whiteTiimeLbl.setVisible(false);whiteTimeLeftLbl.setVisible(
false);
            accountForPlayerTime=false;
        }
        gameTitleLbl.setText("Game Code: "+gameData[0]);//sets the game
title
        currentIndex=chess.getLengthOfStatesFoFar()-1;
        isWhiteAtBottom=isUserWhite;
        initialiseBoard();
        setTimers();
        timer = new Timer(1000, new ActionListener() {//timer that
updates the timer every second
            @Override
            public void actionPerformed(ActionEvent e) {
                if(accountForPlayerTime){//if player time is  tracked
then the timer values are updated
                    if(game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT
&&!isUserWhite){
                        currentTime++;
                        secondsBlackHas--;
                        if(secondsBlackHas<=0){
                            secondsBlackHas=0;
                            setTimers();
                            //updates record as user has run out of time
and the game is over
                            gameData[4]=String.valueOf(currentTime);game
Data[5]=String.valueOf(secondsWhiteHas);gameData[6]=String.valueOf(secon
dsBlackHas);
                            if(isUserPlayerOne){
                                gameData[8]="true";
                                gameData[11]="0";
                            }else{
                                gameData[11]="1";
                                gameData[9]="true";
                            }
                            JOptionPane.showMessageDialog(null,"Black
has run out of time. White wins!","White Wins",JOptionPane.OK_OPTION);
```

```
                                updateRecordAndGoToMenu();
                                timer.stop();
                                timer=null;
                            }
                        }else
if(!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT&&isUserWhite){
                            currentTime++;
                            secondsWhiteHas--;
                            if(secondsWhiteHas<=0){//if the user has run out
of time, the game is ended
                                secondsWhiteHas=0;
                                gameData[4]=String.valueOf(currentTime);game
Data[5]=String.valueOf(secondsWhiteHas);gameData[6]=String.valueOf(secon
dsBlackHas);
                                if(isUserPlayerOne){
                                    gameData[11]="0";
                                    gameData[8]="true";
                                }else{
                                    gameData[11]="1";
                                    gameData[9]="true";
                                }

                                JOptionPane.showMessageDialog(null,"White
has run out of time. Black wins!","Black Wins",JOptionPane.OK_OPTION);
                                updateRecordAndGoToMenu();
                                timer.stop();
                                timer=null;
                            }
                        }
                    }
                    //updates the current time variable if relevant
                    if(((game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT&&
!isUserWhite)||(!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT&&isUs
erWhite)) && accountForPlayerTime==false){
                        currentTime++;
                    }else{

                    }
                    setTimers();
                }
            });timer.start();//starts the timers
            updateFormBasedOnGameDataRecord();//updates the form based on
the game data stored
            updateImageArrayBasedOnVaraible();//updates the look and feel of
the board depedning for which layout the user has gone for
            updateBoard();//the board is updated to reflect the current
board state
            updateSoundIcon();//updates the sound icon depedning on whether
the game is on mute or not.
            addSquareListeners();//adds action listeners for every square
that is pressed by the user
            this.jPanel1.setDoubleBuffered(true);
            getOpponentMoveIfAppropriate();//look for move from opponent if
```

```
it's their turn to move
    }
    /**
     * Updates the array of image depending on which layout choice the
user has gone for. Basically, the actually images that represent the
chess pieces is altered if needed.
     */
    private void updateImageArrayBasedOnVaraible(){
        try{
            File fileOfPiece;//stores file where image is stored
            BufferedImage bufferedImagePiece;//stores a buffered image
of the piece
            String fileExt="";
             for(int i = 0; i<IMAGES_NAME.length;i++){
                if(isDefaultLayout){
                    fileExt="oldPieceImages";
                }else{
                    fileExt="test";
                }
                fileOfPiece = new
File("Assets"+File.separator+fileExt+File.separator+IMAGES_NAME[i]
+".png");//finds file
                bufferedImagePiece = ImageIO.read(fileOfPiece);
                CustomImageIconBoard cust = new
CustomImageIconBoard(bufferedImagePiece);
                imagePiece[i]= cust;//stores image
            }
            //closes and flushes relevant methods
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An unexpected error
occured when loading the image of pieces: "+e, "Error",
JOptionPane.OK_OPTION);
        }
    }
    /**
     * Gets whether the thread should be killed
     * @return whether the thread should be killed
     */
    private synchronized boolean getKillThread(){
        return killThread;
    }
    /**
     * Updates the game record by requesting the data from the server.
If an error occurs, the user is taken to the menu.
     */
    private void
updateGameDataArrayFromServerAndGoToMenuIfErrorOccurs(){
        try{
            //updates the relevant fields in the game record
            gameData[5]=String.valueOf(secondsWhiteHas);gameData[6]=Stri
ng.valueOf(secondsBlackHas);gameData[4]=String.valueOf(currentTime);
            ThreadClient.updateOnlineGameRecord(gameData);//stores the
new array to the server
```

```
            if(gameData[0]==null){
                JOptionPane.showMessageDialog(this,"An error occurred
when communicating the game result with the server."
,"Error",JOptionPane.OK_OPTION);
                Menu menu = new Menu(session);
                menu.setVisible(true);
                this.dispose();return;
            }
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating the game result with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new Menu(session);
            menu.setVisible(true);
            this.dispose();return;
        }
    }
    /**
     * Saves the game data array to the server. If an error occurs, then
the user is taken to the menu
     */
    private synchronized void saveRecordToServer(){
         try{
             //updates the game record
            gameData[5]=String.valueOf(secondsWhiteHas);gameData[6]=Stri
ng.valueOf(secondsBlackHas);gameData[4]=String.valueOf(currentTime);
            ThreadClient.updateOnlineGameRecord(gameData);//sends the
updated record to the server
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating the game result with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new Menu(session);
            menu.setVisible(true);
            this.dispose();return;
        }
    }
    /**
     * Updates the game record (by storing the game data array in the
server) and goes to the menu
     */
    private void updateRecordAndGoToMenu(){
        try{
            //updates the record
            gameData[5]=String.valueOf(secondsWhiteHas);gameData[6]=Stri
ng.valueOf(secondsBlackHas);gameData[4]=String.valueOf(currentTime);
            ThreadClient.updateOnlineGameRecord(gameData);//saves record
to server
            OnlineMenu menu = new OnlineMenu(session);//returns to
online menu
            menu.setVisible(true);
            this.dispose();
            return;
```

```
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating the game result with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new Menu(session);
            menu.setVisible(true);
            this.dispose();return;
        }
    }
    /**
     * Updates the form after the game data array has been updated
     */
    private void updateFormBasedOnGameDataRecord(){
        //sets labels displaying usernames for each colour and ELO
scores
        if(gameData[10].equals("true")){
            whiteLbl.setText("White: "+gameData[1]);
            whiteEloLbl.setText("ELO: "+eloPlayerOne);
            blackLbl.setText("Black: "+gameData[2]);
            blackEloLbl.setText("ELO: "+eloPlayerTwo);
        }else{
            whiteLbl.setText("White: "+gameData[2]);
            whiteEloLbl.setText("ELO: "+eloPlayerTwo);
            blackLbl.setText("Black: "+gameData[1]);
            blackEloLbl.setText("ELO: "+eloPlayerOne);
        }
        invisibleDrawBtn.setEnabled(false);
        invisibleDrawBtn.setVisible(false);
        //checks to see whether the user has offered a draw or been
offered a draw and updates the form correspondingly
        if((isUserPlayerOne&&gameData[12].equals("false") &&
gameData[13].equals("true")) ||
(!isUserPlayerOne&&gameData[13].equals("false") &&
gameData[12].equals("true"))){//if a draw is being offered to the
player, this code is executed
            invisibleDrawBtn.setEnabled(true);invisibleDrawBtn.setVisibl
e(true);
            offerDrawBtn.setBackground(Color.red);offerDrawBtn.setText("
Reject Draw Offer");
            invisibleDrawBtn.setBackground(Color.green);invisibleDrawBtn
.setText("Accept draw offer");
        }else if((isUserPlayerOne&&gameData[12].equals("true") &&
gameData[13].equals("false")) ||
(!isUserPlayerOne&&gameData[13].equals("true") &&
gameData[12].equals("false"))){//checks to see if a draw has been
offered but not accepted
            offerDrawBtn.setText("Withdraw draw
offer");offerDrawBtn.setBackground(DEFAULT_BUTTON_BACKGROUND);
        }else
if(gameData[12].equals("true")&&gameData[13].equals("true")){//checks to
see if a draw has occurred
        /*    JOptionPane.showMessageDialog(this, "The game was a
draw.","Draw!",JOptionPane.OK_OPTION);
```

```
            gameData[11]="0.5";//sets the game state to a draw
            if(isUserPlayerOne){//stores the fact that the user has seen
the game result
                gameData[8]="true";
            }else{
                gameData[9]="true";
            }
            updateRecordAndGoToMenu(); */
        }else{//no draw has been offered
            offerDrawBtn.setText("Offer
Draw");offerDrawBtn.setBackground(DEFAULT_BUTTON_BACKGROUND);
            invisibleDrawBtn.setEnabled(false);invisibleDrawBtn.setVisib
le(false);
        }
        //stores all board states -adding any on if relevant
        String boards[]=gameData[3].split(OfflinePlay.SPLIT_FEN_STRING);
        for(int i=game.getLengthOfStatesFoFar();i<boards.length;i++){
            game.addState(new
ChessBoardState(boards[i]));currentIndex++;
        }
       // secondsWhiteHas=Integer.parseInt(gameData[5]);
        //secondsBlackHas=Integer.parseInt(gameData[6]);
       // currentTime=Integer.parseInt(gameData[4]);
       //this code used to check whether the board was in a terminal
state. However, this led to issues with threads as multiple different
methods were declaring the game
       //to be over which led to issues. This code has been left in case
it is needed to debug in the future
       /*
        if((gameData[11].equals("1")&&gameData[10].equals("true")) ||
(gameData[11].equals("0")&&gameData[10].equals("false"))){
            JOptionPane.showMessageDialog(this, "White wins!","White
win",JOptionPane.OK_OPTION);
            if(isUserPlayerOne){//stores the fact that the user has seen
the game result
                gameData[8]="true";
            }else{
                gameData[9]="true";
            }
            updateRecordAndGoToMenu();
            return;
        }else if(gameData[11].equals("0.5") ||
(gameData[12].equals("true")&&gameData[13].equals("true"))){
            JOptionPane.showMessageDialog(this, "The game was
drawn!","Draw",JOptionPane.OK_OPTION);
            if(isUserPlayerOne){//stores the fact that the user has seen
the game result
                gameData[8]="true";
            }else{
                gameData[9]="true";
            }
            updateRecordAndGoToMenu();
            return;
```

```
        }else if((gameData[11].equals("0")&&gameData[10].equals("true"))
|| (gameData[11].equals("1")&&gameData[10].equals("false"))){
            JOptionPane.showMessageDialog(this, "Black Won!","Black
Win",JOptionPane.OK_OPTION);
            if(isUserPlayerOne){//stores the fact that the user has seen
the game result
                gameData[8]="true";
            }else{
                gameData[9]="true";
            }
            updateRecordAndGoToMenu();
            return;
        }
        */

    }
    /**
     * Updates the sound icon based on whether sound is currently on or
off
     */
    private void updateSoundIcon(){
        if(soundIsOn){
            speakerLbl.setIcon(soundOnIcon);
        }else{
            speakerLbl.setIcon(muteIcon);
        }
    }
    /**
     * Updates various components of board to reflect currently selected
board state
     */
    public void updateBoard(){
        ChessBoardState state = game.getState(currentIndex);//gets state
specified
        viewPreviousBoard.setEnabled(true);nextBoardBtn.setEnabled(true)
;viewPresentBoard.setEnabled(true);
        moveNumberLbl.setText("Move Number:
"+String.valueOf(state.FULL_MOVE_NUMBER));
        if(currentIndex==0){//sets it so that no previous states can be
accessed
            viewPreviousBoard.setEnabled(false);
        }
        if(currentIndex==game.getLengthOfStatesFoFar()-1){
            nextBoardBtn.setEnabled(false);//sets it so that view next
board is not enabled
            viewPresentBoard.setEnabled(false);
        }
        //adds visual mark to display who's turn it is to play
        if(state.IS_BLACK_TURN_TO_PLAY_NEXT){
            isBlackTurnLbl.setText("*");isWhiteTurnLbl.setText("");
        }else{
            isBlackTurnLbl.setText("");isWhiteTurnLbl.setText("*");
        }
```

```
        if(game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
            blackTimeLbl.setText("*");whiteTiimeLbl.setText("");
        }else{
            blackTimeLbl.setText("");whiteTiimeLbl.setText("*");
        }
        int[][]board=state.getCopyOfBoard();//gets a copy of the game
board
        if(isWhiteAtBottom==false){//checks to see whether board needs
to be flipped
            //rotates board by 180 degrees
            rotateMatrix(board);
        }
        //disables resign button if game is played against computer and
its the computers turn
        yCoordSquareClicked=-1;xCoordSquareClicked=-1;
        legalSquaresToMoveTo=new ArrayList<>();
        //loads the board and the corresponding images
        int yKing=-1;int xKing=-1;
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
               //sets square to the appropriate colour
                if(y%2==x%2){
                    if(isDefaultLayout){
                        boardVisual[y][x].setBackground(Color.white);
                    }else{
                        boardVisual[y][x].setBackground(Color.decode("#F
1D9C0"));
                    }
                }else{
                    if(isDefaultLayout){
                        boardVisual[y][x].setBackground(Color.black);
                    }else{
                        boardVisual[y][x].setBackground(Color.decode("#A
97A65"));
                    }
                }
                //sets board labels to the appropriate piece image
                switch (board[y][x]) {
                    case ChessBoardState.EMPTY_VALUE:
                        boardVisual[y][x].setIcon(null);
                        break;
                    case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[11]);
                        break;
                    case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[5]);
                        break;
                    case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[8]);
                        break;
```

```
                    case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[2]);
                        break;
                    case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[10]);
                        break;
                    case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[4]);
                        break;
                    case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[9]);
                        break;
                    case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[3]);
                        break;
                    case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[7]);
                        break;
                    case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[1]);
                        break;
                    case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        boardVisual[y][x].setIcon(imagePiece[6]);
                        if(!state.IS_BLACK_TURN_TO_PLAY_NEXT){
                            yKing=y;xKing=x;
                        }
                        break;
                    default:
                        //black king
                        boardVisual[y][x].setIcon(imagePiece[0]);
                        if(state.IS_BLACK_TURN_TO_PLAY_NEXT){
                            yKing=y;xKing=x;
                        }
                        break;
                }
            }
        }
        //sets the king's square to be red if it is in check
        if(ChessGame.isInCheck(!state.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(state.BOARD, state.CASTLING_RIGHTS,
!state.IS_BLACK_TURN_TO_PLAY_NEXT, state.FULL_MOVE_NUMBER,
state.HALF_MOVE_CLOCK,
state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT))){//if king is in check,
its square is coloured red
            boardVisual[yKing][xKing].setBackground(Color.red);
```

```
        }
    }
    /**
     * Creates a thread that periodically queries the server for a move
from their opponent
     */
    private void getOpponentMoveIfAppropriate(){
        //creates a thread that contsantly looking to see if the
opponent has made a move. If they have, the board is updated accordingly
            getOpponentMoveThread=new Thread( new Runnable(){
                @Override
                public void run(){
                    int lengthOfBoard=game.getLengthOfStatesFoFar()-1;
                    while(!getKillThread()){
                       synchronized(this){
                            // refreshses the game
                            setGameDataUpToData();
                            if(getKillThread()){//checks to see if
thread is to be killed
                                break;
                            }
                            //updates form based on game data
                            updateFormBasedOnGameDataRecord();
                            if(getKillThread()){
                                break;
                            }
                            if(currentIndex!=lengthOfBoard){
                                //updates board and makes a move if
appropraite
                                lengthOfBoard=currentIndex;
                                updateBoard();
                                playMoveMadeSound();
                                currentTime=Integer.parseInt(gameData[4]
);secondsWhiteHas=Integer.parseInt(gameData[5]);secondsBlackHas=Integer.
parseInt(gameData[6]);
                            }
                            if(getKillThread()){
                                break;
                            }
                            checkForTerminalState();//checks to see
whether the board is in a terminal state
                        }
                        try{
                            Thread.sleep(1000);//pauses the thread to
prevent the thread from neeedlessly spamming the server
                        }catch(InterruptedException e){

                        }
                    }
                }
            });getOpponentMoveThread.start();//starts the thread

    }
```

```
    /**
     * Updates the variable isDefault layout based on text file contents
     */
    private void updateLayoutVariableBasedOnTextFile(){
        //gets the board configuration to use
        try{
            FileReader read = new FileReader(BOARD_SETTING);
            BufferedReader buffRead = new BufferedReader(read);
            isDefaultLayout=buffRead.readLine().equals("true");
            buffRead.close();read.close();
        }catch(Exception e){
            System.out.println(e);
        }
    }
    /**
     * Sets the values for all timers  that are appropriate
     */
    private void setTimers(){
        //sets value of overall timer
        int sec=currentTime%60;
        int hour=currentTime/60;
        int min=hour%60;
        hour=hour/60;
        String s=String.valueOf(sec);if(s.length()==1){s="0"+s;}//makes
seconds at least two digits
        String m = String.valueOf(min);if(m.length()==1){m="0"+m;}
        String h=String.valueOf(hour);if(h.length()==1){h="0"+h;}
        timerLbl.setText(h+":"+m+":"+s);
        if(accountForPlayerTime){//sets time left for each colour if
relevant
            //sets time in format of hours, minutes and seconds.
                m=String.valueOf(secondsBlackHas/60);
                if(m.length()==1){
                    m="0"+m;
                }
                s=String.valueOf(secondsBlackHas%60);
                if(s.length()==1){
                    s="0"+s;
                }
                blackTimeLeftLbl.setText("Black Time: "+m+":"+s);
                m=String.valueOf(secondsWhiteHas/60);
                if(m.length()==1){
                    m="0"+m;
                }
                s=String.valueOf(secondsWhiteHas%60);
                if(s.length()==1){
                    s="0"+s;
                }
                whiteTimeLeftLbl.setText("White Time:
"+m+":"+s);
        }
    }
    /**
```

```
     * This rotates a chess board array by 180 degrees.
     * @param board The board (must be an 8*8 array)
     */
    private static void rotateMatrix(int[][] board) {
        int temp;//used as a buffer
        for (int i = 0; i <= 3; i++) {
            for (int j = 0; j < 8; j++) {
                temp = board[i][j];
                board[i][j] = board[7 - i ][7- j];
                board[7-i][7- j] = temp;
            }
        }
    }
    /**
     * This methods creates the board pieces and is used to initialise
the board. It also changes colour of squares
     * depending on whether the user is white.
     */
    private void initialiseBoard(){
        //creates JLabels and loads them to array
        final int WIDTH_OF_SQUARE=56;//stores the width of the square
        final int HEIGHT_OF_SQUARE=56;//stores the height of the square
        final int SPACE_ABOVE_BOARD=10;//stores the space above the
board
        final int SPACE_TO_RIGHT_OF_BOARD=50;//stores the gap between
the right edge of the screen and the board
        int xCoordOfSquare=SPACE_TO_RIGHT_OF_BOARD;//stores the
coordinate of the square
        int yCoordOfSquare=SPACE_ABOVE_BOARD;//stores the y coordinate
of the square
        final String
RANK_AND_FILE[]={"a","b","c","d","e","f","g","h","1","2","3","4","5","6"
,"7","8"};//stores the values to be used by labels that indicate the
coordinates on the board
        final int HEIGHT_OF_LABEL=16;//stores the height of the label
showing the rank or file of the square
        final int DISTANCE_OF_LETTERS_BELOW=3;//stores distance between
squares and letters
        //adds colour squares to GUI
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                boardVisual[y][x]=new BoardJLabel();
                boardVisual[y][x].setSize(WIDTH_OF_SQUARE,
HEIGHT_OF_SQUARE);//sets size of label
                boardVisual[y][x].setLocation(xCoordOfSquare,
yCoordOfSquare);
                //sets square to the approrpiate colour
                if(y%2==x%2){
                    //boardVisual[y][x].setBackground(Color.white);
                    boardVisual[y][x].setBackground(Color.decode("#F1D9C
0"));
                }else{
                    //boardVisual[y][x].setBackground(Color.black);
```

```
                    boardVisual[y][x].setBackground(Color.decode("#A97A6
5"));
                }
                boardVisual[y][x].setOpaque(true);//allows piece's
background to be seen
                boardVisual[y][x].setDoubleBuffered(true);
                this.jPanel1.add(boardVisual[y][x]);
                if(x==7){//updates coordinates of next label
                    xCoordOfSquare=SPACE_TO_RIGHT_OF_BOARD;
                    yCoordOfSquare=yCoordOfSquare+HEIGHT_OF_SQUARE;
                }else{
                    xCoordOfSquare=xCoordOfSquare+WIDTH_OF_SQUARE;
                }
            }
        }
        //adds labels to indicate rank and file of pieces
        xCoordOfSquare=SPACE_TO_RIGHT_OF_BOARD+(WIDTH_OF_SQUARE/2)-
(HEIGHT_OF_LABEL/2);
        yCoordOfSquare=yCoordOfSquare+DISTANCE_OF_LETTERS_BELOW;
        JLabel lbl;//stores label to be added
        for(int indexForCoord=0;indexForCoord<8;indexForCoord++){//adds
letters to board which indicate file
            lbl=new JLabel(RANK_AND_FILE[indexForCoord]);
            lbl.setSize(HEIGHT_OF_LABEL, HEIGHT_OF_LABEL);
            lbl.setLocation(xCoordOfSquare,yCoordOfSquare);
            xCoordOfSquare=xCoordOfSquare+WIDTH_OF_SQUARE;
            this.jPanel1.add(lbl);
        }
        yCoordOfSquare=SPACE_ABOVE_BOARD+(HEIGHT_OF_SQUARE/2)-
(HEIGHT_OF_LABEL/2);
        xCoordOfSquare=SPACE_TO_RIGHT_OF_BOARD-
DISTANCE_OF_LETTERS_BELOW-HEIGHT_OF_LABEL;
        for(int indexForCoord=15;indexForCoord>=8;indexForCoord--){
            lbl=new JLabel(RANK_AND_FILE[indexForCoord]);
            lbl.setSize(HEIGHT_OF_LABEL, HEIGHT_OF_LABEL);
            lbl.setLocation(xCoordOfSquare,yCoordOfSquare);
            yCoordOfSquare=yCoordOfSquare+HEIGHT_OF_SQUARE;
            this.jPanel1.add(lbl);
        }
        //loads all images and stores them
        try{
            File fileOfPiece;//stores file where image is stored
            BufferedImage bufferedImagePiece=null;//stores a buffered
image of the piece
            Image imgPiece=null;//stores the actual image of the piece
            ImageIcon imageIconPiece;//stores the icon of the image
             for(int i = 0; i<IMAGES_NAME.length;i++){
                fileOfPiece = new
File("Assets"+File.separator+"test"+File.separator+IMAGES_NAME[i]
+".png");//finds file
                bufferedImagePiece = ImageIO.read(fileOfPiece);
                //imgPiece =
bufferedImagePiece.getScaledInstance(WIDTH_OF_SQUARE, HEIGHT_OF_SQUARE ,
```

```
Image.SCALE_SMOOTH);//loads image of piece and scales it to size
                CustomImageIconBoard cust = new
CustomImageIconBoard(bufferedImagePiece);
                //imageIconPiece = new ImageIcon(imgPiece);
                imagePiece[i]= cust;//stores image
            }

             //gets speaker icons
            fileOfPiece=new File("Assets"+File.separator +MUTE_PICTURE);
             bufferedImagePiece=ImageIO.read(fileOfPiece);
             imgPiece=bufferedImagePiece.getScaledInstance(speakerLbl.ge
tPreferredSize().width, speakerLbl.getPreferredSize().width,
Image.SCALE_SMOOTH);
             muteIcon=new CustomImageIconBoardSpeaker(imgPiece);
             fileOfPiece=new File("Assets"+File.separator
+SOUND_ON_PICTURE);
             bufferedImagePiece=ImageIO.read(fileOfPiece);
             imgPiece=bufferedImagePiece.getScaledInstance(speakerLbl.ge
tPreferredSize().width, speakerLbl.getPreferredSize().width,
Image.SCALE_SMOOTH);
             soundOnIcon=new CustomImageIconBoardSpeaker(imgPiece);
            //closes and flushes relevant methods
            imgPiece.flush();bufferedImagePiece.flush();
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An unexpected error
occured when loading the image of pieces: "+e, "Error",
JOptionPane.OK_OPTION);
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        backBtn = new javax.swing.JButton();
        resignBtn = new javax.swing.JButton();
        viewPreviousBoard = new javax.swing.JButton();
        viewPresentBoard = new javax.swing.JButton();
        nextBoardBtn = new javax.swing.JButton();
        timerLbl = new javax.swing.JLabel();
        timerLbl1 = new javax.swing.JLabel();
        blackLbl = new javax.swing.JLabel();
        whiteLbl = new javax.swing.JLabel();
        isBlackTurnLbl = new javax.swing.JLabel();
        isWhiteTurnLbl = new javax.swing.JLabel();
```

```
        moveNumberLbl = new javax.swing.JLabel();
        flipBoardBtn = new javax.swing.JButton();
        blackTimeLeftLbl = new javax.swing.JLabel();
        whiteTimeLeftLbl = new javax.swing.JLabel();
        gameTitleLbl = new javax.swing.JLabel();
        blackTimeLbl = new javax.swing.JLabel();
        whiteTiimeLbl = new javax.swing.JLabel();
        speakerLbl = new BoardJLabel();
        whiteEloLbl = new javax.swing.JLabel();
        blackEloLbl = new javax.swing.JLabel();
        offerDrawBtn = new javax.swing.JButton();
        refreshBtn = new javax.swing.JButton();
        invisibleDrawBtn = new javax.swing.JButton();
        changeBoardBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Offline Game");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosed(java.awt.event.WindowEvent evt) {
                formWindowClosed(evt);
            }
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        backBtn.setText("Back");
        backBtn.setDoubleBuffered(true);
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        resignBtn.setText("Resign");
        resignBtn.setDoubleBuffered(true);
        resignBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                resignBtnActionPerformed(evt);
            }
        });

        viewPreviousBoard.setText("Previous Board");
        viewPreviousBoard.setDoubleBuffered(true);
        viewPreviousBoard.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewPreviousBoardActionPerformed(evt);
```

```
            }
        });

        viewPresentBoard.setText("Current Board");
        viewPresentBoard.setDoubleBuffered(true);
        viewPresentBoard.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewPresentBoardActionPerformed(evt);
            }
        });

        nextBoardBtn.setText("Next Board");
        nextBoardBtn.setDoubleBuffered(true);
        nextBoardBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                nextBoardBtnActionPerformed(evt);
            }
        });

        timerLbl.setText(" ");
        timerLbl.setDoubleBuffered(true);

        timerLbl1.setText("Timer:");
        timerLbl1.setDoubleBuffered(true);

        blackLbl.setBackground(new java.awt.Color(204, 204, 204));
        blackLbl.setText("Black:");
        blackLbl.setDoubleBuffered(true);
        blackLbl.setOpaque(true);

        whiteLbl.setBackground(new java.awt.Color(255, 255, 255));
        whiteLbl.setText("White:");
        whiteLbl.setDoubleBuffered(true);
        whiteLbl.setOpaque(true);

        isBlackTurnLbl.setForeground(new java.awt.Color(255, 0, 0));
        isBlackTurnLbl.setText("*");
        isBlackTurnLbl.setDoubleBuffered(true);

        isWhiteTurnLbl.setForeground(new java.awt.Color(255, 0, 0));
        isWhiteTurnLbl.setText("*");
        isWhiteTurnLbl.setDoubleBuffered(true);

        moveNumberLbl.setText("Move Number: ");
        moveNumberLbl.setDoubleBuffered(true);

        flipBoardBtn.setText("Flip Board");
        flipBoardBtn.setDoubleBuffered(true);
        flipBoardBtn.addActionListener(new
```

```
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                flipBoardBtnActionPerformed(evt);
            }
        });

        blackTimeLeftLbl.setBackground(new java.awt.Color(204, 204,
204));
        blackTimeLeftLbl.setText("Black Time: ");
        blackTimeLeftLbl.setDoubleBuffered(true);
        blackTimeLeftLbl.setOpaque(true);

        whiteTimeLeftLbl.setBackground(new java.awt.Color(255, 255,
255));
        whiteTimeLeftLbl.setText("White Time:");
        whiteTimeLeftLbl.setDoubleBuffered(true);
        whiteTimeLeftLbl.setOpaque(true);

        gameTitleLbl.setText("Game Code: ");
        gameTitleLbl.setDoubleBuffered(true);

        blackTimeLbl.setForeground(new java.awt.Color(255, 0, 0));
        blackTimeLbl.setText("*");
        blackTimeLbl.setDoubleBuffered(true);

        whiteTiimeLbl.setForeground(new java.awt.Color(255, 0, 0));
        whiteTiimeLbl.setText("*");
        whiteTiimeLbl.setDoubleBuffered(true);

        speakerLbl.setPreferredSize(new java.awt.Dimension(15, 15));
        speakerLbl.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                speakerLblMouseClicked(evt);
            }
        });

        whiteEloLbl.setBackground(new java.awt.Color(255, 255, 255));
        whiteEloLbl.setText("ELO: ");
        whiteEloLbl.setDoubleBuffered(true);
        whiteEloLbl.setOpaque(true);

        blackEloLbl.setBackground(new java.awt.Color(204, 204, 204));
        blackEloLbl.setText("ELO:");
        blackEloLbl.setDoubleBuffered(true);
        blackEloLbl.setOpaque(true);

        offerDrawBtn.setText("Offer Draw");
        offerDrawBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                offerDrawBtnActionPerformed(evt);
```

```
            }
        });

        refreshBtn.setText("Refresh");
        refreshBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                refreshBtnActionPerformed(evt);
            }
        });

        invisibleDrawBtn.setText("Offer Draw");
        invisibleDrawBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                invisibleDrawBtnActionPerformed(evt);
            }
        });

        changeBoardBtn.setText("Chane Board");
        changeBoardBtn.setActionCommand("Change Board");
        changeBoardBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                changeBoardBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.TRAILING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap(572, Short.MAX_VALUE)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                .addComponent(moveNumberLbl)
                                .addGap(107, 107, 107))
                            .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING)
```

```
                                        .addGroup(jPanel1Layout.createSequen
tialGroup()
                                            .addComponent(invisibleDrawBtn)
                                            .addGap(18, 18, 18)
                                            .addComponent(offerDrawBtn)
                                            .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED)
                                            .addComponent(resignBtn))
                                        .addGroup(jPanel1Layout.createSequen
tialGroup()
                                            .addComponent(viewPreviousBoard)
                                            .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED)
                                            .addComponent(viewPresentBoard)
                                            .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.UNRELATED)
                                            .addComponent(nextBoardBtn)))
                                    .addGap(25, 25, 25))
                                .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING,
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING, false)
                                    .addGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING, jPanel1Layout.createSequentialGroup()
                                        .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                            .addComponent(isWhiteTurnLbl)
                                            .addComponent(isBlackTurnLbl))
                                        .addGap(18, 18, 18)
                                        .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                            .addComponent(blackLbl)
                                            .addComponent(whiteLbl))
                                        .addPreferredGap(javax.swing.LayoutS
tyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                        .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                            .addComponent(blackEloLbl)
                                            .addComponent(whiteEloLbl)))
                                    .addGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING, jPanel1Layout.createSequentialGroup()
                                        .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                            .addComponent(blackTimeLbl)
                                            .addComponent(whiteTiimeLbl))
                                        .addPreferredGap(javax.swing.LayoutS
tyle.ComponentPlacement.UNRELATED)
                                        .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                            .addComponent(whiteTimeLeftLbl)
                                            .addComponent(blackTimeLeftLbl))
                                        .addGap(253, 253, 253)))
```

```
                                .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING,
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING, false)
                                    .addGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING, jPanel1Layout.createSequentialGroup()
                                        .addComponent(changeBoardBtn)
                                        .addPreferredGap(javax.swing.LayoutS
tyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                        .addComponent(backBtn))
                                    .addGroup(javax.swing.GroupLayout.Alignm
ent.TRAILING, jPanel1Layout.createSequentialGroup()
                                        .addComponent(flipBoardBtn)
                                        .addGap(33, 33, 33)
                                        .addComponent(refreshBtn)
                                        .addGap(195, 195, 195)))))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(0, 0, Short.MAX_VALUE)
                            .addComponent(gameTitleLbl)
                            .addGap(184, 184, 184)
                            .addComponent(timerLbl1)))
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addComponent(timerLbl)
                            .addGap(27, 27, 27))
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                            .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                            .addComponent(speakerLbl,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addContainerGap())))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                        .addComponent(timerLbl)
                        .addComponent(timerLbl1)
                        .addComponent(gameTitleLbl))
                .addGap(58, 58, 58)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGroup(jPanel1Layout.createParallelGroup(java
```

```
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(blackLbl)
                            .addComponent(isBlackTurnLbl))
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.UNRELATED)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(whiteLbl)
                            .addComponent(isWhiteTurnLbl)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(blackEloLbl)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.UNRELATED)
                        .addComponent(whiteEloLbl)))
                .addGap(36, 36, 36)
                .addComponent(moveNumberLbl)
                .addGap(52, 52, 52)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(blackTimeLeftLbl)
                    .addComponent(blackTimeLbl))
                .addGap(18, 18, 18)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(whiteTimeLeftLbl)
                    .addComponent(whiteTiimeLbl))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 22, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(viewPreviousBoard)
                            .addComponent(viewPresentBoard)
                            .addComponent(nextBoardBtn))
                        .addGap(31, 31, 31)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(resignBtn)
                            .addComponent(offerDrawBtn)
                            .addComponent(invisibleDrawBtn)))
                    .addComponent(speakerLbl,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 52,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(16, 16, 16)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(flipBoardBtn)
                    .addComponent(refreshBtn))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
```

```
ent.RELATED)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(backBtn)
                    .addComponent(changeBoardBtn))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void flipBoardBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //flips the board
        isWhiteAtBottom=!isWhiteAtBottom;
        updateBoard();
    }

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //saves the game and takes the user back to the online menu
        saveRecordToServer();
        OnlineMenu online = new OnlineMenu(session);
        online.setVisible(true);
        this.dispose();return;
    }

    private void
viewPreviousBoardActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes user to previous board state
        currentIndex--;
        updateBoard();
    }

    private void nextBoardBtnActionPerformed(java.awt.event.ActionEvent
evt) {
```

```
        // takes user to a more recent board state
        currentIndex++;
        updateBoard();
    }

    private void
viewPresentBoardActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes user to current board
        currentIndex=game.getLengthOfStatesFoFar()-1;
        updateBoard();
    }

    private void resignBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // gets player to resign
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to resign? Doing so means you will lose the
game.","Resign?",JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            killThread=true;
            if(isUserWhite==false){
                JOptionPane.showMessageDialog(this, "Black resigned -
White wins!","White Wins!",JOptionPane.OK_OPTION);
            }else{
                JOptionPane.showMessageDialog(this, "White resigned -
Black wins!","Black Wins!",JOptionPane.OK_OPTION);
            }
            if(isUserPlayerOne){
                gameData[11]="0";//sets the game result
                gameData[8]="true";
            }else{
                gameData[11]="1";
                gameData[9]="true";
            }
            saveRecordToServer();//updates game record
            //takes user back to menu
            OnlineMenu online = new OnlineMenu(session);
            online.setVisible(true);
            this.dispose();return;
        }
    }

    private void formWindowClosed(java.awt.event.WindowEvent evt)
{
      //updates file when window is closed and kills the thread that is
getting the opponent move
        if(isUserWhite &&
!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
          gameData[4]=String.valueOf(currentTime);gameData[5]=String.val
ueOf(secondsWhiteHas);gameData[6]=String.valueOf(secondsBlackHas);
      }else if(!isUserWhite
&&game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
```

```
          gameData[4]=String.valueOf(currentTime);gameData[5]=String.val
ueOf(secondsWhiteHas);gameData[6]=String.valueOf(secondsBlackHas);
      }
        saveRecordToServer();
        if(getOpponentMoveThread!=null){//kills thread
            getOpponentMoveThread.interrupt();
            killThread=true;
        }
        getOpponentMoveThread=null;
        if(timer!=null){
          timer.stop();
          timer=null;
      }
    }

    private void formWindowClosing(java.awt.event.WindowEvent evt)
{
      //if a user closes the window by pressing the cross in the top
right corner, this method is invoked. It saves the board state
      if(isUserWhite &&
!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
          gameData[4]=String.valueOf(currentTime);gameData[5]=String.val
ueOf(secondsWhiteHas);gameData[6]=String.valueOf(secondsBlackHas);
      }else if(!isUserWhite
&&game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
          gameData[4]=String.valueOf(currentTime);gameData[5]=String.val
ueOf(secondsWhiteHas);gameData[6]=String.valueOf(secondsBlackHas);
      }
      saveRecordToServer();
      if(timer!=null){
          timer.stop();
          timer=null;
      }
      if(getOpponentMoveThread!=null){
            getOpponentMoveThread.interrupt();
            killThread=true;
        }
        getOpponentMoveThread=null;
    }

    private void speakerLblMouseClicked(java.awt.event.MouseEvent evt)
{
    //updates whether sound is on or not and sets speaker icon to
appropraite value.
    soundIsOn=!soundIsOn;
    updateSoundIcon();
    }

    private void refreshBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // refreshses the game
        int gameLength=game.getLengthOfStatesFoFar()-1;
        setGameDataUpToData();
```

```
        updateFormBasedOnGameDataRecord();
        if(currentIndex!=gameLength){
            updateBoard();
            playMoveMadeSound();
        }
        checkForTerminalState();
    }

    private void offerDrawBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // handles draw offers
        if(offerDrawBtn.getText().equals("Offer Draw")){
                if(isUserPlayerOne){
                    gameData[12]="true";
                }else{
                    gameData[13]="true";
                }
                //offers a draw and updates the online record
                try{
                    ThreadClient.updateOnlineGameRecord(gameData);
                }catch(Exception e){
                    JOptionPane.showMessageDialog(this,"An unexpected
error occurred when communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
                    Menu menu = new Menu(session);
                    menu.setVisible(true);
                    this.dispose();return;
                }
                updateGameDataArrayFromServerAndGoToMenuIfErrorOccurs();
                updateFormBasedOnGameDataRecord();
        }else if(offerDrawBtn.getText().equals("Withdraw draw offer")){
            //withdraws the draw offer
                if(isUserPlayerOne){
                    gameData[12]="false";
                }else{
                    gameData[13]="false";
                }
                //updates game data
                try{
                    ThreadClient.updateOnlineGameRecord(gameData);
                }catch(Exception e){
                    JOptionPane.showMessageDialog(this,"An unexpected
error occurred when communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
                    Menu menu = new Menu(session);
                    menu.setVisible(true);
                    this.dispose();return;
                }
                updateGameDataArrayFromServerAndGoToMenuIfErrorOccurs();
//updates form following draw handling
                updateFormBasedOnGameDataRecord();
        }else{//rejects draw offer
            gameData[12]="false";gameData[13]="false";
```

```
                //updates game data
                try{
                    ThreadClient.updateOnlineGameRecord(gameData);
                }catch(Exception e){
                    JOptionPane.showMessageDialog(this,"An unexpected
error occurred when communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
                    Menu menu = new Menu(session);
                    menu.setVisible(true);
                    this.dispose();return;
                }
                updateGameDataArrayFromServerAndGoToMenuIfErrorOccurs();
                updateFormBasedOnGameDataRecord();
        }
    }

    private void
invisibleDrawBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //this accepts a draw offer and saves this to the server
        killThread=true;getOpponentMoveThread.interrupt();timer.stop();
        synchronized(this){
            // this button can only be pressed to accept a draw offer
            gameData[11]="0.5";//sets the game to a draw
            gameData[12]="true";gameData[13]="true";
            JOptionPane.showMessageDialog(this,"The game was a
draw.","Draw",JOptionPane.OK_OPTION);
            if(isUserPlayerOne){
                gameData[8]="true";
            }else{
                gameData[9]="true";
            }
            //updates record stored by server
            saveRecordToServer();
            //takes user back to menu
            OnlineMenu online = new OnlineMenu(session);
            online.setVisible(true);
            this.dispose(); return;
        }
    }

    private void
changeBoardBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // updates the variable is Default layout and stores it to a
text file
        isDefaultLayout=!isDefaultLayout;
        try{
            FileWriter write = new FileWriter(BOARD_SETTING,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write(String.valueOf(isDefaultLayout));
            buffWrite.flush();write.flush();
            buffWrite.close();write.close();
```

```
        }catch(IOException e){
            System.out.println(e);
        }
        updateImageArrayBasedOnVaraible();//updates form following the
change in board layout
        updateBoard();
    }
    /**
     * Updates the game data variable by requesting the game data from
the server
     */
    private void setGameDataUpToData(){
        try{
            gameData=ThreadClient.getGameData(Integer.parseInt(gameData[
0]));
            if(gameData[0]==null){
                JOptionPane.showMessageDialog(this,"An unexpected error
occurred.","Error",JOptionPane.OK_OPTION);
                OnlineMenu menu= new OnlineMenu(session);
                menu.setVisible(true);
                this.dispose();
                return;
            }
            //secondsWhiteHas=Integer.parseInt(gameData[5]);
            //secondsBlackHas=Integer.parseInt(gameData[6]);
            //currentTime=Integer.parseInt(gameData[4]);
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
            OnlineMenu menu= new OnlineMenu(session);
            menu.setVisible(true);
            this.dispose();
            return;
        }
    }
    /**
     * Checks to see if current board is in a terminal state. If it is,
the game result is announced and then the system terminates
     */
    private void checkForTerminalState(){
        if((gameData[12].equals("true") &&
gameData[13].equals("true"))|| (gameData[11].equals("0.5"))){//checks
for a draw
            killThread=true;
            JOptionPane.showMessageDialog(this,"The game was a
draw.","Draw",JOptionPane.OK_OPTION);
            gameData[11]="0.5";
            if(isUserPlayerOne){
                gameData[8]="true";
            }else{
                gameData[9]="true";
            }
```

```
            saveRecordToServer();//updates record and quits the game
            OnlineMenu online = new OnlineMenu(session);
            online.setVisible(true);this.dispose();
            return;
        }
        if(gameData[11].equals("-1")==false){//checks to see if a game
result has been registered by another player
            killThread=true;
            if((gameData[11].equals("0")&& gameData[10].equals("true"))
|| (gameData[11].equals("1")&&gameData[10].equals("false"))){
                JOptionPane.showMessageDialog(this,"Black wins!","Black
win!",JOptionPane.OK_OPTION);
            }else if((gameData[11].equals("1")&&
gameData[10].equals("true")) ||
(gameData[11].equals("0")&&gameData[10].equals("false"))){
                JOptionPane.showMessageDialog(this,"Black wins!","Black
win!",JOptionPane.OK_OPTION);
            }else{
                JOptionPane.showMessageDialog(this,"Draw!","Draw",JOptio
nPane.OK_OPTION);
            }
            if(isUserPlayerOne){//sets the appropriate variable to show
that the player has seen the game result
                gameData[8]="true";
            }else{
                gameData[9]="true";
            }
            try{
                ThreadClient.updateOnlineGameRecord(gameData);//updates
form to server
            }catch(Exception  e){
                JOptionPane.showMessageDialog(this,"An error occurred
when communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
                Menu menu = new Menu(session);
                menu.setVisible(true);
                this.dispose();
                return;
            }
            OnlineMenu online = new OnlineMenu(session);
            online.setVisible(true);this.dispose();
            return;
        }
        if(game.getAllLegalBoardsFromState(game.getCurrentState()).size(
)==0){//checks whether the user can make any more moves
            killThread=true;
            timer.stop();
            if(game.getCurrentState().getIsDraw()){//sets game to a draw
and exits
                JOptionPane.showMessageDialog(this, "The game was a
draw!","Draw!",JOptionPane.OK_OPTION);
                gameData[11]="0.5";
                if(isUserPlayerOne){
```

```
                    gameData[8]="true";
                }else{
                    gameData[9]="true";
                }
                saveRecordToServer();
                OnlineMenu menu = new OnlineMenu(session);
                menu.setVisible(true);
                this.dispose();
                return;
            }
            //checks to see which colour wins the game and updates the
game record accordingly. The system then returns to a menu.
            if(ChessGame.isInCheck(!game.getCurrentState().IS_BLACK_TURN
_TO_PLAY_NEXT, new ChessBoardState(game.getCurrentState().BOARD,
game.getCurrentState().CASTLING_RIGHTS,
!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT,
game.getCurrentState().FULL_MOVE_NUMBER,
game.getCurrentState().HALF_MOVE_CLOCK,
game.getCurrentState().SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT))){
                killThread=true;
                if(game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
                    JOptionPane.showMessageDialog(this, "White
wins!","Checkmate!",JOptionPane.OK_OPTION);
                    if(isUserPlayerOne && isUserWhite){
                        gameData[11]="1";
                    }else{
                        gameData[11]="0";
                    }
                     if(isUserPlayerOne){
                         gameData[8]="true";

                     }else{
                         gameData[9]="true";
                     }
                     saveRecordToServer();
                     OnlineMenu menu = new OnlineMenu(session);
                     menu.setVisible(true);
                     this.dispose();
                    return;
                }else{
                    killThread=true;
                    JOptionPane.showMessageDialog(this, "Black
wins.","Checkmate!",JOptionPane.OK_OPTION);
                    if(isUserPlayerOne && !isUserWhite){
                        gameData[11]="1";
                    }else{
                        gameData[11]="0";
                    }
                     if(isUserPlayerOne){
                         gameData[8]="true";

                     }else{
                         gameData[9]="true";
```

```
                    }
                    saveRecordToServer();
                    OnlineMenu menu = new OnlineMenu(session);
                    menu.setVisible(true);
                    this.dispose();
                   return;
                }
            }
            JOptionPane.showMessageDialog(this, "The game was a
draw!","Draw!",JOptionPane.OK_OPTION);
                gameData[11]="0.5";
                if(isUserPlayerOne){
                    gameData[8]="true";
                }else{
                    gameData[9]="true";
                }
                saveRecordToServer();
                OnlineMenu menu = new OnlineMenu(session);
                menu.setVisible(true);
                this.dispose();
            return;
        }
        //checks to see if opponent has run out of time
        if(accountForPlayerTime){
            if(Integer.parseInt(gameData[5])<=0){
                JOptionPane.showMessageDialog(this,"White has run out of
time. Black wins!","Black wins!",JOptionPane.OK_OPTION);
                if(isUserPlayerOne){
                    gameData[8]="true";
                }else{
                    gameData[9]="true";
                }
                if(gameData[10].equals("true")){
                    gameData[11]="0";
                }else{
                    gameData[11]="1";
                }
                saveRecordToServer();
                OnlineMenu menu = new OnlineMenu(session);
                menu.setVisible(true);
                this.dispose();  return;
            }else if(Integer.parseInt(gameData[6])<=0){
                JOptionPane.showMessageDialog(this,"Black has run out of
time. White wins!","White wins!",JOptionPane.OK_OPTION);
                if(isUserPlayerOne){
                    gameData[8]="true";
                }else{
                    gameData[9]="true";
                }
                if(gameData[10].equals("true")){
                    gameData[11]="1";
                }else{
                    gameData[11]="0";
```

```
                }
                saveRecordToServer();
                OnlineMenu menu = new OnlineMenu(session);
                menu.setVisible(true);
                this.dispose();  return;
            }
        }
    }
    /**
     * Plays a sound to indicate that a move has been made if the game
is not muted
     */
    public void playMoveMadeSound(){
        if(soundIsOn){
            Random rnd = new Random();
            Audio.playSound("Assets"+File.separator+String.valueOf(rnd.n
extInt(8)+1)+"Sound.wav");
        }
    }
    /**
     * Handles when a square is clicked. This includes highlighting the
square and all available moves if appropriate.
     * @param y The y index of the square
     * @param x The x index of the square
     */
    private void squareClicked(int y,int x){
        //if opponent is a computer and its the computer's turn to play,
no squares should be highlighted
        if((isUserWhite&&game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NE
XT)||(!isUserWhite&&!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT))
{
            return;
        }
        if(currentIndex!=game.getLengthOfStatesFoFar()-1){//if the board
is not the current board, no action is needed
            return;
        }
        int yBoard=y;int xBoard=x;//stores the y and x coords regarding
to the actual board array stored
        if(isWhiteAtBottom==false){//accounts for a flipped board
            yBoard=7-y;xBoard=7-x;
        }
        //updateBoard();
        //checks to see if square clicked is a legal square to move to.
If no square has been clicked, the array is of length 0
        for(int move=0;move<legalSquaresToMoveTo.size();move=move+2){
            if(legalSquaresToMoveTo.get(move)==yBoard&&legalSquaresToMov
eTo.get(move+1)==xBoard){//if square choosen is a move, it updates game
object and board
                ArrayList<ChessBoardState>states=ChessGame.getAllLegalBo
ardsFromMove(yCoordSquareClicked, xCoordSquareClicked, yBoard, xBoard,
game.getCurrentState());
                if(states.size()==1){
```

```
                    game.addState(states.get(0));
                }else{//used to allow user to make choice about pawn
promotion
                    Object[] options =
{"Queen","Rook","Bishop","Knight"};//stores the possible pieces to
promote
                    int choice=JOptionPane.showOptionDialog(this,"Please
select piece to promote pawn
to.","Piece",JOptionPane.YES_NO_CANCEL_OPTION,JOptionPane.QUESTION_MESSA
GE,null,options,options[0]);
                    if(choice==-1){//if no option is seleted, the queen
is chosen
                        game.addState(states.get(0));
                    }else{
                        game.addState(states.get(choice));
                    }
                }
                //updates the board following the moving having been
                currentIndex++;
                gameData[3]=gameData[3]+OfflinePlay.SPLIT_FEN_STRING+gam
e.getCurrentState().getFenString();
                saveRecordToServer();
                updateBoard();
                updateFormBasedOnGameDataRecord();
                playMoveMadeSound();
                checkForTerminalState();
                return;
            }
        }
        //if invalid square is selected, nothing occurrs
        if((game.getCurrentState().BOARD[yBoard][xBoard]>=0&&game.getCur
rentState().IS_BLACK_TURN_TO_PLAY_NEXT)||(game.getCurrentState().BOARD[y
Board][xBoard]<=0&&!game.getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT)){
            return;
        }
        if(yCoordSquareClicked==yBoard&&xCoordSquareClicked==xBoard){//i
f selected square is being clicked again, it will be deselected
            updateBoard();
            return;
        }
        updateBoard();
        //gets all legal moves from square pressed and highlights square
pressed as well as any legal moves
        yCoordSquareClicked=yBoard;xCoordSquareClicked=xBoard;
        boardVisual[y][x].setBackground(Color.orange);
        legalSquaresToMoveTo=ChessGame.getAllLegalMovesFromSquare(yBoard
, xBoard, game.getCurrentState());
        for(int
move=0;move<legalSquaresToMoveTo.size();move=move+2){//goes through each
legal move and highlights square
            if(isWhiteAtBottom){//accounts for flipped board
                boardVisual[legalSquaresToMoveTo.get(move)][legalSquares
ToMoveTo.get(move+1)].setBackground(Color.blue);
```

```
            }else{
                boardVisual[7-legalSquaresToMoveTo.get(move)][7-
legalSquaresToMoveTo.get(move+1)].setBackground(Color.blue);
            }
        }
    }
    /**
     * Adds listeners to each square on the board. This code is very
long and has been generated by a computer. It has been collapsed as it
takes up a large amount of space
     */
    private void addSquareListeners(){
        //<editor-fold defaultstate="collapsed" desc="Computer generated
action listeners for each square of the board">
        boardVisual[0][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,0);}});
        boardVisual[0][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,1);}});
        boardVisual[0][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,2);}});
        boardVisual[0][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,3);}});
        boardVisual[0][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,4);}});
        boardVisual[0][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,5);}});
        boardVisual[0][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,6);}});
        boardVisual[0][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(0,7);}});
        boardVisual[1][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,0);}});
        boardVisual[1][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,1);}});
        boardVisual[1][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,2);}});
        boardVisual[1][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,3);}});
        boardVisual[1][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,4);}});
```

```
        boardVisual[1][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,5);}});
        boardVisual[1][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,6);}});
        boardVisual[1][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(1,7);}});
        boardVisual[2][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,0);}});
        boardVisual[2][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,1);}});
        boardVisual[2][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,2);}});
        boardVisual[2][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,3);}});
        boardVisual[2][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,4);}});
        boardVisual[2][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,5);}});
        boardVisual[2][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,6);}});
        boardVisual[2][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(2,7);}});
        boardVisual[3][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,0);}});
        boardVisual[3][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,1);}});
        boardVisual[3][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,2);}});
        boardVisual[3][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,3);}});
        boardVisual[3][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,4);}});
        boardVisual[3][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,5);}});
        boardVisual[3][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
```

```
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,6);}});
        boardVisual[3][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(3,7);}});
        boardVisual[4][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,0);}});
        boardVisual[4][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,1);}});
        boardVisual[4][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,2);}});
        boardVisual[4][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,3);}});
        boardVisual[4][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,4);}});
        boardVisual[4][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,5);}});
        boardVisual[4][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,6);}});
        boardVisual[4][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(4,7);}});
        boardVisual[5][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,0);}});
        boardVisual[5][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,1);}});
        boardVisual[5][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,2);}});
        boardVisual[5][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,3);}});
        boardVisual[5][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,4);}});
        boardVisual[5][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,5);}});
        boardVisual[5][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,6);}});
        boardVisual[5][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(5,7);}});
        boardVisual[6][0].addMouseListener(new
```

```
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,0);}});
        boardVisual[6][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,1);}});
        boardVisual[6][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,2);}});
        boardVisual[6][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,3);}});
        boardVisual[6][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,4);}});
        boardVisual[6][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,5);}});
        boardVisual[6][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,6);}});
        boardVisual[6][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(6,7);}});
        boardVisual[7][0].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,0);}});
        boardVisual[7][1].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,1);}});
        boardVisual[7][2].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,2);}});
        boardVisual[7][3].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,3);}});
        boardVisual[7][4].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,4);}});
        boardVisual[7][5].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,5);}});
        boardVisual[7][6].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent evt){squareClicked(7,6);}});
        boardVisual[7][7].addMouseListener(new
java.awt.event.MouseAdapter(){public void
mouseClicked(java.awt.event.MouseEvent
evt){squareClicked(7,7);}});
        //</editor-fold>
    }
    /**
     * @param args the command line arguments
     */
```

```
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ChessBoardOnline.class.ge
tName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ChessBoardOnline.class.ge
tName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ChessBoardOnline.class.ge
tName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ChessBoardOnline.class.ge
tName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ChessBoardOnline().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JLabel blackEloLbl;
    private javax.swing.JLabel blackLbl;
    private javax.swing.JLabel blackTimeLbl;
    private javax.swing.JLabel blackTimeLeftLbl;
    private javax.swing.JButton changeBoardBtn;
    private javax.swing.JButton flipBoardBtn;
    private javax.swing.JLabel gameTitleLbl;
    private javax.swing.JButton invisibleDrawBtn;
    private javax.swing.JLabel isBlackTurnLbl;
```

```
    private javax.swing.JLabel isWhiteTurnLbl;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JLabel moveNumberLbl;
    private javax.swing.JButton nextBoardBtn;
    private javax.swing.JButton offerDrawBtn;
    private javax.swing.JButton refreshBtn;
    private javax.swing.JButton resignBtn;
    private javax.swing.JLabel speakerLbl;
    private javax.swing.JLabel timerLbl;
    private javax.swing.JLabel timerLbl1;
    private javax.swing.JButton viewPresentBoard;
    private javax.swing.JButton viewPreviousBoard;
    private javax.swing.JLabel whiteEloLbl;
    private javax.swing.JLabel whiteLbl;
    private javax.swing.JLabel whiteTiimeLbl;
    private javax.swing.JLabel whiteTimeLeftLbl;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

/**
 * This class is used to store all data needed for a single board state
 * @author mortimer
 */
public class ChessBoardState {
    public final int BOARD[][]= new int[8][8];//stores chess
board  white always starts at bottom
    public boolean IS_BLACK_TURN_TO_PLAY_NEXT;//stores which colour is
to move next
    public final boolean[] CASTLING_RIGHTS=new boolean[4];//stores the
castling rights. Stored in format: White kingside, white queenside,
black kingside, black queenside
    public final int FULL_MOVE_NUMBER;//the numbe of moves black has
made. This value starts at one
    public final int HALF_MOVE_CLOCK;//the number of moves made since a
piece was capture or a pawn advanced a square. This is used to enforce
the fifty move rule. This value starts at 0
    public final  int[]SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT;//stores
sqaure that can be captured via en passant. Has a length of 0 if no such
squares are available
    private  String fenString;//stores fen string of board state
    private boolean isDraw;//stores whether the state is a draw. Note
this must be set externally
```

```
    private boolean isWin;//stores whether state is a win for player
currently moving. Note this must be set externally
    private boolean isLoss;//stores whether state is a loss for player
currently moving. Note this must be set externally
    private boolean hasFenBeenCreated=false;
    public static final int WHITE_MULTIPLIER=1;public static final int
BLACK_MULTIPLIER=-1;
    public static final int ROOK_VALUE=5;public static final int
BISHOP_VALUE =4;public static final int KNIGHT_VALUE=3;
    public static final int PAWN_VALUE=1;
    public static final int QUEEN_VALUE=8;public static final int
KING_VALUE=100; public static final int EMPTY_VALUE=0;
    /**
     * Takes the FEN String (standard representation of chess board as
input)
     * @param fenString The FEN string. For explanation of FEN format,
view https://www.chess.com/terms/fen-chess
     */
    public ChessBoardState (String fenString){
        this.fenString=fenString;hasFenBeenCreated=true;
        String[]fenSplit = fenString.split(" ");//splits the FEN String
into its fields
        //generates board based on FEN String
        String board[]=fenSplit[0].split("/");//splits the board data
into ranks
        int xIndex;//stores current x index of piece
        //converts fen representation of board to an 8 by 8 two
dimensional array which represents the board
        for(int y=0;y<8;y++){//generates board data for each rank
            xIndex=0;
            for(int ch=0;ch<board[y].length();ch++){
                switch (board[y].charAt(ch)) {
                    case 'r':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*ROOK_VALUE;
                        xIndex++;
                        break;
                    case 'n':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*KNIGHT_VALUE;
                        xIndex++;
                        break;
                    case 'b':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*BISHOP_VALUE;
                        xIndex++;
                        break;
                    case 'q':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*QUEEN_VALUE;
                        xIndex++;
                        break;
                    case 'k':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*KING_VALUE;
                        xIndex++;
                        break;
                    case 'p':
```

```
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*PAWN_VALUE;
                        xIndex++;
                        break;
                    case 'N':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*KNIGHT_VALUE;
                        xIndex++;
                        break;
                    case 'B':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*BISHOP_VALUE;
                        xIndex++;
                        break;
                    case 'Q':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*QUEEN_VALUE;
                        xIndex++;
                        break;
                    case 'K':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*KING_VALUE;
                        xIndex++;
                        break;
                    case 'R':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*ROOK_VALUE;
                        xIndex++;
                        break;
                    case 'P':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*PAWN_VALUE;
                        xIndex++;
                        break;
                    default:
                        for(int
emp=0;emp<Character.getNumericValue(board[y].charAt(ch));emp++){
                            BOARD[y][xIndex]=EMPTY_VALUE;
                            xIndex++;
                        }   break;
                }
            }
        }
        IS_BLACK_TURN_TO_PLAY_NEXT=fenSplit[1].equals("b");//gets who's
move it is
        //gets castling rights
        //stored in format White kingside, white queenside, black
kingside, black queenside
        CASTLING_RIGHTS[0]=false;CASTLING_RIGHTS[1]=false;CASTLING_RIGHT
S[2]=false;CASTLING_RIGHTS[3]=false;
        if(!fenSplit[2].equals("-")){
            for(int ch =0;ch<fenSplit[2].length();ch++){
                switch (fenSplit[2].charAt(ch)) {
                    case 'K':
                        CASTLING_RIGHTS[0]=true;
                        break;
                    case 'Q':
                        CASTLING_RIGHTS[1]=true;
                        break;
                    case 'k':
```

```
                        CASTLING_RIGHTS[2]=true;
                        break;
                    default:
                        CASTLING_RIGHTS[3]=true;
                        break;
                }
            }
        }
        //gets possible en passant targets
        if(fenSplit[3].equals("-")){
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT=new int[0];//no
available squares
        }else{
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT =
getYAndXOfSquare(fenSplit[3]);//one available en passant square
        }
        //gets half move clock
        HALF_MOVE_CLOCK=Integer.parseInt(fenSplit[4]);
        //gets full move number
        FULL_MOVE_NUMBER=Integer.parseInt(fenSplit[5]);
    }
    /**
     * Creates a new chess board state
     * @param gameBoard The board array
     * @param castlingRights The castling rights. In the form, white
kingside, white queen side, black kingside, black queen side
     * @param isBlackToPlayNext whether black is to make the next move
     * @param fullMoveNumber the number of moves black has played
     * @param halfMoveClock the number of moves both players have made
since the last pawn advance or capture
     * @param enPassantSquare the y and x coordinates of a square that
can be captured via en passant. If no such square exists, use new int[0]
     */
    public ChessBoardState(int[][]gameBoard,boolean
castlingRights[],boolean isBlackToPlayNext, int fullMoveNumber,int
halfMoveClock,int[]enPassantSquare){
        //stores board
        for(int y=0;y<8;y++){//generates board
            for(int x =0;x<8;x++){
                BOARD[y][x]=gameBoard[y][x];
            }
        }
        //creates castling rights
        CASTLING_RIGHTS[0]=castlingRights[0];CASTLING_RIGHTS[1]=castling
Rights[1];CASTLING_RIGHTS[2]=castlingRights[2];CASTLING_RIGHTS[3]=castli
ngRights[3];
        IS_BLACK_TURN_TO_PLAY_NEXT=isBlackToPlayNext;
        FULL_MOVE_NUMBER=fullMoveNumber;
        HALF_MOVE_CLOCK=halfMoveClock;
        //creates en passant data
        if(enPassantSquare.length==0){
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT= new int[0];
        }else{
```

```
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT=new int [2];
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[0]=enPassantSquare[0]
;
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]=enPassantSquare[1]
;
        }
    }
    /**
     * Converts the current board state to a form where it can be input
to the neural network.
     * @return The double array that can be input to the trained neural
network.
     */
    public double[]getNeuralNetInputDouble(){
        double data[]= new double [777];//creates the array that stores
the neural network attributes
        //stores castling rights
        if(CASTLING_RIGHTS[0]){
            data[768]=1;
        }
        if(CASTLING_RIGHTS[1]){
            data[769]=1;
        }
        if(CASTLING_RIGHTS[2]){
            data[770]=1;
        }
        if(CASTLING_RIGHTS[3]){
            data[771]=1;
        }
        //stores who is going to play next
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            data[772]=1;
        }
        //sets half move data
        data[773]=HALF_MOVE_CLOCK;
        //sets en passant data
        data[775]=9;data[776]=9;
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            data[774]=1;data[775]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT
[0];data[776]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1];
        }
        //generates bitboard data
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                switch (BOARD[y][x]) {
                    case WHITE_MULTIPLIER*KING_VALUE:
                        data[x+(8*y)]=1;
                        break;
                    case BLACK_MULTIPLIER*KING_VALUE:
                        data[x+(8*y)+64]=1;
                        break;
                    case WHITE_MULTIPLIER*ROOK_VALUE:
                        data[x+(8*y)+128]=1;
```

```
                        break;
                    case BLACK_MULTIPLIER*ROOK_VALUE:
                        data[x+(8*y)+192]=1;
                        break;
                    case WHITE_MULTIPLIER*BISHOP_VALUE:
                        data[x+(8*y)+256]=1;
                        break;
                    case BLACK_MULTIPLIER*BISHOP_VALUE:
                        data[x+(8*y)+320]=1;
                        break;
                    case WHITE_MULTIPLIER*QUEEN_VALUE:
                        data[x+(8*y)+384]=1;
                        break;
                    case BLACK_MULTIPLIER*QUEEN_VALUE:
                        data[x+(8*y)+448]=1;
                        break;
                    case WHITE_MULTIPLIER*PAWN_VALUE:
                        data[x+(8*y)+512]=1;
                        break;
                    case BLACK_MULTIPLIER*PAWN_VALUE:
                        data[x+(8*y)+576]=1;
                        break;
                    case WHITE_MULTIPLIER*KNIGHT_VALUE:
                        data[x+(8*y)+640]=1;
                        break;
                    case BLACK_MULTIPLIER*KNIGHT_VALUE:
                        data[x+(8*y)+704]=1;
                        break;
                    default:
                        break;
                }
            }
        }
        return data;
    }
    /**
     * Gets the string used for the neural network input given the
current board state
     * @return The input to the neural network
     */
    public int[] getNeuralNetInput(){
        int data[]= new int [777];//creates the array that stores the
neural network attributes
        //stores castling rights
        if(CASTLING_RIGHTS[0]){
            data[768]=1;
        }
        if(CASTLING_RIGHTS[1]){
            data[769]=1;
        }
        if(CASTLING_RIGHTS[2]){
            data[770]=1;
        }
```

```
        if(CASTLING_RIGHTS[3]){
            data[771]=1;
        }
        //stores who is going to play next
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            data[772]=1;
        }
        //sets half move data
        data[773]=HALF_MOVE_CLOCK;
        //sets en passant data
        data[775]=9;data[776]=9;
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            data[774]=1;data[775]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT
[0];data[776]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1];
        }
        //generates bitboard data
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                switch (BOARD[y][x]) {
                    case WHITE_MULTIPLIER*KING_VALUE:
                        data[x+(8*y)]=1;
                        break;
                    case BLACK_MULTIPLIER*KING_VALUE:
                        data[x+(8*y)+64]=1;
                        break;
                    case WHITE_MULTIPLIER*ROOK_VALUE:
                        data[x+(8*y)+128]=1;
                        break;
                    case BLACK_MULTIPLIER*ROOK_VALUE:
                        data[x+(8*y)+192]=1;
                        break;
                    case WHITE_MULTIPLIER*BISHOP_VALUE:
                        data[x+(8*y)+256]=1;
                        break;
                    case BLACK_MULTIPLIER*BISHOP_VALUE:
                        data[x+(8*y)+320]=1;
                        break;
                    case WHITE_MULTIPLIER*QUEEN_VALUE:
                        data[x+(8*y)+384]=1;
                        break;
                    case BLACK_MULTIPLIER*QUEEN_VALUE:
                        data[x+(8*y)+448]=1;
                        break;
                    case WHITE_MULTIPLIER*PAWN_VALUE:
                        data[x+(8*y)+512]=1;
                        break;
                    case BLACK_MULTIPLIER*PAWN_VALUE:
                        data[x+(8*y)+576]=1;
                        break;
                    case WHITE_MULTIPLIER*KNIGHT_VALUE:
                        data[x+(8*y)+640]=1;
                        break;
                    case BLACK_MULTIPLIER*KNIGHT_VALUE:
```

```
                        data[x+(8*y)+704]=1;
                        break;
                    default:
                        break;
                }
            }
        }
        return data;
    }
    /**
     * Gets FEN string from board state
     * @return The FEN String
     */
    public String getFenString(){
        if(this.hasFenBeenCreated){//if a fen value has already been
created then it is simply returned
            return this.fenString;
        }
        //creates a fen string from scratch
        String fenStringBoard="";//stores board as FEN string
        int noOfEmptySquares;//stores the number of empty squares next
to each other in a row
        for(int y=0;y<8;y++){//generates board and FEN String for the
board
            if(y!=0){
                fenStringBoard=fenStringBoard+"/";//splits the ranks
            }
            noOfEmptySquares=0;
            //for each square in the rank, the correct fen symbol is
calculated
            for(int x =0;x<8;x++){
                switch (BOARD[y][x]) {
                    case EMPTY_VALUE:
                        noOfEmptySquares++;
                        if(x==7){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                        }   break;
                    case BLACK_MULTIPLIER*ROOK_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"r";
                        break;
                    case BLACK_MULTIPLIER*KNIGHT_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"n";
                        break;
                    case BLACK_MULTIPLIER*BISHOP_VALUE:
```

```
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"b";
                        break;
                    case BLACK_MULTIPLIER*QUEEN_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"q";
                        break;
                    case BLACK_MULTIPLIER*KING_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"k";
                        break;
                    case BLACK_MULTIPLIER*PAWN_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"p";
                        break;
                    case WHITE_MULTIPLIER*ROOK_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"R";
                        break;
                    case WHITE_MULTIPLIER*KNIGHT_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"N";
                        break;
                    case WHITE_MULTIPLIER*BISHOP_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"B";
                        break;
                    case WHITE_MULTIPLIER*QUEEN_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
```

```
                        }   fenStringBoard=fenStringBoard+"Q";
                        break;
                    case WHITE_MULTIPLIER*KING_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"K";
                        break;
                    case WHITE_MULTIPLIER*PAWN_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"P";
                        break;
                    default:
                        break;
                }
            }
        }
        //creates the castling rights
        String fenCastlingRights="";
        if(CASTLING_RIGHTS[0]){
            fenCastlingRights="K";
        }
        if(CASTLING_RIGHTS[1]){
            fenCastlingRights=fenCastlingRights+"Q";
        }
        if(CASTLING_RIGHTS[2]){
            fenCastlingRights=fenCastlingRights+"k";
        }
        if(CASTLING_RIGHTS[3]){
            fenCastlingRights=fenCastlingRights+"q";
        }
        if(fenCastlingRights.equals("")){
            fenCastlingRights="-";
        }
        String fenColourToPlay="";
        //sets who's turn it is to play next
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            fenColourToPlay="b";
        }else{
            fenColourToPlay="w";
        }
        //stores the square (if any) on which en passant can be
performed on
        String enPassantSquareFen="-";
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            enPassantSquareFen=getStringOfSquare(SQUARE_THAT_CAN_BE_CAPT
URED_EN_PASSANT[0], SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]);
        }
        //creates the final fen string
```

```
        this.fenString=fenStringBoard+" "+fenColourToPlay+"
"+fenCastlingRights+" "+enPassantSquareFen+"
"+String.valueOf(HALF_MOVE_CLOCK)+" "+String.valueOf(FULL_MOVE_NUMBER);
        this.hasFenBeenCreated=true;
        return this.fenString;
    }
    /**
     * Gets the string value of a square given its y and x coordinates.
E.g. getStringOfSquare(1,0) = "a7"
     * @param y The y index in the board array
     * @param x The x index in the board array
     * @return The square
     */
    private static String getStringOfSquare(int y, int x){
        switch (x) {
            case 0:
                return "a"+String.valueOf(8-y);
            case 1:
                return "b"+String.valueOf(8-y);
            case 2:
                return "c"+String.valueOf(8-y);
            case 3:
                return "d"+String.valueOf(8-y);
            case 4:
                return "e"+String.valueOf(8-y);
            case 5:
                return "f"+String.valueOf(8-y);
            case 6:
                return "g"+String.valueOf(8-y);
            default:
                return "h"+String.valueOf(8-y);
        }
    }
    /**
     * Gets the neural network input for a reduced size neural network
that does not use the bitboard representation.
     * @return The network input
     */
    public int[]getNeuralNetInputReducedSize(){
        int data[]=new int[64+4+2+1];
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                data[8*y +x]=BOARD[y][x];
            }
        }
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            data[64]=1;
        }
        if(CASTLING_RIGHTS[0]){
            data[65]=1;
        }
        if(CASTLING_RIGHTS[1]){
            data[66]=1;
```

```
        }
        if(CASTLING_RIGHTS[2]){
            data[67]=1;
        }
        if(CASTLING_RIGHTS[3]){
            data[68]=1;
        }
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            data[69]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[0];
            data[70]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1];
        }
        return data;
    }
    /**
     * Gets a reduced size neural network input as a double array.
     * @return The neural network input
     */
    public double[]getNeuralNetInputReducedSizeDouble(){
        double data[]=new double[71];
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                data[8*y +x]=BOARD[y][x];
            }
        }
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            data[64]=1;
        }
        if(CASTLING_RIGHTS[0]){
            data[65]=1;
        }
        if(CASTLING_RIGHTS[1]){
            data[66]=1;
        }
        if(CASTLING_RIGHTS[2]){
            data[67]=1;
        }
        if(CASTLING_RIGHTS[3]){
            data[68]=1;
        }
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            data[69]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[0];
            data[70]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1];
        }
        return data;
    }
    /**
     * Takes a chess square, such as "a3" as input and returns the y and
x coordinates of the square on the board
     * @param square The square
     * @return An int array of length [2], with the y and x index for
the board array
     */
    private static int[] getYAndXOfSquare(String square){
```

```
        int[]ret = new int[2];
        ret[0]=8-Character.getNumericValue(square.charAt(1));
        switch (square.charAt(0)) {
            case 'a':
                ret[1]=0;
                break;
            case 'b':
                ret[1]=1;
                break;
            case 'c':
                ret[1]=2;
                break;
            case 'd':
                ret[1]=3;
                break;
            case 'e':
                ret[1]=4;
                break;
            case 'f':
                ret[1]=5;
                break;
            case 'g':
                ret[1]=6;
                break;
            default:
                ret[1]=7;
                break;
        }
        return ret;
    }
    /**
     * Gets a copy of the board array. use this when writing with board
     * @return A copy of the board array
     */
    public int[][]getCopyOfBoard(){
        int[][]board=new int[8][8];
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                board[y][x]=BOARD[y][x];
            }
        }
        return board;
    }
    /**
     * Returns a copy of the castling rights array
     * @return The castling rights
     */
    public boolean []getCopyOfCastlingRights(){
        boolean rights []=new boolean[4];
        for(int i=0;i<4;i++){
            rights[i]=CASTLING_RIGHTS[i];
        }
        return rights;
```

```
    }
    /**
     * Gets a copy of the en passant square array and returns it.
     * @return The copy of the array
     */
    public int[]getCopyOfEnPassantSquare(){
        int ret[]=new
int[SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length];
        for(int i=0;i<ret.length;i++){
            ret[i]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[i];
        }
        return ret;
    }
    /**
     * Returns whether this game is a win. Note this variable may not be
initialised and can only be set by external methods. Hence, only
     * call this method if you are certain that the board state has been
set.
     * @return Whether the board is a win for the current user
     */
    public boolean getIsWin(){
        return this.isWin;
    }
    /**
     * Returns whether this game is a draw. Note: this variable is not
automatically initialised and is set externally.
     * @return Whether the game is a draw
     */
    public boolean getIsDraw(){
        return this.isDraw;
    }
    /**
     * Returns whether this game is a loss for the user to play
currently. Note this variable is not automatically initialised, so only
call this method if the variable has been initialised.
     * @return Whether the game is a loss
     */
    public boolean getIsLoss(){
        return this.isLoss;
    }
    /**
     * Sets the game as win
     */
    public void setIsWinTrue(){
        this.isWin=true;this.isLoss=false;this.isDraw=false;
    }
    /**
     * Sets the game as a draw
     */
    public void setIsDrawTrue(){
        this.isDraw=true;this.isLoss=false;this.isWin=false;
    }
    /**
```

```
     * Sets the game as a loss
     */
    public void setIsLossTrue(){
        this.isLoss=true;this.isDraw=false;this.isWin=false;
    }
    /**
     * Sets the FEN string if it has already been created
     * @param fen The FEN
     */
    private void setFen(String fen){
        this.fenString=fen;
    }
    /**
     * Sets whether the FEN has been created
     * @param hasFenBeenCreated Has the FEN been created
     */
    private void setHasFenBeenCreated(boolean hasFenBeenCreated){
        this.hasFenBeenCreated=hasFenBeenCreated;
    }
    /**
     * Sets whether the game is a loss
     * @param isLoss Whether the game is a loss
     */
    private void setIsLoss(boolean isLoss){
        this.isLoss=isLoss;
    }
    /**
     * Sets whether the game is a draw
     * @param isDraw Whether the game is a draw
     */
    private void setIsDraw(boolean isDraw){
        this.isDraw=isDraw;
    }
    /**
     * Sets whether the game is a win
     * @param isWin Whether the game is a win
     */
    private void setIsWin(boolean isWin){
        this.isWin=isWin;
    }
    /**
     * Gets a copy of the current chess board state.
     * @return The copy of the current board state
     */
    public ChessBoardState getCopyOfState(){
        int[][]board=new int[8][8];//creates a copy of the board
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                board[y][x]=BOARD[y][x];
            }
        }
        //copies all values from current state and use them to create a
new board state object
```

```
        int[]enPassant = new
int[SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length];
        for(int
i=0;i<SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length;i++){
            enPassant[i]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[i];
        }
        boolean castling[]=new boolean[CASTLING_RIGHTS.length];
        for(int i=0;i<CASTLING_RIGHTS.length;i++){
            castling[i]=CASTLING_RIGHTS[i];
        }
        //creates a new object copying the board
        ChessBoardState newState = new ChessBoardState(board, castling,
IS_BLACK_TURN_TO_PLAY_NEXT, FULL_MOVE_NUMBER, HALF_MOVE_CLOCK,
enPassant);
        newState.setHasFenBeenCreated(hasFenBeenCreated);
        newState.setFen(fenString);
        newState.setIsWin(isWin);
        newState.setIsLoss(isLoss);
        newState.setIsDraw(isDraw);
        return newState;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;
import javax.swing.JOptionPane;

/**
 * This class stores the chess game (i.e. all board states). It also
handles getting any potential computer moves.
 * @author mortimer
 */
public class ChessGame {
    private ChessBoardState [] states;//stores all board states so far
    private static final int BLACK_MULT=-1;final int WHITE_MULT=1;
    //these values are used for level 2 evalaution
    private static final int VALUE_OF_PAWN=1;//stores the relative value
```

```
of the pawn
    private static final int VALUE_OF_ROOK=5;//stores relative value of
rook
    private static final int VALUE_OF_BISHOP=3;//stores relative value
of bishop
    private static final int VALUE_OF_KNIGHT=3;//stores relative value
of knight
    private static final int VALUE_OF_QUEEN=9;//stores relative value of
queen
    private static final int VALUE_OF_KING=600000;//stores the value of
the king - which is an arbitarily high value
    private static final int DRAW_VALUE=0;//stores the value assigned to
a draw
    private static final int WHITE_WIN_VALUE=(Integer.MAX_VALUE-1)-1;
    private static final int BLACK_WIN_VALUE=(Integer.MAX_VALUE*-1)+1;
    //stores the relative positional value for pieces
    private static final
int[][]PAWN_POSITIONAL_WHITE={{0,0,0,0,0,0,0,0},{50,50,50,50,50,50,50,50
},{10,10,20,30,30,20,10,10},{5,5,10,25,25,10,5,5},{0,0,0,20,20,0,0,0},{5
,-5,-10,0,0,-10,-5,5},{5,10,10,-20,-20,10,10,5},{0,0,0,0,0,0,0,0}};
    private static final
int[][]PAWN_POSITIONAL_BLACK={{0,0,0,0,0,0,0,0},{-5,-10,-10,20,20,-10,-
10,-5},{-5,5,10,0,0,10,5,-5},{0,0,0,-20,-20,0,0,0},{-5,-5,-10,-25,-25,-
10,-5,-5},{-10,-10,-20,-30,-30,-20,-10,-10},{-50,-50,-50,-50,-50,-50,-
50,-50},{0,0,0,0,0,0,0,0}};
    private static final int[][]KNIGHT_POSITIONAL_WHITE={{-50,-40,-30,-
30,-30,-30,-40,-50},{-40,-20,0,0,0,0,-20,-40},{-30,0,10,15,15,10,0,-
30},{-30,5,15,20,20,15,5,-30},{-30,0,15,20,20,15,0,-30},{-
30,5,10,15,15,10,5,-30},{-40,-20,0,5,5,0,-20,-40},{-50,-40,-30,-30,-30,-
30,-40,-50}};
    private static final
int[][]KNIGHT_POSITIONAL_BLACK={{50,40,30,30,30,30,40,50},{40,20,0,-5,-
5,0,20,40},{30,-5,-10,-15,-15,-10,-5,30},{30,0,-15,-20,-20,-
15,0,30},{30,-5,-15,-20,-20,-15,-5,30},{30,0,-10,-15,-15,-
10,0,30},{40,20,0,0,0,0,20,40},{50,40,30,30,30,30,40,50}};
    private static final int[][]QUEEN_POSITIONAL_WHITE={{-20,-10,-10,-
5,-5,-10,-10,-20},{-10,0,0,0,0,0,0,-10},{-10,0,5,5,5,5,0,-10},{-
5,0,5,5,5,5,0,-5},{0,0,5,5,5,5,0,-5},{-10,5,5,5,5,5,0,-10},{-
10,0,5,0,0,0,0,-10},{-20,-10,-10,-5,-5,-10,-10,-20}};
    private static final
int[][]QUEEN_POSITIONAL_BLACK={{20,10,10,5,5,10,10,20},{10,0,0,0,0,-
5,0,10},{10,0,-5,-5,-5,-5,-5,10},{5,0,-5,-5,-5,-5,0,0},{5,0,-5,-5,-5,-
5,0,5},{10,0,-5,-5,-5,-
5,0,10},{10,0,0,0,0,0,0,10},{20,10,10,5,5,10,10,20}};
    private static final int[][]BISHOP_POSITIONAL_WHITE={{-20,-10,-10,-
10,-10,-10,-10,-20},{-10,0,0,0,0,0,0,-10},{-10,0,5,10,10,5,0,-10},{-
10,5,5,10,10,5,5,-10},{-10,0,10,10,10,10,0,-10},{-10,10,10,10,10,10,10,-
10},{-10,5,0,0,0,0,5,-10},{-20,-10,-10,-10,-10,-10,-10,-20}};
    private static final
int[][]BISHOP_POSITIONAL_BLACK={{20,10,10,10,10,10,10,20},{10,-
5,0,0,0,0,-5,10},{10,-10,-10,-10,-10,-10,-10,10},{10,0,-10,-10,-10,-
10,0,10},{10,-5,-5,-10,-10,-5,-5,10},{10,0,-5,-10,-10,-
5,0,10},{10,0,0,0,0,0,0,10},{20,10,10,10,10,10,10,20}};
```

```
    private static final
int[][]ROOK_POSITIONAL_WHITE={{0,0,0,0,0,0,0,0},{5,10,10,10,10,10,10,5},
{-5,0,0,0,0,0,0,-5},{-5,0,0,0,0,0,0,-5},{-5,0,0,0,0,0,0,-5},{-
5,0,0,0,0,0,0,-5},{-5,0,0,0,0,0,0,-5},{0,0,0,5,5,0,0,0}};
    private static final int[][]ROOK_POSITIONAL_BLACK={{0,0,0,-5,-
5,0,0,0},{5,0,0,0,0,0,0,5},{5,0,0,0,0,0,0,5},{5,0,0,0,0,0,0,5},{5,0,0,0,
0,0,0,5},{5,0,0,0,0,0,0,5},{-5,-10,-10,-10,-10,-10,-10,-
5},{0,0,0,0,0,0,0,0}};
    private static final int[][]KING_POSITIONAL_WHITE_MID={{-30,-40,-
40,-50,-50,-40,-40,-30},{-30,-40,-40,-50,-50,-40,-40,-30},{-30,-40,-40,-
50,-50,-40,-40,-30},{-30,-40,-40,-50,-50,-40,-40,-30},{-20,-30,-30,-40,-
40,-30,-30,-20},{-10,-20,-20,-20,-20,-20,-20,-
10},{20,20,0,0,0,0,20,20},{20,30,10,0,0,10,30,20}};
    private static final int[][]KING_POSITIONAL_BLACK_MID={{-20,-30,-
10,0,0,-10,-30,-20},{-20,-20,0,0,0,0,-20,-
20},{10,20,20,20,20,20,20,10},{20,30,30,40,40,30,30,20},{30,40,40,50,50,
40,40,30},{30,40,40,50,50,40,40,30},{30,40,40,50,50,40,40,30},{30,40,40,
50,50,40,40,30}};
    private static final int[][]KING_POSITIONAL_WHITE_END={{-50,-40,-
30,-20,-20,-30,-40,-50},{-30,-20,-10,0,0,-10,-20,-30},{-30,-
10,20,30,30,20,-10,-30},{-30,-10,30,40,40,30,-10,-30},{-30,-
10,30,40,40,30,-10,-30},{-30,-10,20,30,30,20,-10,-30},{-30,-30,0,0,0,0,-
30,-30},{-50,-30,-30,-30,-30,-30,-30,-50}};
    private static final
int[][]KING_POSITIONAL_BLACK_END={{50,30,30,30,30,30,30,50},{30,30,0,0,0
,0,30,30},{30,10,-20,-30,-30,-20,10,30},{30,10,-30,-40,-40,-
30,10,30},{30,10,-30,-40,-40,-30,10,30},{30,10,-20,-30,-30,-
20,10,30},{30,20,10,0,0,10,20,30},{50,40,30,20,20,30,40,50}};
    //stores the rough value of each of the chess pieces in centipawns
    private static final int PAWN=100;
    private static final int KNIGHT=320;
    private static final int BISHOP=330;
    private static final int ROOK =500;
    private static final int KING=20000;
    private static final int QUEEN=900;
    private ChessNeuralNetwork net;//stores a neural network trained to
evalaute board positions
    private  final String[][]OPENING_BOOK;//stores the opening book -
small number of moves made by chess grandmasters early in a game
    /**
     * Creates a chess game object storing all the states played so far
     * @param statesSoFar All chess board states that have already
occurred
     * @param loadNeuralNet Whether a neural network should be loaded
when initialising the chess game
     */
    public ChessGame(ChessBoardState[]statesSoFar,boolean
loadNeuralNet){
        OPENING_BOOK=getOpeningBook();//loads the opening book
        states =statesSoFar;//stores all board states so far in the game
        if(loadNeuralNet){
            //loads the neural network to evaluate the board states
            try{
```

```
                net= new ChessNeuralNetwork(new
int[]{/*777,610,300,200,100,2*/777,300,200,100,2});
            }catch(IOException e){
                net=null;
                JOptionPane.showMessageDialog(null, "An unexpected error
occured when loading the neural network. Setting the computer to level 4
insetad. Error details: "+e,"Error",JOptionPane.OK_OPTION);
            }
        }

    }
    /**
     * Reads and loads opening book
     * @return The opening book
     */
    private static String[][] getOpeningBook(){
        String arr[][]=new String[11318][2];//stores the data set of
opening moves. It is stored in the form of boardState(t) ->
boardState(t+1)
        try{
            FileReader read =new FileReader("openingMoves.txt");//reads
all opening moves from the text file that stores them
            BufferedReader buffRead = new BufferedReader(read);
            int i=0;
            String lineRead;String line[];
            while((lineRead=buffRead.readLine())!=null){
                line=lineRead.split(",");
                arr[i][0]=line[0];arr[i][1]=line[1];
                i++;
            }
            buffRead.close();
            read.close();
        }catch(IOException e){
            System.out.println("error with opening book: "+e);
        }
        return arr;
    }
    /**
     * Gets the computer move given the computer level as input.
     * @param computerLevel An integer between 1 and 5 (both inclusive).
     * @return The board state representing the move that has been
executed
     */
    public  ChessBoardState getComputerMove(int computerLevel){
        if(getCurrentState().FULL_MOVE_NUMBER<=10 &&
computerLevel!=5){//plays a move out of the opening book if possible
            String fen =getCurrentState().getFenString();//gets the fen
string of the current board state
            String fenData[]=fen.split(" ");
            int i=Search.searchForRecordIndex(OPENING_BOOK,
fen);//searches for the current board state
            if(i!=Search.INDEX_NOT_FOUND){
                int start=i;int end=i;
```

```
                //finds the lower and upper bound of it
                for(;start>0;start--){
                    if(OPENING_BOOK[start][0].equals(fen)==false){
                        start++;break;
                    }
                }
                for(;end<OPENING_BOOK.length;end++){
                    if(OPENING_BOOK[end][0].equals(fen)==false){
                        end--;
                        break;
                    }
                }
                //returns random board state from possible options
                Random rnd = new Random();
                if(getCurrentState().IS_BLACK_TURN_TO_PLAY_NEXT){
                    fen="
"+String.valueOf(getCurrentState().FULL_MOVE_NUMBER+1);
                }else{
                    fen="
"+String.valueOf(getCurrentState().FULL_MOVE_NUMBER);
                }
                try{
                    Thread.sleep(500);//doesn't make the move instantly
                }catch(InterruptedException e){

                }
                return new
ChessBoardState(OPENING_BOOK[rnd.nextInt((end-
start)+1)+start][1]+fen);//creates a new chess board state and returns
it
            }
        }
        //gets the computer move depedning on the computer level
selected
        switch (computerLevel) {
            case 1:
                //gets a random move and plays it
                ArrayList<ChessBoardState>possibleStates=getAllLegalBoar
dsFromState(this.getCurrentState());
                Random rnd = new Random();
                try{
                    Thread.sleep(500);//doesn't make the move instantly
                }catch(InterruptedException e){

                }
                return
(possibleStates.get(rnd.nextInt(possibleStates.size())));//returns the
random move
            case 3:
                //implements a depth limited minimax algorithm looking
at the value of pieces and summing them
                return getComputerMoveMinimax(getCurrentState(),
computerLevel, 3);
```

```
            case 2:
                //uses minimax algorithm alongside a neural network I
have trained
                if(net==null){// if a network error occurs then the move
for the level four computer is retrieved instead
                    return getComputerMove(4);
                }
                return getComputerMoveMinimax(getCurrentState(),
computerLevel, 3);
            case 4:
                //uses an advanced evalaution function
                return getComputerMoveMinimax(getCurrentState(),
computerLevel, 3);
            default://computer level five
                //uses StockFish to get a move;
                try{
                    StockFish stock = new StockFish();//loads stockfish
                    ChessBoardState state =new
ChessBoardState(stock.getStockFishMove(getCurrentState().getFenString())
);//fetches the appropriate board state
                    stock.exit();//exits stock fish
                    return state;
                }catch(IOException e){
                    JOptionPane.showMessageDialog(null, "Failed to load
stockfish (the level five computer). Hence, a move is being played by
the level four computer. This could be because your system is not a
windows 64 bit device.","Error",JOptionPane.OK_OPTION);
                }
                return getComputerMove(4);
        }
    }
    /**
     * Implements the minimax algorithm and returns the board state that
is to be played.
     * @param currentState The starting board state
     * @param computerLevel The computer level. Levels from 2-5 are
valid.
     * @param searchDepth The search depth for the minimax algorithm.
     * @return The chess board
     */
    private  ChessBoardState getComputerMoveMinimax(ChessBoardState
currentState, int computerLevel,int searchDepth){
        ArrayList<ChessBoardState>allStates =
getAllLegalBoardsFromState(currentState);//gets all legal board states
from the current board
        int bestIndex=0;//stores the index of the state that is best
        int currentBest;
        int valOfMove;
        if(currentState.IS_BLACK_TURN_TO_PLAY_NEXT){
            currentBest=Integer.MAX_VALUE;
        }else{
            currentBest=Integer.MAX_VALUE*-1;
            Collections.reverse(allStates);
```

```
        }
        //loops through all board states and finds the board state which
is best for the current player
        for(int i=0;i<allStates.size();i++){
            if(currentState.IS_BLACK_TURN_TO_PLAY_NEXT){//if it's
black's turn, they are aiming to minimise the final value
                valOfMove=maxTurnMinimax(allStates.get(i), 1, -
1*Integer.MAX_VALUE, Integer.MAX_VALUE, searchDepth,
computerLevel);//gets value of board state
                if(valOfMove<=currentBest){
                    currentBest=valOfMove;
                    bestIndex=i;
                }
            }else{//white aims to maximise the board value
                valOfMove=minTurnMinimax(allStates.get(i), 1, -
1*Integer.MAX_VALUE, Integer.MAX_VALUE, searchDepth, computerLevel);
                if(valOfMove>=currentBest){
                    currentBest=valOfMove;
                    bestIndex=i;
                }
            }
        }
        //returns the best move calculated
        return allStates.get(bestIndex);
    }
    /**
     * Implements a recursive depth limited alpha-beta pruned minimax
algorithm to evaluate which move the computer must make next. This is
called by the maximising player
     * @param state The board state to evaluate
     * @param depth The current search depth
     * @param alpha The value of alpha
     * @param beta The value of beta
     * @param maxSearchDepth The maximum search depth
     * @param computerLevel The computer level
     * @return The board value
     */
    private  int maxTurnMinimax(ChessBoardState state, int depth, int
alpha, int beta, int maxSearchDepth, int computerLevel){
        ArrayList<ChessBoardState>possibleStates =
getAllLegalBoardsFromState(state);//gets all possible board from all
current board
        Collections.reverse(possibleStates);//reverses the board states
as move ordering is employed to maximise the number of nodes pruned. As
the moves are odered from lowest points to highest, the move order needs
to be reversed
        if(possibleStates.isEmpty()){//if board is in a terminal state,
it returns the appropraite value
            if(state.getIsDraw()){
                return DRAW_VALUE;
            }
            if(state.IS_BLACK_TURN_TO_PLAY_NEXT){
                return WHITE_WIN_VALUE;
```

```
            }else{
                return BLACK_WIN_VALUE;
            }
        }
        if(depth==maxSearchDepth){//if the maximum search depth has been
reached, the appropriate evaluation is called
            return getBoardValue(state,computerLevel);
        }
        //if the search depth and terminal staten haven't been reached,
the minimax process continues
        int max=-1*Integer.MAX_VALUE;//stores the max value
        int currentMoveValue;//stores the value of a single move
        for (ChessBoardState possibleState : possibleStates) {//loops
through all possible board states
            currentMoveValue=
minTurnMinimax(possibleState,depth+1,alpha,beta,maxSearchDepth,computerL
evel);
            if(currentMoveValue>max){//sets max value
                max=currentMoveValue;
            }
            //implements alpha beta pruning
            if(currentMoveValue>=beta){
                return max;
            }
            if(currentMoveValue>alpha){
                alpha =currentMoveValue;
            }
        }
        //returns the value of the board state
        return max;
    }
    /**
     * Implements a recursive depth limited alpha-beta pruned minimax
algorithm to evaluate which move the computer must make next. This is
called by the minimising player
     * @param state The board state to evaluate
     * @param depth The current search depth
     * @param alpha The value of alpha
     * @param beta The value of beta
     * @param maxSearchDepth The maximum search depth
     * @param computerLevel The computer level
     * @return The board value
     */
    private  int minTurnMinimax(ChessBoardState state, int depth, int
alpha, int beta, int maxSearchDepth, int computerLevel){
        ArrayList<ChessBoardState>possibleStates
=getAllLegalBoardsFromState(state);//gets all possible board from all
current board
        if(possibleStates.isEmpty()){//if board is in a terminal state,
it returns the appropraite value
            if(state.getIsDraw()){
                return DRAW_VALUE;
            }
```

```
            if(state.IS_BLACK_TURN_TO_PLAY_NEXT){
                return WHITE_WIN_VALUE;
            }else{
                return BLACK_WIN_VALUE;
            }
        }
        if(depth==maxSearchDepth){//if the maximum search depth has been
reached, the appropriate evaluation is called
            return getBoardValue(state, computerLevel);
        }
        int min=Integer.MAX_VALUE;//stores the minimum value
        int currentMoveValue;//stores the value of a single move
        for (ChessBoardState possibleState : possibleStates) {//loops
through all possible board states
            currentMoveValue=
maxTurnMinimax(possibleState,depth+1,alpha,beta,maxSearchDepth,computerL
evel);
            if(currentMoveValue<min){//sets min value
                min=currentMoveValue;
            }
            //implements alpha beta pruning
            if(currentMoveValue<=alpha){
                return min;
            }
            if(currentMoveValue<beta){
                beta =currentMoveValue;
            }
        }
        //returns value of board state
        return min;
    }
    /**
     * Gets the value of a non terminal board state using an appropriate
heuristic
     * @param state The board state
     * @param computerLevel The computer level
     * @return The value of the board
     */
    private int getBoardValue(ChessBoardState state, int computerLevel){
        int boardValue=0;//stores the board value
        switch(computerLevel){
            case 3:
                //sums the value of the pieces on the board to calculate
a board value
                for(int y=0;y<8;y++){
                    for(int x=0;x<8;x++){
                        switch (state.BOARD[y][x]) {
                            case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(VALUE_OF_PAWN*BLA
CK_MULT);
                                break;
                            case
```

```
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(VALUE_OF_KNIGHT*B
LACK_MULT);
                                break;
                            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(VALUE_OF_BISHOP*B
LACK_MULT);
                                break;
                            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(VALUE_OF_QUEEN*BL
ACK_MULT);
                                break;
                            case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(VALUE_OF_ROOK*BLA
CK_MULT);
                                break;
                            case
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(VALUE_OF_KING*BLA
CK_MULT);
                                break;
                            case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                boardValue=boardValue+(VALUE_OF_PAWN*WHI
TE_MULT);
                                break;
                            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                boardValue=boardValue+(VALUE_OF_KNIGHT*W
HITE_MULT);
                                break;
                            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                boardValue=boardValue+(WHITE_MULT*VALUE_
OF_BISHOP);
                                break;
                            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                boardValue=boardValue+(WHITE_MULT*VALUE_
OF_QUEEN);
                                break;
                            case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                boardValue =
boardValue+(WHITE_MULT*VALUE_OF_ROOK);
                                break;
                            case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                boardValue=boardValue+(WHITE_MULT*VALUE_
OF_KING);
```

```
                                break;
                            default:
                                break;
                        }
                    }
                }
                break;
            case 5:
                //this evaluation is used as sometimes users may not be
able to use stockfish.
                //stores various variables to determine whether board is
in endgame or not
                //evaluation is based heavily on
https://www.chessprogramming.org/Simplified_Evaluation_Function
                int []coordBlackKing=new int[2];int[]coordWhiteKing=new
int[2];
                int noOfBlackMinorPieces=0;int noOfWhiteMinorPieces=0;
                int noOfPiecesBlack=0;int noOfPiecesWhite=0;
                boolean blackHasQueen=false;boolean whiteHasQueen=false;
                for(int y=0;y<8;y++){
                    for(int x=0;x<8;x++){
                        switch (state.BOARD[y][x]) {
                            case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(PAWN*BLACK_MULT)+
PAWN_POSITIONAL_BLACK[y][x];
                                noOfPiecesBlack++;
                                break;
                            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(KNIGHT*BLACK_MULT
)+KNIGHT_POSITIONAL_BLACK[y][x];
                                noOfBlackMinorPieces++;
                                break;
                            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                noOfBlackMinorPieces++;
                                boardValue=boardValue+(BISHOP*BLACK_MULT
)+BISHOP_POSITIONAL_BLACK[y][x];
                                break;
                            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                blackHasQueen=true;
                                boardValue=boardValue+(QUEEN*BLACK_MULT)
+QUEEN_POSITIONAL_BLACK[y][x];
                                break;
                            case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                noOfPiecesBlack++;
                                boardValue=boardValue+(ROOK*BLACK_MULT)+
ROOK_POSITIONAL_BLACK[y][x];
                                break;
                            case
```

```
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(KING*BLACK_MULT);
                                coordBlackKing[0]=y;coordBlackKing[1]=x;
                                break;
                            case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                noOfPiecesWhite++;
                                boardValue=boardValue+(PAWN*WHITE_MULT)+
PAWN_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                noOfWhiteMinorPieces++;
                                boardValue=boardValue+(KNIGHT*WHITE_MULT
)+KNIGHT_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                noOfWhiteMinorPieces++;
                                boardValue=boardValue+(WHITE_MULT*BISHOP
)+BISHOP_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                whiteHasQueen=true;
                                boardValue=boardValue+(WHITE_MULT*QUEEN)
+QUEEN_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                noOfPiecesWhite++;
                                boardValue =
boardValue+(WHITE_MULT*ROOK)+ROOK_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                boardValue=boardValue+(WHITE_MULT*KING);
                                coordWhiteKing[0]=y;coordWhiteKing[1]=x;
                                break;
                            default:
                                break;
                        }
                    }
                }
                if(((noOfPiecesBlack==0&&noOfBlackMinorPieces<=1)||!blac
kHasQueen)&&((noOfPiecesWhite==0&&noOfWhiteMinorPieces<=1)||!whiteHasQue
en)){//this statement is true if the game is in an endgame state
                    boardValue=boardValue+KING_POSITIONAL_BLACK_END[coor
dBlackKing[0]][coordBlackKing[1]]+KING_POSITIONAL_WHITE_END[coordWhiteKi
ng[0]][coordWhiteKing[1]];
                }else{
                    boardValue =
boardValue+KING_POSITIONAL_BLACK_MID[coordBlackKing[0]][coordBlackKing[1
```

```
]]+KING_POSITIONAL_WHITE_MID[coordWhiteKing[0]][coordWhiteKing[1]];
                }
                break;
            case 4:
                //calculates the board value by summing the value of the
pieces and their relative positional values
                //stores various variables to determine whether board is
in endgame or not
                //evaluation is based heavily on
https://www.chessprogramming.org/Simplified_Evaluation_Function
                coordBlackKing=new int[2];coordWhiteKing=new int[2];
                noOfBlackMinorPieces=0;noOfWhiteMinorPieces=0;
                noOfPiecesBlack=0;noOfPiecesWhite=0;
                blackHasQueen=false;whiteHasQueen=false;
                for(int y=0;y<8;y++){
                    for(int x=0;x<8;x++){
                        switch (state.BOARD[y][x]) {
                            case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(PAWN*BLACK_MULT)+
PAWN_POSITIONAL_BLACK[y][x];
                                noOfPiecesBlack++;
                                break;
                            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(KNIGHT*BLACK_MULT
)+KNIGHT_POSITIONAL_BLACK[y][x];
                                noOfBlackMinorPieces++;
                                break;
                            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                noOfBlackMinorPieces++;
                                boardValue=boardValue+(BISHOP*BLACK_MULT
)+BISHOP_POSITIONAL_BLACK[y][x];
                                break;
                            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                blackHasQueen=true;
                                boardValue=boardValue+(QUEEN*BLACK_MULT)
+QUEEN_POSITIONAL_BLACK[y][x];
                                break;
                            case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                noOfPiecesBlack++;
                                boardValue=boardValue+(ROOK*BLACK_MULT)+
ROOK_POSITIONAL_BLACK[y][x];
                                break;
                            case
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                                boardValue=boardValue+(KING*BLACK_MULT);
                                coordBlackKing[0]=y;coordBlackKing[1]=x;
                                break;
                            case
```

```
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                noOfPiecesWhite++;
                                boardValue=boardValue+(PAWN*WHITE_MULT)+
PAWN_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                noOfWhiteMinorPieces++;
                                boardValue=boardValue+(KNIGHT*WHITE_MULT
)+KNIGHT_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                noOfWhiteMinorPieces++;
                                boardValue=boardValue+(WHITE_MULT*BISHOP
)+BISHOP_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                whiteHasQueen=true;
                                boardValue=boardValue+(WHITE_MULT*QUEEN)
+QUEEN_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                noOfPiecesWhite++;
                                boardValue =
boardValue+(WHITE_MULT*ROOK)+ROOK_POSITIONAL_WHITE[y][x];
                                break;
                            case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                                boardValue=boardValue+(WHITE_MULT*KING);
                                coordWhiteKing[0]=y;coordWhiteKing[1]=x;
                                break;
                            default:
                                break;
                        }
                    }
                }
                if(((noOfPiecesBlack==0&&noOfBlackMinorPieces<=1)||!blac
kHasQueen)&&((noOfPiecesWhite==0&&noOfWhiteMinorPieces<=1)||!whiteHasQue
en)){//this statement is true if the game is in an endgame state
                    boardValue=boardValue+KING_POSITIONAL_BLACK_END[coor
dBlackKing[0]][coordBlackKing[1]]+KING_POSITIONAL_WHITE_END[coordWhiteKi
ng[0]][coordWhiteKing[1]];
                }else{
                    boardValue =
boardValue+KING_POSITIONAL_BLACK_MID[coordBlackKing[0]][coordBlackKing[1
]]+KING_POSITIONAL_WHITE_MID[coordWhiteKing[0]][coordWhiteKing[1]];
                }
                break;
            case 2:
                //feeds the board through a neural network to get the
```

```
board value
                double
ret[]=net.feedThroughNet(state.getNeuralNetInput());
                boardValue = (int)(Math.round((ret[0]-
ret[1])*(Integer.MAX_VALUE-10)));
                break;
        }

        return boardValue;
    }
    /**
     * Gets the number of states in the current game
     * @return The number of states
     */
    public int getLengthOfStatesFoFar(){
        return states.length;
    }
    /**
     * Returns the state stored at the specified index in the array of
all states.
     * @param element The index
     * @return The state
     */
    public ChessBoardState getState(int element){
        return this.states[element];
    }
    /**
     * Adds board state to game.
     * @param state The board state
     */
    public void addState(ChessBoardState state){
        ChessBoardState[]statesNew = new
ChessBoardState[states.length+1];
        statesNew[statesNew.length-1]=state;
        for(int i=0;i<statesNew.length-1;i++){
            statesNew[i]=states[i];
        }
        states = statesNew;
    }
    /**
     * Returns all boards that have been played so far
     * @return All chess board states as an array. Note this array is
not a copy, so states should not be written to unless appropriate.
     */
    public ChessBoardState[]getAllStates(){
        return this.states;
    }
    /**
     * Updates copy of castling rights if move is made for that square.
This is used to ensure that castle rights reflect a potential rook
capture.
     * If the y and x coordinate match the starting position of any
rook, the castling option for that rook is set to false.
```

```
     * @param yCoord The y coordinate of the move
     * @param xCoord The x coordinate of the move
     * @param copyOfCastlingRights A copy of the castling rights. This
array will be manipulated.
     */
    private static void getCastlingRightsAfterCapture(int yCoord, int
xCoord,boolean copyOfCastlingRights[]){
        if(yCoord==0&&xCoord==0){//castle used for black queenside
            copyOfCastlingRights[3]=false;
        }else if(yCoord==0&&xCoord==7){//castle used for black kingside
            copyOfCastlingRights[2]=false;
        }else if(yCoord==7&& xCoord==0){//white queenside
            copyOfCastlingRights[1]=false;
        }else if(yCoord==7&&xCoord==7){//white kingside
            copyOfCastlingRights[0]=false;
        }
    }
    /**
     * Gets all legal board states from the chosen position. Note:
normally only one legal board state will be returned but
     * pawn promotion will return four states (one for each possible
choice of piece). DOES NOT handle 50 move rule
     * or other draw / win /loss conditions for current board, these
must be handled elsewhere where appropriate.Note moves input must be
legal
     * @param yStart The y coordinate of the piece
     * @param xStart The x coordinate of the piece
     * @param yFinal The y coordinate of the move
     * @param xFinal The x coordinate of the move
     * @param state The board state to analyse
     * @return All possible states. If no states exist, an array of
length 0 is returned
     */
    public static  ArrayList<ChessBoardState>
getAllLegalBoardsFromMove(int yStart,int xStart, int yFinal,int
xFinal,ChessBoardState state){
        int[][]boardNew=state.getCopyOfBoard();//stores the new board
        boolean
[]castlingRights=state.getCopyOfCastlingRights();//stores the new
castling rights
        ArrayList<ChessBoardState> legalBoard=new ArrayList<>(1);
        int[]enPassantSquare;//stores the en passsant square
        int halfMove=state.HALF_MOVE_CLOCK;//stores the number of half
moves;
        int fullMoveNo=state.FULL_MOVE_NUMBER;//stores the number of
moves made by black
        switch (state.BOARD[yStart][xStart]) {
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                legalBoard=new ArrayList<>(5);
                fullMoveNo++;//increments moves as black has moved
                halfMove=0;//as pawn advance resets half move clock,
half move will always be at 0 after a pawn move
```

```
                //checks to see if pawn has moved two forward (and
updates en passant rights accoridngly)
                enPassantSquare=new int[0];
                if(yFinal == yStart+2){
                    enPassantSquare =new
int[2];enPassantSquare[0]=yStart+1;enPassantSquare[1]=xStart;
                }
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                //processes pawn move
                //processes possible pawn promotion
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;
                if(yFinal==7){//can create four possible boards
                    //creates queen board
                    boardNew[7][xFinal]=ChessBoardState.QUEEN_VALUE*Ches
sBoardState.BLACK_MULTIPLIER;
                    legalBoard.add(new ChessBoardState(boardNew,
castlingRights, !state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                    //handles bishop
                    boardNew[7][xFinal]=ChessBoardState.ROOK_VALUE*Chess
BoardState.BLACK_MULTIPLIER;
                    legalBoard.add(new ChessBoardState(boardNew,
castlingRights, !state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                    //handles knight
                    boardNew[7][xFinal]=ChessBoardState.BISHOP_VALUE*Che
ssBoardState.BLACK_MULTIPLIER;
                    legalBoard.add(new ChessBoardState(boardNew,
castlingRights, !state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                    //handles rook
                    boardNew[7][xFinal]=ChessBoardState.KNIGHT_VALUE*Che
ssBoardState.BLACK_MULTIPLIER;
                    legalBoard.add(new ChessBoardState(boardNew,
castlingRights, !state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                    return legalBoard;//returns states
                }
                boardNew[yFinal][xFinal]=ChessBoardState.PAWN_VALUE*Ches
sBoardState.BLACK_MULTIPLIER;
                if(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==
2){//checks for en passant capture
                    if(yFinal==state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASS
ANT[0]&&xFinal==state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]){
                        boardNew[yStart][xFinal]=ChessBoardState.EMPTY_V
ALUE;//captures pawn via en passant
                    }
                }
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                return legalBoard;//returns move
```

```
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                fullMoveNo++;//increments moves as black has moved
                //simply replaces moves knight to square
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK
_MULTIPLIER;
                //checks to see if a capture has been made
                if(state.BOARD[yFinal][xFinal]!=ChessBoardState.EMPTY_VA
LUE){
                    halfMove=0;
                }else{
                    halfMove++;
                }
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                fullMoveNo++;//increments moves as black has moved
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                //simply replaces moves bishop to square
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK
_MULTIPLIER;
                //checks to see if a capture has been made
                if(state.BOARD[yFinal][xFinal]!=ChessBoardState.EMPTY_VA
LUE){
                    halfMove=0;
                }else{
                    halfMove++;
                }
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                fullMoveNo++;//increments moves as black has moved
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                //simply replaces moves rook to square
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
```

```
rdNew[yFinal][xFinal]=ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_M
ULTIPLIER;
                //checks to see if a capture has been made
                if(state.BOARD[yFinal][xFinal]!=ChessBoardState.EMPTY_VA
LUE){
                    halfMove=0;
                }else{
                    halfMove++;
                }
                //updates castling rights if appropriate
                if(yStart==0){
                    if(xStart==0){//black queenside
                        castlingRights[3]=false;
                    }else if(xStart==7){//black kingside
                        castlingRights[2]=false;
                    }
                }else if(yStart==7){
                    if(xStart==0){//white queenside
                        castlingRights[1]=false;
                    }else if(xStart==7){//white kingside
                        castlingRights[0]=false;
                    }
                }
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                fullMoveNo++;//increments moves as black has moved
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                //simply replaces moves queen to square
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_
MULTIPLIER;
                //checks to see if a capture has been made
                if(state.BOARD[yFinal][xFinal]!=ChessBoardState.EMPTY_VA
LUE){
                    halfMove=0;
                }else{
                    halfMove++;
                }
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
```

```
            case
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                fullMoveNo++;//increments moves as black has moved
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                //moves king
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.KING_VALUE*ChessBoardState.BLACK_M
ULTIPLIER;
                //checks to see if castling is happening and updates
rook as appropraite
                if(xFinal==xStart+2){//kingside castling
                    boardNew[0][5]=ChessBoardState.BLACK_MULTIPLIER*Ches
sBoardState.ROOK_VALUE;boardNew[0][7]=ChessBoardState.EMPTY_VALUE;
                }else if(xFinal==xStart-2){//queenside castling
                    boardNew[0][3]=ChessBoardState.BLACK_MULTIPLIER*Ches
sBoardState.ROOK_VALUE;boardNew[0][0]=ChessBoardState.EMPTY_VALUE;
                }
                //updates castling rights
                castlingRights[2]=false;castlingRights[3]=false;
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                legalBoard=new ArrayList<>(5);
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                halfMove=0;//as pawn advance resets half move clock,
half move will always be at 0 after a pawn move
                //checks to see if pawn has moved two forward (and
updates en passant rights accoridngly)
                enPassantSquare=new int[0];
                if(yFinal == yStart-2){
                    enPassantSquare =new
int[2];enPassantSquare[0]=yStart-1;enPassantSquare[1]=xStart;
                }
                //processes pawn move
                //processes possible pawn promotion
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;
                if(yFinal==0){//can create four possible boards
                    //creates queen board
                    boardNew[0][xFinal]=ChessBoardState.QUEEN_VALUE*Ches
sBoardState.WHITE_MULTIPLIER;
                    legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                    //handles bishop
                    boardNew[0][xFinal]=ChessBoardState.ROOK_VALUE*Chess
BoardState.WHITE_MULTIPLIER;
```

```
                    legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                    //handles knight
                    boardNew[0][xFinal]=ChessBoardState.BISHOP_VALUE*Che
ssBoardState.WHITE_MULTIPLIER;
                    legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                    //handles rook
                    boardNew[0][xFinal]=ChessBoardState.KNIGHT_VALUE*Che
ssBoardState.WHITE_MULTIPLIER;
                    legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));
                    return legalBoard;//returns states
                }
                boardNew[yFinal][xFinal]=ChessBoardState.PAWN_VALUE*Ches
sBoardState.WHITE_MULTIPLIER;
                if(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==
2){//checks for en passant capture
                    if(yFinal==state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASS
ANT[0]&&xFinal==state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]){
                        boardNew[yStart][xFinal]=ChessBoardState.EMPTY_V
ALUE;//captures pawn via en passant
                    }
                }
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                //simply replaces moves knight to square
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE
_MULTIPLIER;
                //checks to see if a capture has been made
                if(state.BOARD[yFinal][xFinal]!=ChessBoardState.EMPTY_VA
LUE){
                    halfMove=0;
                }else{
                    halfMove++;
                }
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            case
```

```
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                //simply replaces moves bishop to square
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE
_MULTIPLIER;
                //checks to see if a capture has been made
                if(state.BOARD[yFinal][xFinal]!=ChessBoardState.EMPTY_VA
LUE){
                    halfMove=0;
                }else{
                    halfMove++;
                }
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                //simply replaces moves rook to square
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_M
ULTIPLIER;
                //checks to see if a capture has been made
                if(state.BOARD[yFinal][xFinal]!=ChessBoardState.EMPTY_VA
LUE){
                    halfMove=0;
                }else{
                    halfMove++;
                }
                //updates castling rights if appropriate
                if(yStart==0){
                    if(xStart==0){//black queenside
                        castlingRights[3]=false;
                    }else if(xStart==7){//black kingside
                        castlingRights[2]=false;
                    }
                }else if(yStart==7){
                    if(xStart==0){//white queenside
                        castlingRights[1]=false;
                    }else if(xStart==7){//white kingside
                        castlingRights[0]=false;
                    }
                }
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
```

```
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                //simply replaces moves queen to square
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_
MULTIPLIER;
                //checks to see if a capture has been made
                if(state.BOARD[yFinal][xFinal]!=ChessBoardState.EMPTY_VA
LUE){
                    halfMove=0;
                }else{
                    halfMove++;
                }
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
            default:
                //white king
                getCastlingRightsAfterCapture(yFinal, xFinal,
castlingRights);//updates castling rights where appropriate
                enPassantSquare=new int[0];//as pawn is not moved, no
squares can be avaliable for en passant
                //moves king
                boardNew[yStart][xStart]=ChessBoardState.EMPTY_VALUE;boa
rdNew[yFinal][xFinal]=ChessBoardState.KING_VALUE*ChessBoardState.WHITE_M
ULTIPLIER;
                //checks to see if castling is happening and updates
rook as appropraite
                if(xFinal==xStart+2){//kingside castling
                    boardNew[7][5]=ChessBoardState.WHITE_MULTIPLIER*Ches
sBoardState.ROOK_VALUE;boardNew[7][7]=ChessBoardState.EMPTY_VALUE;
                }else if(xFinal==xStart-2){//queenside castling
                    boardNew[7][3]=ChessBoardState.WHITE_MULTIPLIER*Ches
sBoardState.ROOK_VALUE;boardNew[7][0]=ChessBoardState.EMPTY_VALUE;
                }
                //updates castling rights
                castlingRights[0]=false;castlingRights[1]=false;
                legalBoard.add(new ChessBoardState(boardNew,
castlingRights,!state.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMove,
enPassantSquare));//returns move
                return legalBoard;
        }
```

```
    }
    /**
     * Gets all legal board states from the input current board state.
DOES handle 50 move rule
     * @param currentBoard The board state
     * @return An array of all legal board states. If this array is of
length 0, there are no legal board states from this board state
     */
    public ArrayList<ChessBoardState>
getAllLegalBoardsFromState(ChessBoardState currentBoard){
        ArrayList<ChessBoardState> allNewStates=new
ArrayList<>(35);//stores all legal board states resulting from current
board state
        //checks for draw due to 50 move rule - players are
automatically drawn
        if(currentBoard.HALF_MOVE_CLOCK>=100){
            currentBoard.setIsDrawTrue();
            return allNewStates;//returns no legal moves
        }
        //checks for a draw due to insufficient pieces
        /*draw condition with insufficient pieces:
        only king and king, only king-bishop and king, king-knight and
king, king-bishop and bishop-king with bishops on same colour square
        */
        int noOfWhiteKnights=0;int noOfBlackKnights=0;int
noOfWhiteBishops=0;int noOfBlackBishops=0; boolean
isBlackBishopOnBlack=false;
        int noOfOtherPieces=0;//stores number of pieces that are not
king, bishop or knights
        boolean isWhiteBishopOnBlack=false;boolean
isBlackBishopOnWhite=false;boolean isWhiteBishopOnWhite=false;boolean
buff;
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                switch (currentBoard.BOARD[y][x]) {
                    case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        noOfWhiteKnights++;
                        break;
                    case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        noOfBlackKnights++;
                        break;
                    case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        noOfBlackBishops++;
                        buff=(y%2==0&&x%2==0)||(y%2==1&&x%2==1);//calcul
ates what colour bishop is on
                        isBlackBishopOnWhite=buff||isBlackBishopOnWhite;
                        isBlackBishopOnBlack=!buff
||isBlackBishopOnBlack;
                        break;
                    case
```

```
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                        noOfWhiteBishops++;
                        buff=(y%2==0&&x%2==0)||(y%2==1&&x%2==1);//calcul
ates what colur bishop is on
                        isWhiteBishopOnWhite=buff||isWhiteBishopOnWhite;
                        isWhiteBishopOnBlack=!buff
||isWhiteBishopOnBlack;
                        break;
                    case ChessBoardState.EMPTY_VALUE:
                        break;
                    case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                            break;
                    case
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                        break;
                    default:
                        noOfOtherPieces++;
                        break;
                }
            }
        }
        //checks for draw due to insufficient material
        if(!(noOfOtherPieces>0 ||
((noOfBlackKnights+noOfWhiteKnights)>1) ||
((noOfBlackKnights+noOfWhiteKnights==1) &&
(noOfBlackBishops+noOfWhiteBishops>0)) ||
(noOfBlackBishops>1)||(noOfWhiteBishops>1)||(noOfBlackBishops==1 &&
noOfWhiteBishops==1 &&
((isBlackBishopOnBlack&&isWhiteBishopOnWhite)||(isBlackBishopOnWhite&&is
WhiteBishopOnBlack))))){
            currentBoard.setIsDrawTrue();//sets game as draw
            return allNewStates;//returns no valid options
        }
        ArrayList<Integer>allLegalMoves=getAllLegalMoves(currentBoard);
        if(allLegalMoves.isEmpty()){//when no moves can be made, need to
check whether it's checkmate or stalemate
            if(isInCheck(!currentBoard.IS_BLACK_TURN_TO_PLAY_NEXT,  new
ChessBoardState(currentBoard.BOARD, currentBoard.CASTLING_RIGHTS,
!currentBoard.IS_BLACK_TURN_TO_PLAY_NEXT, currentBoard.FULL_MOVE_NUMBER,
currentBoard.HALF_MOVE_CLOCK,
currentBoard.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT))){
                currentBoard.setIsLossTrue();//checkmate
            }else{
                currentBoard.setIsDrawTrue();//stalemate
            }
            return allNewStates;
        }
        for(int move=0;move<allLegalMoves.size();move=move+4){//loops
through all legal moves and generates all resulting possible board
states
            allNewStates.addAll(getAllLegalBoardsFromMove(allLegalMoves.
get(move), allLegalMoves.get(move+1), allLegalMoves.get(move+2),
```

```
allLegalMoves.get(move+3), currentBoard));
        }
        return allNewStates;
    }
    /**
     * Creates new array of chessboards. Note this should only be called
in the getAllLegalBoardsFromState method.
     * @param array The array
     * @param stateToAdd The new state
     * @return the new array
     */
    private static
ChessBoardState[]addChessBoardStateToArray(ChessBoardState[]array,ChessB
oardState stateToAdd){
        ChessBoardState newStates[]=new ChessBoardState[array.length+1];
        for(int i=0;i<newStates.length;i++){//copies old states
            newStates[i]=array[i];
        }
        newStates[array.length]=stateToAdd;//adds new state
        return newStates;
    }
    /**
     * Creates new array of chessboards. Note this should only be called
in the getAllLegalBoardsFromState method.
     * @param array The array
     * @param statesToAdd The new states
     * @return the new array
     */
    private static
ChessBoardState[]addChessBoardStateToArray(ChessBoardState[]array,ChessB
oardState[]statesToAdd){
        ChessBoardState newStates[]=new
ChessBoardState[array.length+statesToAdd.length];
        for(int i=0;i<array.length;i++){//copies old states to array
            newStates[i]=array[i];
        }
        for(int i=0;i<statesToAdd.length;i++){//adds new states to array
            newStates[i+array.length]=statesToAdd[i];
        }
        return newStates;
    }
    /**
     * Calls getAllLegalMovesFromSquare method on every square of the
chess board and returns all moves. Read documentation of that method for
details on moves
     * @param board The board to analyse
     * @return An ArrayList with all legal moves from the current board.
     */
    private  ArrayList <Integer> getAllLegalMoves(ChessBoardState
board){
        ArrayList<Integer> allMoves=new ArrayList<>(140);//stores all
moves in the form yStart, xStart, yEnd, xEnd
        ArrayList<Integer>movesFromSquare;
```

```
        int index=-1;
        //gets all legal moves from each square on the board
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                movesFromSquare=getAllLegalMovesFromSquare(y, x, board);
                for(int move=0;move<movesFromSquare.size();move=move+2){
                    allMoves.add(++index,y);allMoves.add(++index,x);allM
oves.add(++index,movesFromSquare.get(move));allMoves.add(++index,movesFr
omSquare.get(move+1));
                }
            }
        }
        //moves ordering is employed
        final int SIZE=allMoves.size();
        int allMovesIndex=-1;
        int moveVal[][]=new int[SIZE][2];//stores the approximate value
of the move and the pointer to where it is stored - this is used to
order the moves rougly in order of best to worst
        for(int move=0;move<SIZE/4;move++){
            allMovesIndex++;
            moveVal[move][1]=allMovesIndex;
            moveVal[move][0]=getValueOfMove(board,
allMoves.get(allMovesIndex), allMoves.get(++allMovesIndex),
allMoves.get(++allMovesIndex), allMoves.get(++allMovesIndex));
        }
        //sorts moves by value
        Sort.mergeSortInt(moveVal);
        //swaps moves around so they are sorted from best to worst
        allMovesIndex=-1;
        for(int move=0;move<SIZE/4;move++){
            //swaps moves so they are ordered
            Collections.swap(allMoves,++allMovesIndex,moveVal[move][1]);
            Collections.swap(allMoves,++allMovesIndex,moveVal[move][1]+1
);
            Collections.swap(allMoves,++allMovesIndex,moveVal[move][1]+2
);
            Collections.swap(allMoves,++allMovesIndex,moveVal[move][1]+3
);
        }
        return allMoves;//returns all legal moves
    }
    /**
     * Estimates the value of a move, in order to rank the move values
     * @param state The board state
     * @param yStart The starting y coordinate
     * @param xStart The starting x coordinate
     * @param yEnd The final y coordinate
     * @param xEnd The final x coordinate
     * @return The value of the move
     */
    private int getValueOfMove(ChessBoardState state, int yStart, int
xStart, int yEnd, int xEnd){
        int val=0;
```

```
        //gets the move value by looking how the value of the piece
changes due to it's positional change. The value aslo changes depending
on whether captures are made.
        //alters value for square
        switch (state.BOARD[yStart][xStart]) {
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=PAWN_POSITIONAL_WHITE[yEnd][xEnd]-
PAWN_POSITIONAL_WHITE[yStart][xStart];
                break;
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=BISHOP_POSITIONAL_WHITE[yEnd][xEnd]-
BISHOP_POSITIONAL_WHITE[yStart][xStart];
                break;
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=KNIGHT_POSITIONAL_WHITE[yEnd][xEnd]-
KNIGHT_POSITIONAL_WHITE[yStart][xStart];
                break;
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=ROOK_POSITIONAL_WHITE[yEnd][xEnd]-
ROOK_POSITIONAL_WHITE[yStart][xStart];
                break;
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=QUEEN_POSITIONAL_WHITE[yEnd][xEnd]-
QUEEN_POSITIONAL_WHITE[yStart][xStart];
                break;
            case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=KING_POSITIONAL_WHITE_MID[yEnd][xEnd]-
KING_POSITIONAL_WHITE_MID[yStart][xStart];
                break;
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=PAWN_POSITIONAL_BLACK[yEnd][xEnd]-
PAWN_POSITIONAL_BLACK[yStart][xStart];
                break;
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=BISHOP_POSITIONAL_BLACK[yEnd][xEnd]-
BISHOP_POSITIONAL_BLACK[yStart][xStart];
                break;
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=KNIGHT_POSITIONAL_BLACK[yEnd][xEnd]-
KNIGHT_POSITIONAL_BLACK[yStart][xStart];
                break;
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=ROOK_POSITIONAL_BLACK[yEnd][xEnd]-
```

```
ROOK_POSITIONAL_BLACK[yStart][xStart];
                break;
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=QUEEN_POSITIONAL_BLACK[yEnd][xEnd]-
QUEEN_POSITIONAL_BLACK[yStart][xStart];
                break;
            default:
                val=KING_POSITIONAL_BLACK_MID[yEnd][xEnd]-
KING_POSITIONAL_BLACK_MID[yStart][xStart];
                break;
        }
        switch (state.BOARD[yEnd][xEnd]) {
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=val-(PAWN*ChessBoardState.WHITE_MULTIPLIER +
PAWN_POSITIONAL_WHITE[yEnd][xEnd]);
                break;
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=val-(BISHOP*ChessBoardState.WHITE_MULTIPLIER +
BISHOP_POSITIONAL_WHITE[yEnd][xEnd]);
                break;
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=val-(KNIGHT*ChessBoardState.WHITE_MULTIPLIER +
KNIGHT_POSITIONAL_WHITE[yEnd][xEnd]);
                break;
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=val-(ROOK*ChessBoardState.WHITE_MULTIPLIER +
ROOK_POSITIONAL_WHITE[yEnd][xEnd]);
                break;
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=val-(QUEEN*ChessBoardState.WHITE_MULTIPLIER +
QUEEN_POSITIONAL_WHITE[yEnd][xEnd]);
                break;
            case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                val=val-(KING*ChessBoardState.WHITE_MULTIPLIER +
KING_POSITIONAL_WHITE_MID[yEnd][xEnd]);
                break;
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=PAWN_POSITIONAL_BLACK[yEnd][xEnd]-
PAWN_POSITIONAL_BLACK[yStart][xStart];
                val=val-(PAWN*ChessBoardState.BLACK_MULTIPLIER +
PAWN_POSITIONAL_BLACK[yEnd][xEnd]);
                break;
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=val-(BISHOP*ChessBoardState.BLACK_MULTIPLIER +
```

```
BISHOP_POSITIONAL_BLACK[yEnd][xEnd]);
                break;
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=val-(KNIGHT*ChessBoardState.BLACK_MULTIPLIER +
KNIGHT_POSITIONAL_BLACK[yEnd][xEnd]);
                break;
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=val-(ROOK*ChessBoardState.BLACK_MULTIPLIER +
ROOK_POSITIONAL_BLACK[yEnd][xEnd]);
                break;
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=val-(QUEEN*ChessBoardState.BLACK_MULTIPLIER +
QUEEN_POSITIONAL_BLACK[yEnd][xEnd]);
                break;
            case
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                val=val-(KING*ChessBoardState.BLACK_MULTIPLIER +
KING_POSITIONAL_BLACK_MID[yEnd][xEnd]);
                break;
            default:
                break;
        }
        return val;
    }
    /**
     * gets the value of a piece and returns it
     * @param PIECE The piece
     * @return The relative value of the piece
     */
    private static int getPieceValue(final int PIECE){
       //gets the value of the piece
        switch (PIECE) {
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                return PAWN;
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                return KNIGHT;
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                return ROOK;
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                return BISHOP;
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                return QUEEN;
            case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                return KING;
```

```
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                return PAWN;
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                return KNIGHT;
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                return ROOK;
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                return BISHOP;
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                return QUEEN;
            case
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                return KING;
            default:
                return 0;
        }
    }
    /**
     * Gets all legal moves that can be made from the square
     * Notes: pawn promotion is treated as moving a pawn to the final
rank. However, the actual piece chosen needs to be handled.
     * Notes: half move clock is NOT handled by this method
     * Notes: castling is treated as king moving two spaces to the left
or right
     * @param yStartingSquare The y index of the starting square
     * @param xStartingSquare The x index of the starting square
     * @param board The board state to be analysed
     * @return An array list returning all legal moves from a square
     */
    public static ArrayList <Integer> getAllLegalMovesFromSquare(int
yStartingSquare, int xStartingSquare,ChessBoardState board){
        //checks to see if square selected is empty or a piece of the
wrong colour
        int xVal;
        if(board.BOARD[yStartingSquare][xStartingSquare]==ChessBoardStat
e.EMPTY_VALUE ||
(board.IS_BLACK_TURN_TO_PLAY_NEXT&&board.BOARD[yStartingSquare][xStartin
gSquare]>0)||
(!board.IS_BLACK_TURN_TO_PLAY_NEXT&&board.BOARD[yStartingSquare][xStarti
ngSquare]<0)){
            return new ArrayList<>(0);//returns no legal moves
        }
        ArrayList<Integer> moves;
        switch (board.BOARD[yStartingSquare][xStartingSquare]) {
            case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
            {
                moves=getAllLegalMovesForPawnIgnoringCheck(yStartingSqua
```

```
re, xStartingSquare, board);
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew;//stores possible en passant
squares
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                fullMoveNo++;
                int halfMoveNo=0;//stores half move number -will always
be zero for pawn moves
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    if(moves.get(i)==yStartingSquare+2){//checks for
black pawn moving two squares forward
                        enPassantRightsNew=new
int[]{yStartingSquare+1,xStartingSquare};
                        newBoard=board.getCopyOfBoard();
                        //adds move to board
                        newBoard[yStartingSquare][xStartingSquare]=Chess
BoardState.EMPTY_VALUE;newBoard[yStartingSquare+2][xStartingSquare]=Ches
sBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER;
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }
                    }else if(moves.get(i)==yStartingSquare-2){//checks
for white pawn moving two squares forward
                        enPassantRightsNew=new int[]{yStartingSquare-
1,xStartingSquare};
                        newBoard=board.getCopyOfBoard();
                        //adds move to board
                        newBoard[yStartingSquare][xStartingSquare]=Chess
BoardState.EMPTY_VALUE;newBoard[yStartingSquare-
2][xStartingSquare]=ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MUL
TIPLIER;
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }
                    }else if((moves.get(i)==yStartingSquare-1 &&
moves.get(xVal)==xStartingSquare)||(moves.get(i)==yStartingSquare+1 &&
moves.get(xVal)==xStartingSquare)){//checks for white or black pawn
```

```
moving one square forward
                        enPassantRightsNew= new int[0];//no en passant
rights as piece is moving one square
                        newBoard=board.getCopyOfBoard();//gets copy of
board
                        //adds move
                        newBoard[moves.get(i)][moves.get(xVal)]=ChessBoa
rdState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER;newBoard[yStartingSq
uare][xStartingSquare]=ChessBoardState.EMPTY_VALUE;
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }
                    }else{//must be a diagonal capture
                        enPassantRightsNew=new int[0];
                        if(board.BOARD[moves.get(i)][moves.get(xVal)]==C
hessBoardState.EMPTY_VALUE){//en passant capture
                            newBoard=board.getCopyOfBoard();//gets copy
of board
                            //adds move
                            newBoard[yStartingSquare][moves.get(xVal)]=C
hessBoardState.EMPTY_VALUE;
                            newBoard[yStartingSquare][xStartingSquare]=C
hessBoardState.EMPTY_VALUE;
                            newBoard[moves.get(i)][moves.get(xVal)]=Ches
sBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER;
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NE
XT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                                moves.remove(i);moves.remove(i);//remove
s invalid board states from list
                                i=i-2;
                            }
                        }else{//normal capture
                            newBoard=board.getCopyOfBoard();//gets copy
of board
                            //adds move
                            newBoard[yStartingSquare][xStartingSquare]=C
hessBoardState.EMPTY_VALUE;newBoard[moves.get(i)][moves.get(xVal)]=Chess
BoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER;
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NE
XT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                                moves.remove(i);moves.remove(i);//remove
s invalid board states from list
                                i=i-2;
                            }
```

```
                        }
                    }
                }
                return moves;
            }
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
            {
                moves=getLegalMovesForRookIgnoringCheck(yStartingSquare,
xStartingSquare, board);
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew=new int[0];//no square will be
available for en passant after rook move
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){//updates move
number and en passant rights
                    fullMoveNo++;
                    if(yStartingSquare==0&&xStartingSquare==0){
                        castlingRights[1]=false;
                    }else if(yStartingSquare==0&&xStartingSquare==7){
                        castlingRights[0]=false;
                    }
                }else{
                    if(yStartingSquare==7&&xStartingSquare==0){
                        castlingRights[3]=false;
                    }else if(yStartingSquare==7&&xStartingSquare==7){
                        castlingRights[2]=false;
                    }
                }
                int halfMoveNo;//stores half move number
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    //generates new board
                    newBoard=board.getCopyOfBoard();
                    newBoard[moves.get(i)][moves.get(xVal)]=ChessBoardSt
ate.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER;//moves castle to new
square
                    halfMoveNo=board.HALF_MOVE_CLOCK+1;
                    newBoard[yStartingSquare][xStartingSquare]=ChessBoar
dState.EMPTY_VALUE;//sets old square to empty
                    if(board.BOARD[moves.get(i)][moves.get(xVal)]!=Chess
BoardState.EMPTY_VALUE){//checks to see if rook has captured a piece
                        halfMoveNo=0;
                    }
                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){//checks to make sure move does not result
```

```
in own check getting in check
                        moves.remove(i);moves.remove(i);//removes
invalid board states from list
                        i=i-2;
                    }
                }
                return moves;
            }
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
            {
                moves =
getAllLegalMovesForKnightIgnoringCheck(yStartingSquare, xStartingSquare,
board);
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights - this wont be changed by any knight move
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew=new int[0];//no square will be
available for en passant after knight move
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                    fullMoveNo++;
                }
                int halfMoveNo;//stores half move number
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    //generates new board
                    newBoard=board.getCopyOfBoard();
                    newBoard[moves.get(i)][moves.get(xVal)]=ChessBoardSt
ate.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER;//moves knight to new
square
                    halfMoveNo=board.HALF_MOVE_CLOCK+1;
                    newBoard[yStartingSquare][xStartingSquare]=ChessBoar
dState.EMPTY_VALUE;//sets old square to empty
                    if(board.BOARD[moves.get(i)][moves.get(xVal)]!=Chess
BoardState.EMPTY_VALUE){//checks to see if knight has captured a piece
                        halfMoveNo=0;
                    }
                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){//checks to make sure move does not result
in own check getting in check
                        moves.remove(i);moves.remove(i);//removes
invalid board states from list
                        i=i-2;
                    }
                }
                return moves;
            }
```

```
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
            {
                moves =
getLegalMovesForBishopIgnoringCheck(yStartingSquare, xStartingSquare,
board);
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights - this wont be changed by any bishop move
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew=new int[0];//no square will be
available for en passant after bishop move
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                    fullMoveNo++;
                }
                int halfMoveNo;//stores half move number
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    //generates new board
                    newBoard=board.getCopyOfBoard();
                    newBoard[moves.get(i)][moves.get(xVal)]=ChessBoardSt
ate.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER;//moves bishop to new
square
                    halfMoveNo=board.HALF_MOVE_CLOCK+1;
                    newBoard[yStartingSquare][xStartingSquare]=ChessBoar
dState.EMPTY_VALUE;//sets old square to empty
                    if(board.BOARD[moves.get(i)][moves.get(xVal)]!=Chess
BoardState.EMPTY_VALUE){//checks to see if bishop has captured a piece
                        halfMoveNo=0;
                    }
                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){//checks to make sure move does not result
in own check getting in check
                        moves.remove(i);moves.remove(i);//removes
invalid board states from list
                        i=i-2;
                    }
                }
                return moves;
            }
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
            {
                moves=getLegalMovesForBishopIgnoringCheck(yStartingSquar
e, xStartingSquare, board);
                moves.addAll(getLegalMovesForRookIgnoringCheck(yStarting
Square, xStartingSquare, board));
                boolean
```

```
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights - this wont be changed by any queen move
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew=new int[0];//no square will be
available for en passant after queen move
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                    fullMoveNo++;
                }
                int halfMoveNo;//stores half move number
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    //generates new board
                    newBoard=board.getCopyOfBoard();
                    newBoard[moves.get(i)][moves.get(xVal)]=ChessBoardSt
ate.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER;//moves queen to new
square
                    halfMoveNo=board.HALF_MOVE_CLOCK+1;
                    newBoard[yStartingSquare][xStartingSquare]=ChessBoar
dState.EMPTY_VALUE;//sets old square to empty
                    if(board.BOARD[moves.get(i)][moves.get(xVal)]!=Chess
BoardState.EMPTY_VALUE){//checks to see if queen has captured a piece
                        halfMoveNo=0;
                    }
                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){//checks to make sure move does not result
in own check getting in check
                        moves.remove(i);moves.remove(i);//removes
invalid board states from list
                        i=i-2;
                    }
                }
                return moves;
            }
            case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
            {//king move
                moves=getAllLegalMovesForKingIgnoringCheck(yStartingSqua
re, xStartingSquare, board);
                boolean []castlingRights;//stores castling rights of new
boards;
                int[][]newBoard;//stores new board;
                int[]enPassantSquares;//stores squares for en passant
                int fullMoveNo;//stores the move number
                int halfMoveNo;//stores the half move no
                for(int i=0;i<moves.size();i=i+2){//loops through every
move, updating the board with the move and seeing whether the move would
be legal
                    xVal=i+1;
```

```
                    if(moves.get(xVal)==xStartingSquare+2 ){//checks for
kingside castling
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new
ChessBoardState(board.BOARD,board.CASTLING_RIGHTS,!board.IS_BLACK_TURN_T
O_PLAY_NEXT,board.FULL_MOVE_NUMBER,board.HALF_MOVE_CLOCK,board.SQUARE_TH
AT_CAN_BE_CAPTURED_EN_PASSANT))==true){//king can't castle out of check
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }else{
                            //checks whether king would be in check for
either of the two squares moved
                            //checks for check for move one across
                            newBoard = board.getCopyOfBoard();//gets and
updates board with king moving across
                            newBoard[yStartingSquare][xStartingSquare]=C
hessBoardState.EMPTY_VALUE;newBoard[yStartingSquare][xStartingSquare+1]=
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER;
                            fullMoveNo=board.FULL_MOVE_NUMBER;//updates
move number if required
                            halfMoveNo=board.HALF_MOVE_CLOCK+1;//updates
half move number
                            enPassantSquares=new int[0];//no en passant
square available as king has been moved
                            castlingRights=board.getCopyOfCastlingRights
();//stores castling rights
                            if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                                castlingRights[2]=false;castlingRights[3
]=false;
                                fullMoveNo++;
                            }else{
                                castlingRights[0]=false;castlingRights[1
]=false;
                            }
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NE
XT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){//king can't castle through squares where he
would be in check
                                moves.remove(i);moves.remove(i);//remove
s invalid board states from list
                                i=i-2;
                            }else{
                                newBoard[yStartingSquare][xStartingSquar
e+1]=ChessBoardState.WHITE_MULTIPLIER*ChessBoardState.ROOK_VALUE;newBoar
d[yStartingSquare][7]=ChessBoardState.EMPTY_VALUE;
                                newBoard[yStartingSquare][xStartingSquar
e+2]=ChessBoardState.WHITE_MULTIPLIER*ChessBoardState.KING_VALUE;
                                if(isInCheck(!board.IS_BLACK_TURN_TO_PLA
Y_NEXT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){
```

```
                                    moves.remove(i);moves.remove(i);//re
moves invalid board states from list
                                    i=i-2;
                                }
                            }
                        }
                    }else if(moves.get(xVal)==xStartingSquare-
2){//checks for queenside castling
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new
ChessBoardState(board.BOARD,board.CASTLING_RIGHTS,!board.IS_BLACK_TURN_T
O_PLAY_NEXT,board.FULL_MOVE_NUMBER,board.HALF_MOVE_CLOCK,board.SQUARE_TH
AT_CAN_BE_CAPTURED_EN_PASSANT))==true){//king can't castle out of check
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }else{
                            //checks whether king would be in check for
either of the two squares moved
                            //checks for check for move one across
                            newBoard = board.getCopyOfBoard();//gets and
updates board with king moving across
                            newBoard[yStartingSquare][xStartingSquare]=C
hessBoardState.EMPTY_VALUE;newBoard[yStartingSquare][xStartingSquare-
1]=ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER;
                            fullMoveNo=board.FULL_MOVE_NUMBER;//updates
move number if required
                            halfMoveNo=board.HALF_MOVE_CLOCK+1;//updates
half move number
                            enPassantSquares=new int[0];//no en passant
square available as king has been moved
                            castlingRights=board.getCopyOfCastlingRights
();//stores castling rights
                            if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                                castlingRights[2]=false;castlingRights[3
]=false;
                                fullMoveNo++;
                            }else{
                                castlingRights[0]=false;castlingRights[1
]=false;
                            }
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NE
XT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){//king can't castle trhough squares where he
would be in check
                                moves.remove(i);moves.remove(i);//remove
s invalid board states from list
                                i=i-2;
                            }else{
                                newBoard[yStartingSquare][xStartingSquar
e-
1]=ChessBoardState.WHITE_MULTIPLIER*ChessBoardState.ROOK_VALUE;newBoard[
```

```
yStartingSquare][0]=ChessBoardState.EMPTY_VALUE;
                                    newBoard[yStartingSquare][xStartingSquar
e-2]=ChessBoardState.WHITE_MULTIPLIER*ChessBoardState.KING_VALUE;
                                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLA
Y_NEXT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){
                                        moves.remove(i);moves.remove(i);//re
moves invalid board states from list
                                        i=i-2;
                                    }
                                }
                            }
                        }else{//checks for moves where the king moves one
square
                            newBoard=board.getCopyOfBoard();
                            //creates a new board where king has been moved
                            newBoard[yStartingSquare][xStartingSquare]=Chess
BoardState.EMPTY_VALUE;newBoard[moves.get(i)][moves.get(xVal)]=ChessBoar
dState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER;
                            castlingRights=board.getCopyOfCastlingRights();
                            fullMoveNo=board.FULL_MOVE_NUMBER;
                            halfMoveNo=board.HALF_MOVE_CLOCK+1;
                            if(board.BOARD[moves.get(i)][moves.get(xVal)]!=C
hessBoardState.EMPTY_VALUE){//checks to see if king has made a capture
                                halfMoveNo=0;
                            }
                            enPassantSquares = new int[0];//no en passant
rights as move is made by king
                            if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                                fullMoveNo++;
                                castlingRights[2]=false;castlingRights[3]=fa
lse;
                            }else{
                                castlingRights[0]=false;castlingRights[1]=fa
lse;
                            }
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,n
ew ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){
                                moves.remove(i);moves.remove(i);//removes
invalid board states from list
                                i=i-2;
                            }
                        }
                    }
                    return moves;//returns legal moves
                }
                case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                {
                    moves=getAllLegalMovesForPawnIgnoringCheck(yStartingSqua
```

```
re, xStartingSquare, board);
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew;//stores possible en passant
squares
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){//updates move
number and en passant rights
                    fullMoveNo++;
                }
                int halfMoveNo=0;//stores half move number -will always
be zero for pawn moves
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    if(moves.get(i)==yStartingSquare+2){//checks for
black pawn moving two squares forward
                        enPassantRightsNew=new
int[]{yStartingSquare+1,xStartingSquare};
                        newBoard=board.getCopyOfBoard();
                        //adds move to board
                        newBoard[yStartingSquare][xStartingSquare]=Chess
BoardState.EMPTY_VALUE;newBoard[yStartingSquare+2][xStartingSquare]=Ches
sBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER;
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }
                    }else if(moves.get(i)==yStartingSquare-2){//checks
for white pawn moving two squares forward
                        enPassantRightsNew=new int[]{yStartingSquare-
1,xStartingSquare};
                        newBoard=board.getCopyOfBoard();
                        //adds move to board
                        newBoard[yStartingSquare][xStartingSquare]=Chess
BoardState.EMPTY_VALUE;newBoard[yStartingSquare-
2][xStartingSquare]=ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MUL
TIPLIER;
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }
```

```
                    }else if((moves.get(i)==yStartingSquare-1 &&
moves.get(xVal)==xStartingSquare)||(moves.get(i)==yStartingSquare+1 &&
moves.get(xVal)==xStartingSquare)){//checks for white or black pawn
moving one square forward
                        enPassantRightsNew= new int[0];//no en passant
rights as piece is moving one square
                        newBoard=board.getCopyOfBoard();//gets copy of
board
                        //adds move
                        newBoard[moves.get(i)][moves.get(xVal)]=ChessBoa
rdState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER;newBoard[yStartingSq
uare][xStartingSquare]=ChessBoardState.EMPTY_VALUE;
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }
                    }else{//must be a diagonal capture
                        enPassantRightsNew=new int[0];
                        if(board.BOARD[moves.get(i)][moves.get(xVal)]==C
hessBoardState.EMPTY_VALUE){//en passant capture
                            newBoard=board.getCopyOfBoard();//gets copy
of board
                            //adds move
                            newBoard[yStartingSquare][moves.get(xVal)]=C
hessBoardState.EMPTY_VALUE;
                            newBoard[yStartingSquare][xStartingSquare]=C
hessBoardState.EMPTY_VALUE;
                            newBoard[moves.get(i)][moves.get(xVal)]=Ches
sBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER;
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NE
XT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                                moves.remove(i);moves.remove(i);//remove
s invalid board states from list
                                i=i-2;
                            }
                        }else{//normal capture
                            newBoard=board.getCopyOfBoard();//gets copy
of board
                            //adds move
                            newBoard[yStartingSquare][xStartingSquare]=C
hessBoardState.EMPTY_VALUE;newBoard[moves.get(i)][moves.get(xVal)]=Chess
BoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER;
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NE
XT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){
                                moves.remove(i);moves.remove(i);//remove
```

```
s invalid board states from list
                                i=i-2;
                            }
                        }
                    }
                }
                return moves;
            }
            case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
            {
                moves=getLegalMovesForRookIgnoringCheck(yStartingSquare,
xStartingSquare, board);
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew=new int[0];//no square will be
available for en passant after rook move
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){//updates move
number and en passant rights
                    fullMoveNo++;
                    if(yStartingSquare==0&&xStartingSquare==0){
                        castlingRights[1]=false;
                    }else if(yStartingSquare==0&&xStartingSquare==7){
                        castlingRights[0]=false;
                    }
                }else{
                    if(yStartingSquare==7&&xStartingSquare==0){
                        castlingRights[3]=false;
                    }else if(yStartingSquare==7&&xStartingSquare==7){
                        castlingRights[2]=false;
                    }
                }
                int halfMoveNo;//stores half move number
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    //generates new board
                    newBoard=board.getCopyOfBoard();
                    newBoard[moves.get(i)][moves.get(xVal)]=ChessBoardSt
ate.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER;//moves castle to new
square
                    halfMoveNo=board.HALF_MOVE_CLOCK+1;
                    newBoard[yStartingSquare][xStartingSquare]=ChessBoar
dState.EMPTY_VALUE;//sets old square to empty
                    if(board.BOARD[moves.get(i)][moves.get(xVal)]!=Chess
BoardState.EMPTY_VALUE){//checks to see if rook has captured a piece
                        halfMoveNo=0;
                    }
                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT, new
```

```
ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){//checks to make sure move does not result
in own check getting in check
                        moves.remove(i);moves.remove(i);//removes
invalid board states from list
                        i=i-2;
                    }
                }
                return moves;
            }
            case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
            {
                moves =
getAllLegalMovesForKnightIgnoringCheck(yStartingSquare, xStartingSquare,
board);
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights - this wont be changed by any knight move
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew=new int[0];//no square will be
available for en passant after knight move
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                    fullMoveNo++;
                }
                int halfMoveNo;//stores half move number
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    //generates new board
                    newBoard=board.getCopyOfBoard();
                    newBoard[moves.get(i)][moves.get(xVal)]=ChessBoardSt
ate.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER;//moves knight to new
square
                    halfMoveNo=board.HALF_MOVE_CLOCK+1;
                    newBoard[yStartingSquare][xStartingSquare]=ChessBoar
dState.EMPTY_VALUE;//sets old square to empty
                    if(board.BOARD[moves.get(i)][moves.get(xVal)]!=Chess
BoardState.EMPTY_VALUE){//checks to see if knight has captured a piece
                        halfMoveNo=0;
                    }
                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){//checks to make sure move does not result
in own check getting in check
                        moves.remove(i);moves.remove(i);//removes
invalid board states from list
                        i=i-2;
                    }
```

```
                }
                return moves;
            }
            case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
            {
                moves =
getLegalMovesForBishopIgnoringCheck(yStartingSquare, xStartingSquare,
board);
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights - this wont be changed by any bishop move
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew=new int[0];//no square will be
available for en passant after bishop move
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                    fullMoveNo++;
                }
                int halfMoveNo;//stores half move number
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    //generates new board
                    newBoard=board.getCopyOfBoard();
                    newBoard[moves.get(i)][moves.get(xVal)]=ChessBoardSt
ate.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER;//moves bishop to new
square
                    halfMoveNo=board.HALF_MOVE_CLOCK+1;
                    newBoard[yStartingSquare][xStartingSquare]=ChessBoar
dState.EMPTY_VALUE;//sets old square to empty
                    if(board.BOARD[moves.get(i)][moves.get(xVal)]!=Chess
BoardState.EMPTY_VALUE){//checks to see if bishop has captured a piece
                        halfMoveNo=0;
                    }
                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){//checks to make sure move does not result
in own check getting in check
                        moves.remove(i);moves.remove(i);//removes
invalid board states from list
                        i=i-2;
                    }
                }
                return moves;
            }
            case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
            {
                moves=getLegalMovesForBishopIgnoringCheck(yStartingSquar
e, xStartingSquare, board);
```

```
                moves.addAll(getLegalMovesForRookIgnoringCheck(yStarting
Square, xStartingSquare, board));
                boolean
castlingRights[]=board.getCopyOfCastlingRights();//stores castling
rights - this wont be changed by any queen move
                int[][]newBoard;//stores copy of board after move made;
                int[]enPassantRightsNew=new int[0];//no square will be
available for en passant after queen move
                int fullMoveNo=board.FULL_MOVE_NUMBER;//stores the full
move number of the next move
                if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                    fullMoveNo++;
                }
                int halfMoveNo;//stores half move number
                for(int i=0;i<moves.size();i=i+2){//goes through all
moves
                    xVal=i+1;
                    //generates new board
                    newBoard=board.getCopyOfBoard();
                    newBoard[moves.get(i)][moves.get(xVal)]=ChessBoardSt
ate.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER;//moves queen to new
square
                    halfMoveNo=board.HALF_MOVE_CLOCK+1;
                    newBoard[yStartingSquare][xStartingSquare]=ChessBoar
dState.EMPTY_VALUE;//sets old square to empty
                    if(board.BOARD[moves.get(i)][moves.get(xVal)]!=Chess
BoardState.EMPTY_VALUE){//checks to see if queen has captured a piece
                        halfMoveNo=0;
                    }
                    if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT, new
ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantRightsNew))==true){//checks to make sure move does not result
in own check getting in check
                        moves.remove(i);moves.remove(i);//removes
invalid board states from list
                        i=i-2;
                    }
                }
                return moves;
            }
            default:
            {//king move
                moves=getAllLegalMovesForKingIgnoringCheck(yStartingSqua
re, xStartingSquare, board);
                boolean []castlingRights;//stores castling rights of new
boards;
                int[][]newBoard;//stores new board;
                int[]enPassantSquares;//stores squares for en passant
                int fullMoveNo;//stores the move number
                int halfMoveNo;//stores the half move no
                for(int i=0;i<moves.size();i=i+2){//loops through every
move, updating the board with the move and seeing whether the move would
```

```
be legal
                    xVal=i+1;
                    if(moves.get(xVal)==xStartingSquare+2 ){//checks for
kingside castling
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new
ChessBoardState(board.BOARD,board.CASTLING_RIGHTS,!board.IS_BLACK_TURN_T
O_PLAY_NEXT,board.FULL_MOVE_NUMBER,board.HALF_MOVE_CLOCK,board.SQUARE_TH
AT_CAN_BE_CAPTURED_EN_PASSANT))==true){//king can't castle out of check
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }else{
                            //checks whether king would be in check for
either of the two squares moved
                            //checks for check for move one across
                            newBoard = board.getCopyOfBoard();//gets and
updates board with king moving across
                            newBoard[yStartingSquare][xStartingSquare]=C
hessBoardState.EMPTY_VALUE;newBoard[yStartingSquare][xStartingSquare+1]=
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER;
                            fullMoveNo=board.FULL_MOVE_NUMBER;//updates
move number if required
                            halfMoveNo=board.HALF_MOVE_CLOCK+1;//updates
half move number
                            enPassantSquares=new int[0];//no en passant
square available as king has been moved
                            castlingRights=board.getCopyOfCastlingRights
();//stores castling rights
                            if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                                castlingRights[2]=false;castlingRights[3
]=false;
                                fullMoveNo++;
                            }else{
                                castlingRights[0]=false;castlingRights[1
]=false;
                            }
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NE
XT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){//king can't castle through squares where he
would be in check
                                moves.remove(i);moves.remove(i);//remove
s invalid board states from list
                                i=i-2;
                            }else{
                                newBoard[yStartingSquare][xStartingSquar
e+1]=ChessBoardState.BLACK_MULTIPLIER*ChessBoardState.ROOK_VALUE;newBoar
d[yStartingSquare][7]=ChessBoardState.EMPTY_VALUE;
                                newBoard[yStartingSquare][xStartingSquar
e+2]=ChessBoardState.BLACK_MULTIPLIER*ChessBoardState.KING_VALUE;
                                if(isInCheck(!board.IS_BLACK_TURN_TO_PLA
Y_NEXT, new ChessBoardState(newBoard, castlingRights,
```

```
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){
                                    moves.remove(i);moves.remove(i);//re
moves invalid board states from list
                                    i=i-2;
                                }
                            }
                        }
                    }else if(moves.get(xVal)==xStartingSquare-
2){//checks for queenside castling
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,
new
ChessBoardState(board.BOARD,board.CASTLING_RIGHTS,!board.IS_BLACK_TURN_T
O_PLAY_NEXT,board.FULL_MOVE_NUMBER,board.HALF_MOVE_CLOCK,board.SQUARE_TH
AT_CAN_BE_CAPTURED_EN_PASSANT))==true){//king can't castle out of check
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }else{
                            //checks whether king would be in check for
either of the two squares moved
                            //checks for check for move one across
                            newBoard = board.getCopyOfBoard();//gets and
updates board with king moving across
                            newBoard[yStartingSquare][xStartingSquare]=C
hessBoardState.EMPTY_VALUE;newBoard[yStartingSquare][xStartingSquare-
1]=ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER;
                            fullMoveNo=board.FULL_MOVE_NUMBER;//updates
move number if required
                            halfMoveNo=board.HALF_MOVE_CLOCK+1;//updates
half move number
                            enPassantSquares=new int[0];//no en passant
square available as king has been moved
                            castlingRights=board.getCopyOfCastlingRights
();//stores castling rights
                            if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                                castlingRights[2]=false;castlingRights[3
]=false;
                                fullMoveNo++;
                            }else{
                                castlingRights[0]=false;castlingRights[1
]=false;
                            }
                            if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NE
XT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){//king can't castle trhough squares where he
would be in check
                                moves.remove(i);moves.remove(i);//remove
s invalid board states from list
                                i=i-2;
                            }else{
                                newBoard[yStartingSquare][xStartingSquar
```

```
e-
1]=ChessBoardState.BLACK_MULTIPLIER*ChessBoardState.ROOK_VALUE;newBoard[
yStartingSquare][0]=ChessBoardState.EMPTY_VALUE;
                                newBoard[yStartingSquare][xStartingSquar
e-2]=ChessBoardState.BLACK_MULTIPLIER*ChessBoardState.KING_VALUE;
                                if(isInCheck(!board.IS_BLACK_TURN_TO_PLA
Y_NEXT, new ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){
                                    moves.remove(i);moves.remove(i);//re
moves invalid board states from list
                                    i=i-2;
                                }
                            }
                        }
                    }else{//checks for moves where the king moves one
square
                        newBoard=board.getCopyOfBoard();
                        //creates a new board where king has been moved
                        newBoard[yStartingSquare][xStartingSquare]=Chess
BoardState.EMPTY_VALUE;newBoard[moves.get(i)][moves.get(xVal)]=ChessBoar
dState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER;
                        castlingRights=board.getCopyOfCastlingRights();
                        fullMoveNo=board.FULL_MOVE_NUMBER;
                        halfMoveNo=board.HALF_MOVE_CLOCK+1;
                        if(board.BOARD[moves.get(i)][moves.get(xVal)]!=C
hessBoardState.EMPTY_VALUE){//checks to see if king has made a capture
                            halfMoveNo=0;
                        }
                        enPassantSquares = new int[0];//no en passant
rights as move is made by king
                        if(board.IS_BLACK_TURN_TO_PLAY_NEXT){
                            fullMoveNo++;
                            castlingRights[2]=false;castlingRights[3]=fa
lse;
                        }else{
                            castlingRights[0]=false;castlingRights[1]=fa
lse;
                        }
                        if(isInCheck(!board.IS_BLACK_TURN_TO_PLAY_NEXT,n
ew ChessBoardState(newBoard, castlingRights,
!board.IS_BLACK_TURN_TO_PLAY_NEXT, fullMoveNo, halfMoveNo,
enPassantSquares))==true){
                            moves.remove(i);moves.remove(i);//removes
invalid board states from list
                            i=i-2;
                        }
                    }
                }
                return moves;//returns legal moves
            }
        }
    }
```

```
    /**
     * Checks to see if king of input colour is in check in the current
state. This method is used as part of the process to obtain all legal
moves.
     * Must be called on the correct king given who is about to move.
     * @param isKingWhite The colour of the king to check
     * @param state The board state to check
     * @return True is returned if the king chosen is in check and false
is returned otherwise
     */
    public static boolean isInCheck (boolean isKingWhite,ChessBoardState
state){
        //checks to see whether the king square is being targeted by
pieces of the opposing colour
        int yCoordKing=-1;int xCoordKing=-1;//stores the coordinate of
the king
        ArrayList<Integer> allAvailableMoves=new
ArrayList<>(70);//stores any array list storing all legal moves
        //int[][]allAvailableMoves = new int[0][2];//stores all moves
that can be got from a square
        if(isKingWhite){
            for(int y=0;y<8;y++){//loops through board to find king and
get all available moves from other colour
                for(int x=0;x<8;x++){
                    switch (state.BOARD[y][x]) {
                        case
ChessBoardState.PAWN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                            allAvailableMoves.addAll(getAllLegalMovesFor
PawnIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.ROOK_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                            allAvailableMoves.addAll(getLegalMovesForRoo
kIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                            allAvailableMoves.addAll(getAllLegalMovesFor
KnightIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.BISHOP_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                            allAvailableMoves.addAll(getLegalMovesForBis
hopIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.QUEEN_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                            allAvailableMoves.addAll(getLegalMovesForRoo
kIgnoringCheck(y, x,
state));allAvailableMoves.addAll(getLegalMovesForBishopIgnoringCheck(y,
x, state));
                            break;
                        case
```

```
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                            allAvailableMoves.addAll(getAllLegalMovesFor
KingIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                            yCoordKing=y;
                            xCoordKing=x;
                            break;
                        default:
                            break;
                    }
                }
            }

        }else{
            for(int y=0;y<8;y++){//loops through board to find king and
get all available moves from other colour
                for(int x=0;x<8;x++){
                    switch (state.BOARD[y][x]) {
                        case
ChessBoardState.PAWN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                            allAvailableMoves.addAll(getAllLegalMovesFor
PawnIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.ROOK_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                            allAvailableMoves.addAll(getLegalMovesForRoo
kIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.KNIGHT_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                            allAvailableMoves.addAll(getAllLegalMovesFor
KnightIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.BISHOP_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                            allAvailableMoves.addAll(getLegalMovesForBis
hopIgnoringCheck(y, x, state));
                            break;
                        case
ChessBoardState.QUEEN_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                            allAvailableMoves.addAll(getLegalMovesForRoo
kIgnoringCheck(y, x,
state));allAvailableMoves.addAll(getLegalMovesForBishopIgnoringCheck(y,
x, state));
                            break;
                        case
ChessBoardState.KING_VALUE*ChessBoardState.WHITE_MULTIPLIER:
                            allAvailableMoves.addAll(getAllLegalMovesFor
KingIgnoringCheck(y, x, state));
                            break;
                        case
```

```
ChessBoardState.KING_VALUE*ChessBoardState.BLACK_MULTIPLIER:
                            yCoordKing=y;
                            xCoordKing=x;
                            break;
                        default:
                            break;
                    }
                }
            }
        }
        for(int
move=0;move<allAvailableMoves.size();move=move+2){//loops through all
moves to see if king is in check
            if(allAvailableMoves.get(move)==yCoordKing &&
allAvailableMoves.get(move+1)==xCoordKing){
                return true;
            }
        }
        return false;
    }
    /**
     * Gets all legal moves for king in square. Note king must be of
correct colour. Castling is represented as the king moving two squares
to the left or right.
     * NOTE: You can't castle out of check but as this method ignores
check, this will need to be handled.
     * @param ySquare The y index of the king
     * @param xSquare The x index of the king
     * @param state The board state
     * @return The moves
     */
    private static ArrayList <Integer>
getAllLegalMovesForKingIgnoringCheck(int ySquare, int xSquare
,ChessBoardState state){
        ArrayList<Integer>moves = new ArrayList<>();//array list storing
all moves. Y then x moves are added alternatingly
        if(state.IS_BLACK_TURN_TO_PLAY_NEXT){//if the king is black
            //checks for move one square up
            if(ySquare!=0){
                if(state.BOARD[ySquare-1][xSquare]>=0){//checks if
square is empty or of opposite colour
                    //moves=addToEndOfArray(moves, new int[]{ySquare-
1,xSquare});
                    moves.add(ySquare-1);moves.add(xSquare);
                }
            }
            //checks for one move down
            if(ySquare!=7){
                if(state.BOARD[ySquare+1][xSquare]>=0){//checks if
square is empty or of opposite colour
                    //moves=addToEndOfArray(moves, new
int[]{ySquare+1,xSquare});
                    moves.add(ySquare+1);moves.add(xSquare);
```

```
                }
            }
            //checks for one move to right
            if(xSquare!=7){
                if(state.BOARD[ySquare][xSquare+1]>=0){//checks if
square is empty or of opposite colour
                   // moves=addToEndOfArray(moves, new
int[]{ySquare,xSquare+1});
                    moves.add(ySquare);moves.add(xSquare+1);
                }
                if(ySquare!=0 && state.BOARD[ySquare-
1][xSquare+1]>=0){//checks for right and up
                    //moves=addToEndOfArray(moves, new int[]{ySquare-
1,xSquare+1});
                    moves.add(ySquare-1);moves.add(xSquare+1);
                }
                if(ySquare!=7 &&
state.BOARD[ySquare+1][xSquare+1]>=0){//checks for right and down
                    //moves=addToEndOfArray(moves, new
int[]{ySquare+1,xSquare+1});
                    moves.add(ySquare+1);moves.add(xSquare+1);
                }
            }
            //checks for one move to left
            if(xSquare!=0){
                if(state.BOARD[ySquare][xSquare-1]>=0){//checks if
square is empty or of opposite colour
                   // moves=addToEndOfArray(moves, new
int[]{ySquare,xSquare-1});
                    moves.add(ySquare);moves.add(xSquare-1);
                }
                if(ySquare!=0 && state.BOARD[ySquare-1][xSquare-
1]>=0){//checks for left and up
                    //moves=addToEndOfArray(moves, new int[]{ySquare-
1,xSquare-1});
                    moves.add(ySquare-1);moves.add(xSquare-1);
                }
                if(ySquare!=7 && state.BOARD[ySquare+1][xSquare-
1]>=0){//checks for left and down
                    //moves=addToEndOfArray(moves, new
int[]{ySquare+1,xSquare-1});
                    moves.add(ySquare+1);moves.add(xSquare-1);
                }
            }
            //checks for kingside castling
            if(state.CASTLING_RIGHTS[2]&&state.BOARD[0][5]==ChessBoardSt
ate.EMPTY_VALUE&&state.BOARD[0][6]==ChessBoardState.EMPTY_VALUE){
                //moves=addToEndOfArray(moves, new
int[]{ySquare,xSquare+2});
                moves.add(ySquare);moves.add(xSquare+2);
            }
            //checks for queenside castling
            if(state.CASTLING_RIGHTS[3]&&state.BOARD[0][3]==ChessBoardSt
```

```
ate.EMPTY_VALUE&&state.BOARD[0][2]==ChessBoardState.EMPTY_VALUE&&state.B
OARD[0][1]==ChessBoardState.EMPTY_VALUE){
                //moves=addToEndOfArray(moves, new
int[]{ySquare,xSquare-2});
                moves.add(ySquare);moves.add(xSquare-2);
            }
        }else{//if the king is white
            //checks for move one square up
            if(ySquare!=0){
                if(state.BOARD[ySquare-1][xSquare]<=0){//checks if
square is empty or of opposite colour
                    //moves=addToEndOfArray(moves, new int[]{ySquare-
1,xSquare});
                    moves.add(ySquare-1);moves.add(xSquare);
                }
            }
            //checks for one move down
            if(ySquare!=7){
                if(state.BOARD[ySquare+1][xSquare]<=0){//checks if
square is empty or of opposite colour
                    //moves=addToEndOfArray(moves, new
int[]{ySquare+1,xSquare});
                    moves.add(ySquare+1);moves.add(xSquare);
                }
            }
            //checks for one move to right
            if(xSquare!=7){
                if(state.BOARD[ySquare][xSquare+1]<=0){//checks if
square is empty or of opposite colour
                    //moves=addToEndOfArray(moves, new
int[]{ySquare,xSquare+1});
                    moves.add(ySquare);moves.add(xSquare+1);
                }
                if(ySquare!=0 && state.BOARD[ySquare-
1][xSquare+1]<=0){//checks for right and up
                    //moves=addToEndOfArray(moves, new int[]{ySquare-
1,xSquare+1});
                    moves.add(ySquare-1);moves.add(xSquare+1);
                }
                if(ySquare!=7 &&
state.BOARD[ySquare+1][xSquare+1]<=0){//checks for right and down
                    //moves=addToEndOfArray(moves, new
int[]{ySquare+1,xSquare+1});
                    moves.add(ySquare+1);moves.add(xSquare+1);
                }
            }
            //checks for one move to left
            if(xSquare!=0){
                if(state.BOARD[ySquare][xSquare-1]<=0){//checks if
square is empty or of opposite colour
                    //moves=addToEndOfArray(moves, new
int[]{ySquare,xSquare-1});
                    moves.add(ySquare);moves.add(xSquare-1);
```

```
                }
                if(ySquare!=0 && state.BOARD[ySquare-1][xSquare-
1]<=0){//checks for left and up
                    //moves=addToEndOfArray(moves, new int[]{ySquare-
1,xSquare-1});
                    moves.add(ySquare-1);moves.add(xSquare-1);
                }
                if(ySquare!=7 && state.BOARD[ySquare+1][xSquare-
1]<=0){//checks for left and down
                    //moves=addToEndOfArray(moves, new
int[]{ySquare+1,xSquare-1});
                    moves.add(ySquare+1);moves.add(xSquare-1);
                }
            }
            //checks for kingside castling
            if(state.CASTLING_RIGHTS[0]&&state.BOARD[7][5]==ChessBoardSt
ate.EMPTY_VALUE&&state.BOARD[7][6]==ChessBoardState.EMPTY_VALUE){
                //moves=addToEndOfArray(moves, new
int[]{ySquare,xSquare+2});
                moves.add(ySquare);moves.add(xSquare+2);
            }
            //checks for queenside castling
            if(state.CASTLING_RIGHTS[1]&&state.BOARD[7][3]==ChessBoardSt
ate.EMPTY_VALUE&&state.BOARD[7][2]==ChessBoardState.EMPTY_VALUE&&state.B
OARD[7][1]==ChessBoardState.EMPTY_VALUE){
                //moves=addToEndOfArray(moves, new
int[]{ySquare,xSquare-2});
                moves.add(ySquare);moves.add(xSquare-2);
            }
        }
        return moves;
    }
    /**
     * Gets all legal moves for pawn at the square input. Note check is
ignored and pawn promotion is treated as the pawn
     * sitting on the final rank. En passant is supported but must be
checked carefully. Note: must be called on pawn of the right colour
given the board state
     * @param ySquare The y index of the pawn
     * @param xSquare The x index of the pawn
     * @param state The board state
     * @return All moves
     */
    private static ArrayList <Integer>
getAllLegalMovesForPawnIgnoringCheck(int ySquare, int xSquare,
ChessBoardState state){
        ArrayList<Integer> moves = new ArrayList<>(8);
        //checks to see if pawn can move forward twice
        if(state.IS_BLACK_TURN_TO_PLAY_NEXT&&ySquare==1){//checks to see
if black pawn can move forward two moves
            if(state.BOARD[ySquare+2][xSquare]==ChessBoardState.EMPTY_VA
LUE&& state.BOARD[ySquare+1][xSquare]==ChessBoardState.EMPTY_VALUE){
                moves.add(ySquare+2);moves.add(xSquare);
```

```
            }
        }else if(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&ySquare==6){//checks
to see if white can move forward two
            if(state.BOARD[ySquare-
2][xSquare]==ChessBoardState.EMPTY_VALUE && state.BOARD[ySquare-
1][xSquare]==ChessBoardState.EMPTY_VALUE){
                moves.add(ySquare-2);moves.add(xSquare);
            }
        }
        //checks to see if pawn can move forward once
        if(state.IS_BLACK_TURN_TO_PLAY_NEXT&&ySquare!=7){//the pawn is
black
            if(state.BOARD[ySquare+1][xSquare]==ChessBoardState.EMPTY_VA
LUE){//checks to move forward once
                moves.add(ySquare+1);moves.add(xSquare);
            }
            if(xSquare!=7){//checks for diagonal capture to the right as
black pawn
                if(state.BOARD[ySquare+1][xSquare+1]>0){//checks to see
if there is a white piece on the diagnoal
                    moves.add(ySquare+1);moves.add(xSquare+1);
                }
            }
            if(xSquare!=0){//checks for left diagnoal capture
                if(state.BOARD[ySquare+1][xSquare-1]>0){//checks to see
if white piece is on diagonal
                    moves.add(ySquare+1);moves.add(xSquare-1);
                }
            }
        }else if(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&ySquare!=0){
            if(state.BOARD[ySquare-
1][xSquare]==ChessBoardState.EMPTY_VALUE){
                moves.add(ySquare-1);moves.add(xSquare);
            }
            if(xSquare!=7){//checks for diagonal capture to the right as
white pawn
                if(state.BOARD[ySquare-1][xSquare+1]<0){//checks to see
if there is a black piece on the diagnoal
                    moves.add(ySquare-1);moves.add(xSquare+1);
                }
            }
            if(xSquare!=0){//checks for left diagnoal capture
                if(state.BOARD[ySquare-1][xSquare-1]<0){//checks to see
if black piece is on diagonal
                    moves.add(ySquare-1);moves.add(xSquare-1);
                }
            }
        }
        //checks for en passant captures
        if(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length!=0){//che
cks for en passant
            if(state.IS_BLACK_TURN_TO_PLAY_NEXT){//checks for en passant
captures that can be made by black pawn
```

```
                if((state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[0]==ySq
uare+1) && ((state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]==xSquare-
1)||(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]==xSquare+1))){
                    moves.add(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSA
NT[0]);moves.add(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]);
                }
            }else{
                if((state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[0]==ySq
uare-1) && ((state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]==xSquare-
1)||(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]==xSquare+1))){
                    moves.add(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSA
NT[0]);moves.add(state.SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]);
                }
            }
        }
        return moves;
    }
    /**
     * Adds new records to the end of an array
     * @param currentArray The current array
     * @param newRecords The new records
     * @param arrayWidth The array width
     * @return The new array
     */
    private static
int[][]addToEndOfArray(int[][]currentArray,int[][]newRecords, int
arrayWidth){
        int[][]newArray=new
int[currentArray.length+newRecords.length][arrayWidth];
        for(int i=0;i<currentArray.length;i++){
            for(int x=0;x<arrayWidth;x++){
                newArray[i][x]=currentArray[i][x];
            }
        }
        for(int i=0;i<newRecords.length;i++){
            for(int x=0;x<arrayWidth;x++){
                newArray[i+currentArray.length][x]=newRecords[i][x];
            }
        }
        return newArray;
    }
    /**
     * Adds new record to two 2d array. This record is appended to the
bottom of the new array. Note newRecord must be of the same width as the
current array
     * @param currentArray the array
     * @param newRecord the record to add
     * @return the new array
     */
    private static int[][]
addToEndOfArray(int[][]currentArray,int[]newRecord){
        int[][]newArray=new
int[currentArray.length+1][newRecord.length];//creates new array one
```

```
longer than last one
        for(int i=0;i<newRecord.length;i++){//adds new record to end of
new array
            newArray[currentArray.length][i]=newRecord[i];
        }
        for(int i=0;i<currentArray.length;i++){//copies old array items
to new array
            for(int x=0;x<newRecord.length;x++){
                newArray[i][x]=currentArray[i][x];
            }
        }
        return newArray;//returns new array
    }
    /**
     * Gets all legal moves for a knight ignoring check. Note knight in
square MUST be of the same colour as the colour that moves next
     * @param ySquare The y coordinate of the knight
     * @param xSquare The x coordinate of the knight
     * @param state The board state
     * @return All moves
     */
    private static ArrayList <Integer>
getAllLegalMovesForKnightIgnoringCheck(int ySquare,int xSquare,
ChessBoardState state){
        ArrayList<Integer>moves=new ArrayList<>();
        /* 8 squares to check
        y+2 x+1,y+1 x+2,y-1 x+2,y-2 x+1,y-2 x-1,y-1 x-2,y+1 x-2,y+2 x-2
        */
        //check for y+1 and y+2 moves
        if(ySquare<=6){//check if y+1 is a legal square
            if(xSquare<=5){//checking for Y+1 and x+2
                if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquar
e+1][xSquare+2]>=0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySq
uare+1][xSquare+2]<=0)){
                    moves.add(ySquare+1);moves.add(xSquare+2);
                }
            }
            if(xSquare>=2){//checking for y+1 and x-2
                if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquar
e+1][xSquare-
2]>=0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+1][xSqua
re-2]<=0)){
                    moves.add(ySquare+1);moves.add(xSquare-2);
                }
            }
            if(ySquare<=5){//checking for y+2
                if(xSquare>=1){//checking for Y+2 and x-1
                    if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[yS
quare+2][xSquare-
1]>=0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+2][xSqua
re-1]<=0)){
                        moves.add(ySquare+2);moves.add(xSquare-1);
                    }
```

```
                }
                if(xSquare<=6){//checking for y+2 and x+1
                    if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+2][xSquare+1]>=0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+2][xSquare+1]<=0)){
                        moves.add(ySquare+2);moves.add(xSquare+1);
                    }
                }
            }
        }
        if(ySquare>=1){//checking for y-1
            if(xSquare<=5){//checking for Y-1 and x+2
                if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-1][xSquare+2]>=0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-1][xSquare+2]<=0)){
                    moves.add(ySquare-1);moves.add(xSquare+2);
                }
            }
            if(xSquare>=2){//checking for y-1 and x-2
                if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-1][xSquare-2]>=0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-1][xSquare-2]<=0)){
                    moves.add(ySquare-1);moves.add(xSquare-2);
                }
            }
            if(ySquare>=2){//checking for y-2
                if(xSquare>=1){//checking for Y-2 and x-1
                    if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-2][xSquare-1]>=0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-2][xSquare-1]<=0)){
                        moves.add(ySquare-2);moves.add(xSquare-1);
                    }
                }
                if(xSquare<=6){//checking for y-2 and x+1
                    if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-2][xSquare+1]>=0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-2][xSquare+1]<=0)){
                        moves.add(ySquare-2);moves.add(xSquare+1);
                    }
                }
            }
        }
        return moves;
    }
    /**
     * Gets all legal moves for rook (or queen) in position, ignoring check. This method must be called in a piece of the right colour
     * @param ySquare The y coordinate of the rook (or queen).
     * @param xSquare The x coordinate of the rook (or queen).
```

```
     * @param state The board state
     * @return An ArrayList containing all legal moves of width 2. If no
moves are found, the ArrayList is of length 0.
     */
    private static ArrayList <Integer>
getLegalMovesForRookIgnoringCheck(int ySquare, int xSquare,
ChessBoardState state){
        ArrayList<Integer> moves = new ArrayList<>();//stores all legal
moves for the rook
        //checks for moves up
        for(int inc=1;inc<=ySquare;inc++){
            if(state.BOARD[ySquare-
inc][xSquare]==ChessBoardState.EMPTY_VALUE){//if square is empty
                moves.add(ySquare-inc);moves.add(xSquare);
            }else
if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-
inc][xSquare]<0) || (!state.IS_BLACK_TURN_TO_PLAY_NEXT &&
state.BOARD[ySquare-inc][xSquare]>0)){//if collides with piece of same
colour
                break;
            }else{//if collides with piece of different colours
                moves.add(ySquare-inc);moves.add(xSquare);
                break;
            }
        }
        //checks for moves down
        for(int inc=1;inc<=7-ySquare;inc++){
            if(state.BOARD[ySquare+inc][xSquare]==ChessBoardState.EMPTY_
VALUE){//if square is empty
                moves.add(ySquare+inc);moves.add(xSquare);
            }else
if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+inc][xSquare]<
0) || (!state.IS_BLACK_TURN_TO_PLAY_NEXT &&
state.BOARD[ySquare+inc][xSquare]>0)){//if collides with piece of same
colour
                break;
            }else{//if collides with piece of different colours
                moves.add(ySquare+inc);moves.add(xSquare);
                break;
            }
        }
        //checks for moves to right
        for(int inc=1;inc<=7-xSquare;inc++){
            if(state.BOARD[ySquare][xSquare+inc]==ChessBoardState.EMPTY_
VALUE){//if square is empty
                moves.add(ySquare);moves.add(xSquare+inc);
            }else
if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare][xSquare+inc]<
0) || (!state.IS_BLACK_TURN_TO_PLAY_NEXT &&
state.BOARD[ySquare][xSquare+inc]>0)){//if collides with piece of same
colour
                break;
            }else{//if collides with piece of different colours
```

```
                moves.add(ySquare);moves.add(xSquare+inc);
                break;
            }
        }
        //checks for moves to left
        for(int inc=1;inc<=xSquare;inc++){
            if(state.BOARD[ySquare][xSquare-
inc]==ChessBoardState.EMPTY_VALUE){//if square is empty
                moves.add(ySquare);moves.add(xSquare-inc);
            }else
if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare][xSquare-
inc]<0) || (!state.IS_BLACK_TURN_TO_PLAY_NEXT &&
state.BOARD[ySquare][xSquare-inc]>0)){//if collides with piece of same
colour
                break;
            }else{//if collides with piece of different colours
                moves.add(ySquare);moves.add(xSquare-inc);
                break;
            }
        }
        return moves;
    }
    /**
     * Gets all legal moves for the bishop from the current state,
ignoring whether the king is in check. This method must be called on a
bishop (or queen) of the correct colour.
     * @param ySquare The y index of the bishop (or queen)
     * @param xSquare The x index of the bishop (or queen)
     * @param state The chess board state to get the legal moves from
     * @return An ArrayList containing all legal moves of width 2. If no
moves are found, the ArrayList is of length 0.
     */
    private static ArrayList <Integer>
getLegalMovesForBishopIgnoringCheck(int ySquare, int xSquare,
ChessBoardState state){
        ArrayList<Integer> moves = new ArrayList<>();
        //check for moving down and to the right
        for(int inc=1;inc<=7-Math.max(ySquare, xSquare);inc++){
            if(state.BOARD[ySquare+inc][xSquare+inc]==ChessBoardState.EM
PTY_VALUE){
                moves.add(ySquare+inc);moves.add(xSquare+inc);
            }else
if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+inc][xSquare+i
nc]<0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+inc][xSq
uare+inc]>0)){//if it collides with own piece
                break;
            }else{//collides with enemy piece
                moves.add(ySquare+inc);moves.add(xSquare+inc);
                break;
            }
        }
        //checking for move down and left
        for(int inc=1;inc<=Math.min((7-ySquare),xSquare);inc++){
```

```
            if(state.BOARD[ySquare+inc][xSquare-
inc]==ChessBoardState.EMPTY_VALUE){
                moves.add(ySquare+inc);moves.add(xSquare-inc);
            }else
if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+inc][xSquare-
inc]<0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare+inc][xS
quare-inc]>0)){//if it collides with own piece
                break;
            }else{//collides with enemy piece
               moves.add(ySquare+inc);moves.add(xSquare-inc);
                break;
            }
        }
        //checking for move up and right
        for(int inc=1;inc<=Math.min(ySquare,(7-xSquare));inc++){
            if(state.BOARD[ySquare-
inc][xSquare+inc]==ChessBoardState.EMPTY_VALUE){
                moves.add(ySquare-inc);moves.add(xSquare+inc);
            }else
if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-
inc][xSquare+inc]<0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[yS
quare-inc][xSquare+inc]>0)){//if it collides with own piece
                break;
            }else{//collides with enemy piece
                moves.add(ySquare-inc);moves.add(xSquare+inc);
                break;
            }
        }
        //checking for up and left
        for(int inc=1;inc<=Math.min(ySquare,xSquare);inc++){
            if(state.BOARD[ySquare-inc][xSquare-
inc]==ChessBoardState.EMPTY_VALUE){
               moves.add(ySquare-inc);moves.add(xSquare-inc);
            }else
if((state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-inc][xSquare-
inc]<0)||(!state.IS_BLACK_TURN_TO_PLAY_NEXT&&state.BOARD[ySquare-
inc][xSquare-inc]>0)){//if it collides with own piece
                break;
            }else{//collides with enemy piece
                moves.add(ySquare-inc);moves.add(xSquare-inc);
                break;
            }
        }
        return moves;
    }
    /**
     * Returns the current board state.
     * @return The current board state
     */
    public ChessBoardState getCurrentState(){
        return states[states.length-1];
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

/**
 * This is a neural network trained by me to play chess.
 * @author mortimer
 */
public class ChessNeuralNetwork {
    private static final String
BEST_NET_FILE_NAME="bestNet.txt";//stores the location of the text file
that stores the best network architecture
    private final double[] ALL_WEIGHTS_AND_BIASES;//stores all weights
and biases of the neural network
    private final int[]NETWORK_STRUCTURE;//stores the network structure
    private final int NET_LENGTH;//stores the length of the net
architecture
    private final double[] NET_ACTIVATIONS;//stores the network
activation values
    /**
     * Loads the neural network stored in the "bestNet.txt" file.
     * @param NETWORK_STRUCTURE The network structure of the neural
network
     * @throws java.io.IOException An IOException may be thrown when
reading the values of the network's weights and biases from
"bestNet.txt"
     */
    public ChessNeuralNetwork(final int[] NETWORK_STRUCTURE)throws
IOException{
        this.NETWORK_STRUCTURE= NETWORK_STRUCTURE;
        NET_LENGTH=NETWORK_STRUCTURE.length;
        int totNeurons=NETWORK_STRUCTURE[0]+NETWORK_STRUCTURE[1];
        int
noOfWeightsAndBiases=NETWORK_STRUCTURE[0]*NETWORK_STRUCTURE[1]+NETWORK_S
TRUCTURE[1];//stores the number of weights and biases in the network
        for(int i=2;i<NETWORK_STRUCTURE.length;i++){//calculates the
total number of weights and biases
            noOfWeightsAndBiases=noOfWeightsAndBiases+NETWORK_STRUCTURE[
i]*NETWORK_STRUCTURE[i-1]+NETWORK_STRUCTURE[i];
            totNeurons=totNeurons+NETWORK_STRUCTURE[i];
```

```
        }
        if(NETWORK_STRUCTURE.length>2){
            totNeurons=totNeurons-NETWORK_STRUCTURE[NET_LENGTH-1];
        }
        NET_ACTIVATIONS=new double[totNeurons];
        ALL_WEIGHTS_AND_BIASES=new
double[noOfWeightsAndBiases];//creates array storing all weights and
biases
        //reads network architecture from text file
        FileReader read = new FileReader(BEST_NET_FILE_NAME);
        int i=0;//stores the index of all weights and biases to be
accessed
        BufferedReader buffRead = new BufferedReader(read);
        String lineRead;//stores the line read
        while((lineRead=buffRead.readLine())!=null){//loads values of
neural network
            ALL_WEIGHTS_AND_BIASES[i]=Double.parseDouble(lineRead);
            i++;
        }
        buffRead.close();read.close();
    }
    /**
     * Feeds data through the network and returns the network's output
     * @param input The input
     * @return The output
     */
    public double[] feedThroughNet(int[]input){
        int currentIndexForAll=0;//stores the current index for all
weights and biases
        int activsIndex=0;
        double out[]=new double[NETWORK_STRUCTURE[NET_LENGTH-1]];
        int startOfPrevLayer=0;
        for(;activsIndex<input.length;activsIndex++){
           NET_ACTIVATIONS[activsIndex]=input[activsIndex];
        }
        double sum;//stores the sum that amounts to the input to the
node
        for(int layerIndex=1;layerIndex<NET_LENGTH-1;layerIndex++){
            for(int
neuronInLayer=0;neuronInLayer<NETWORK_STRUCTURE[layerIndex];neuronInLaye
r++){//loops through every neuron in current layer
                sum=ALL_WEIGHTS_AND_BIASES[currentIndexForAll];//sets
the sum equal to the bias
                currentIndexForAll++;
                for(int
neuronInPrevLayer=0;neuronInPrevLayer<NETWORK_STRUCTURE[layerIndex-
1];neuronInPrevLayer++){//sums weights in previous layer to calculate
input value to node
                    sum=sum+NET_ACTIVATIONS[startOfPrevLayer+neuronInPre
vLayer]*ALL_WEIGHTS_AND_BIASES[currentIndexForAll];
                    currentIndexForAll++;
                }
                NET_ACTIVATIONS[activsIndex]=reLu(sum);
```

```
                activsIndex++;
            }
            startOfPrevLayer=startOfPrevLayer+NETWORK_STRUCTURE[layerInd
ex-1];
        }
        //calculates output
        for(int
neuronInLayer=0;neuronInLayer<NETWORK_STRUCTURE[NET_LENGTH-
1];neuronInLayer++){//loops through every neuron in output layer
            sum=ALL_WEIGHTS_AND_BIASES[currentIndexForAll];//sets the
sum equal to the bias
            currentIndexForAll++;
            for(int
neuronInPrevLayer=0;neuronInPrevLayer<NETWORK_STRUCTURE[NET_LENGTH-
2];neuronInPrevLayer++){//sums weights in previous layer to calculate
input value to node
                sum=sum+NET_ACTIVATIONS[startOfPrevLayer+neuronInPrevLay
er]*ALL_WEIGHTS_AND_BIASES[currentIndexForAll];
                currentIndexForAll++;
            }
            out[neuronInLayer]=sum;
        }
        softmax(out);
        return out;
    }
    /**
     * Executes the reLu function
     * @param x The input
     * @return The output
     */
    private static double reLu(double x){
        if(x>0.0){
            return x;
        }else{
            return 0.0;
        }
    }
    /**
     * Performs the softmax function on the inputted array
     * @param input The array to perform the softmax function on
     */
    private static void softmax(double[]input){
        input[0]=Math.exp(input[0]);
        double sum=input[0];//stores the sum of the exponentials
        for(int i=1;i<input.length;i++){
            input[i]=Math.exp(input[i]);
            sum=sum+input[i];
        }
        for(int i=0;i<input.length;i++){
            input[i]=input[i]/sum;
        }
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.Component;
import java.awt.Container;
import java.awt.FocusTraversalPolicy;
import javax.swing.JOptionPane;
/**
 * Creates a form that allows the user to create an account.
 * @author mortimer
 */
public class CreateAccount extends javax.swing.JFrame {
    private boolean viewPass=false;//variable that stores whether the
password is visible or masked
    private boolean viewConfirmPass=false;//variable that stores whether
the confirm password field is visible or masked
    private final char DEFAULT_ECHO_CHAR;//stores the character used to
mask the password
    private final Component[]COMPS;//array used to determine focus
traversal policy
    private int currentCompIndex=0;//stores the index used to focus the
components
    /**
     * Creates new form CreateAccount
     */
    public CreateAccount() {
        initComponents();
        //array that stores which order components should be focused
        Component[]compsBuff={usernameTxt,emailAddressTxt,passwordField,
confirmPasswordField,firstNameTxt,surnameTxt,postcodeTxt,addressLineOneT
xt,addressLineTwoTxt,addressLineThreeTxt,addressLineFourTxt,discordUsern
ameTxt,discordNumberTxt,dateOfBirthTxt,phoneNumberTxt,createAccountBtn,c
learBtn,logInBtn};
        COMPS=compsBuff;
        DEFAULT_ECHO_CHAR=passwordField.getEchoChar();
        //sets the focus traversal policy. Essentially, the traversal
policy is that the focus is assigned to each element of the COMPS array
in the order they are in the array.
        this.setFocusTraversalPolicy(new FocusTraversalPolicy() {
            /**
             * Gets the component to focus next.
             * @param aContainer The current container
             * @param aComponent The current component
             * @return The component to focus next.
             */
```

```
            @Override
            public Component getComponentAfter(Container aContainer,
Component aComponent) {
                currentCompIndex++;
                if(currentCompIndex>COMPS.length-1){//loops the array
back to element 0 if the index exceeds the curent value.
                    currentCompIndex=0;
                }
                return COMPS[currentCompIndex];//returns the component
            }
            /**
             * Gets the component before the focused component in the
focus traversal policy.
             * @param aContainer The current container
             * @param aComponent The current component
             * @return The component before the focused component in the
focus traversal policy
             */
            @Override
            public Component getComponentBefore(Container aContainer,
Component aComponent) {
                currentCompIndex--;
                if(currentCompIndex<0){//wraps the array so that it goes
to the end
                    currentCompIndex=COMPS.length-1;
                }
                return COMPS[currentCompIndex];
            }
            /**
             * Gets the first component in the focus policy
             * @param aContainer The container
             * @return The first component to focus
             */
            @Override
            public Component getFirstComponent(Container aContainer) {
                return COMPS[0];
            }
            /**
             * Gets the last component in the focus traversal policy
             * @param aContainer The container
             * @return The last component in the focus traversal policy
             */
            @Override
            public Component getLastComponent(Container aContainer) {
                return COMPS[COMPS.length-1];
            }
            /**
             * Gets the default component
             * @param aContainer The container
             * @return The default component
             */
            @Override
            public Component getDefaultComponent(Container aContainer) {
```

```
                return COMPS[0];
            }
        });
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        jLabel16 = new javax.swing.JLabel();
        jLabel17 = new javax.swing.JLabel();
        jLabel18 = new javax.swing.JLabel();
        jLabel19 = new javax.swing.JLabel();
        usernameTxt = new javax.swing.JTextField();
        emailAddressTxt = new javax.swing.JTextField();
        passwordField = new javax.swing.JPasswordField();
        confirmPasswordField = new javax.swing.JPasswordField();
        firstNameTxt = new javax.swing.JTextField();
        surnameTxt = new javax.swing.JTextField();
        postcodeTxt = new javax.swing.JTextField();
        addressLineOneTxt = new javax.swing.JTextField();
        addressLineTwoTxt = new javax.swing.JTextField();
        addressLineThreeTxt = new javax.swing.JTextField();
        addressLineFourTxt = new javax.swing.JTextField();
        discordUsernameTxt = new javax.swing.JTextField();
        discordNumberTxt = new javax.swing.JTextField();
        dateOfBirthTxt = new javax.swing.JTextField();
        phoneNumberTxt = new javax.swing.JTextField();
        logInBtn = new javax.swing.JButton();
        clearBtn = new javax.swing.JButton();
```

```
        createAccountBtn = new javax.swing.JButton();
        passwordLabel = new javax.swing.JLabel();
        confirmPasswordLabel = new javax.swing.JLabel();
        firstNameLabel = new javax.swing.JLabel();
        surnameLabel = new javax.swing.JLabel();
        postcodeLabel = new javax.swing.JLabel();
        addressOneLabel = new javax.swing.JLabel();
        addressTwoLabel = new javax.swing.JLabel();
        addressThreeLabel = new javax.swing.JLabel();
        addressFourLabel = new javax.swing.JLabel();
        discordUsernameLabel = new javax.swing.JLabel();
        discordNumberLabel = new javax.swing.JLabel();
        dateOfBirthLabel = new javax.swing.JLabel();
        phoneNumberLabel = new javax.swing.JLabel();
        viewPassBtn = new javax.swing.JButton();
        viewConfirmBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Create Account");

        jLabel1.setText("Username");

        jLabel2.setText("Email Address");

        jLabel3.setText("Password");

        jLabel4.setText("Confirm Password");

        jLabel5.setText("First Name");

        jLabel6.setText("Surname");

        jLabel7.setText("Postcode");

        jLabel8.setText("Address Line 1");

        jLabel9.setText("Address Line 2");

        jLabel10.setText("Address Line 3");

        jLabel11.setText("Address Line 4");

        jLabel12.setText("Discord Username");

        jLabel13.setText("Discord Number");

        jLabel14.setText("Date of Birth");

        jLabel15.setText("Phone Number");

        jLabel16.setForeground(new java.awt.Color(255, 0, 51));
        jLabel16.setText("*");
```

```
        jLabel17.setForeground(new java.awt.Color(255, 0, 51));
        jLabel17.setText("*");

        jLabel18.setForeground(new java.awt.Color(255, 0, 51));
        jLabel18.setText("*");

        jLabel19.setForeground(new java.awt.Color(255, 0, 51));
        jLabel19.setText("*");

        usernameTxt.setFocusCycleRoot(true);
        usernameTxt.addFocusListener(new java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                usernameTxtFocusLost(evt);
            }
        });
        usernameTxt.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyTyped(java.awt.event.KeyEvent evt) {
                usernameTxtKeyTyped(evt);
            }
        });

        emailAddressTxt.setFocusCycleRoot(true);
        emailAddressTxt.addFocusListener(new
java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                emailAddressTxtFocusLost(evt);
            }
        });

        passwordField.setFocusCycleRoot(true);

        confirmPasswordField.setFocusCycleRoot(true);

        firstNameTxt.setFocusCycleRoot(true);

        dateOfBirthTxt.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                dateOfBirthTxtActionPerformed(evt);
            }
        });

        logInBtn.setText("Back to Log In");
        logInBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                logInBtnActionPerformed(evt);
            }
        });

        clearBtn.setText("Clear Form");
```

```
        clearBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                clearBtnActionPerformed(evt);
            }
        });

        createAccountBtn.setText("Create Account");
        createAccountBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                createAccountBtnActionPerformed(evt);
            }
        });

        passwordLabel.setForeground(new java.awt.Color(255, 51, 51));

        confirmPasswordLabel.setForeground(new java.awt.Color(255, 51,
51));

        firstNameLabel.setForeground(new java.awt.Color(255, 51, 51));

        surnameLabel.setForeground(new java.awt.Color(255, 51, 51));

        postcodeLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressOneLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressTwoLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressThreeLabel.setForeground(new java.awt.Color(255, 51,
51));

        addressFourLabel.setForeground(new java.awt.Color(255, 51, 51));

        discordUsernameLabel.setForeground(new java.awt.Color(255, 51,
51));

        discordNumberLabel.setForeground(new java.awt.Color(255, 51,
51));

        dateOfBirthLabel.setForeground(new java.awt.Color(255, 51, 51));

        phoneNumberLabel.setForeground(new java.awt.Color(255, 51, 51));

        viewPassBtn.setText("View");
        viewPassBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewPassBtnActionPerformed(evt);
            }
```

```
        });

        viewConfirmBtn.setText("View");
        viewConfirmBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewConfirmBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING, false)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.TRAILING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel7)
                                .addGap(133, 133, 133)
                                .addComponent(postcodeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(jPanel1Layout.createSequential
Group()
                                    .addComponent(jLabel4)
                                    .addPreferredGap(javax.swing.LayoutS
tyle.ComponentPlacement.RELATED)
                                    .addComponent(jLabel19))
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                    .addGroup(javax.swing.GroupLayout.Al
ignment.LEADING, jPanel1Layout.createSequentialGroup()
                                        .addComponent(jLabel6)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(surnameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(javax.swing.GroupLayout.Al
ignment.LEADING, jPanel1Layout.createSequentialGroup()
                                        .addComponent(jLabel5)
```

```
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(firstNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel1)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED)
                                        .addComponent(jLabel16)
                                        .addGap(107, 107, 107)
                                        .addComponent(usernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                            .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                .addComponent(jLabel2)
                                                .addPreferredGap(javax.s
wing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(jLabel17))
                                            .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                .addComponent(jLabel3)
                                                .addPreferredGap(javax.s
wing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(jLabel18))
)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                            .addComponent(emailAddressTx
t, javax.swing.GroupLayout.DEFAULT_SIZE, 255, Short.MAX_VALUE)
                                            .addComponent(passwordField)
                                            .addComponent(confirmPasswor
dField)))))))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(36, 36, 36)
                        .addComponent(logInBtn)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(clearBtn)))
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
```

```
                        .addGap(60, 60, 60)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(firstNameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(surnameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(postcodeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(viewConfirmBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(viewPassBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(11, 11, 11)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(passwordLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 70, Short.MAX_VALUE)
                            .addComponent(confirmPasswordLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
                .addGap(18, 18, 18)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel9)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(addressLineTwoTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel8)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
Short.MAX_VALUE)
                                    .addComponent(addressLineOneTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(jLabel10)
                                        .addComponent(jLabel11)
                                        .addComponent(jLabel12)
                                        .addComponent(jLabel14)
                                        .addComponent(jLabel13))
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                        .addComponent(discordNumberTxt)
                                        .addComponent(discordUsernameTxt)
                                        .addComponent(addressLineFourTxt)
                                        .addComponent(addressLineThreeTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 255, Short.MAX_VALUE)))
                                .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                    .addComponent(jLabel15)
                                    .addGap(29, 29, 29)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(dateOfBirthTxt,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                            .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, 133, Short.MAX_VALUE)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(addressOneLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressTwoLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressThreeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressFourLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(discordUsernameLabel,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(discordNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(dateOfBirthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(phoneNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addComponent(createAccountBtn))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(60, 60, 60)
                                .addComponent(addressOneLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressTwoLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressThreeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressFourLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(discordUsernameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(26, 26, 26)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel1)
                                    .addComponent(jLabel16)
                                    .addComponent(usernameTxt,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(18, 18, 18)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel2)
                                    .addComponent(jLabel17)
                                    .addComponent(emailAddressTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(18, 18, 18)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jLabel3)
                                        .addComponent(jLabel18)
                                        .addComponent(passwordField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(viewPassBtn))
                                    .addComponent(passwordLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.UNRELATED)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jLabel4)
                                        .addComponent(jLabel19)
                                        .addComponent(confirmPasswordFie
ld, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(viewConfirmBtn))
                                    .addComponent(confirmPasswordLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.UNRELATED)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                        .addComponent(jLabel5)
                                        .addComponent(firstNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addComponent(firstNameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.UNRELATED)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                            .addComponent(jLabel6)
                                            .addComponent(surnameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addComponent(surnameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                            .addGap(18, 18, 18)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(jLabel7)
                                .addGroup(jPanel1Layout.createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addComponent(postcodeTxt,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGroup(jPanel1Layout.createSequential
Group()
                                        .addGroup(jPanel1Layout.createParall
elGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                            .addComponent(discordNumberLabel
, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(postcodeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addGap(18, 18, 18)
                                        .addComponent(dateOfBirthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(18, 18, 18)
                                        .addComponent(phoneNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(26, 26, 26)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel8)
                                .addComponent(addressLineOneTxt,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel9)
                            .addComponent(addressLineTwoTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel10)
                            .addComponent(addressLineThreeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel11)
                            .addComponent(addressLineFourTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel12)
                            .addComponent(discordUsernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel13)
                            .addComponent(discordNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel14)
                            .addComponent(dateOfBirthTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
```

```
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel15)
                                .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addGap(20, 20, 20)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(logInBtn)
                    .addComponent(clearBtn)
                    .addComponent(createAccountBtn))
                .addContainerGap(36, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void
dateOfBirthTxtActionPerformed(java.awt.event.ActionEvent evt)
{

    }
    private void clearForm(){
        //sets all fields to empty
        usernameTxt.setText("");emailAddressTxt.setText("");passwordFiel
d.setText("");confirmPasswordField.setText("");firstNameTxt.setText("");
        surnameTxt.setText("");postcodeTxt.setText("");addressLineOneTxt
.setText("");addressLineTwoTxt.setText("");addressLineThreeTxt.setText("
");
        addressLineFourTxt.setText("");discordNumberTxt.setText("");disc
ordUsernameTxt.setText("");dateOfBirthTxt.setText("");phoneNumberTxt.set
Text("");
    }
    private void clearBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //sets all fields to empty
```

```
        clearForm();
    }

    private void logInBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //goes back to the log in form
        LogIn log = new LogIn();
        log.setVisible(true);
        this.dispose();
    }
    /**
     * Attempts to create an account after performing validation on the
appropriate fields
     * @param evt The button being pressed
     */
    private void
createAccountBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //validates the account details to ensure they are valid. Once
they have been validated, the account is created
        //presence check
        if(!(Validation.isPresent(usernameTxt.getText())&&
Validation.isPresent(emailAddressTxt.getText())&&
Validation.isPresent(new
String(passwordField.getPassword()))&&Validation.isPresent(new
String(confirmPasswordField.getPassword()))))){
            JOptionPane.showMessageDialog(this, "Username, email
address, password and confirm password must not be left blank.","Invalid
input",JOptionPane.OK_OPTION);
            return;
        }
        if(Validation.containsIllegalStrings(usernameTxt.getText()+email
AddressTxt.getText()+new String(passwordField.getPassword())+new
String(confirmPasswordField.getPassword())+firstNameTxt.getText()+surnam
eTxt.getText()+postcodeTxt.getText()+addressLineOneTxt.getText()
                +addressLineTwoTxt.getText()+addressLineThreeTxt.getText
()+addressLineFourTxt.getText()+discordNumberTxt.getText()+discordUserna
meTxt.getText()+dateOfBirthTxt.getText()+phoneNumberTxt.getText())){//pe
rforms a look up check to see whether any field contains an illegal
character
            JOptionPane.showMessageDialog(this, "One or more fields
contain an illegal character sequence. Here is a list of illegal
caharacter sequences: "+Validation.getListOfIllegalStrings(),"Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        //check username inputs
        if(!Validation.isAppropriateLength(15,
usernameTxt.getText())){//checks to see if username is 15 charcters or
less - length check
            JOptionPane.showMessageDialog(this, "The username can not
exceed 15 characters in length","Invalid Input",JOptionPane.OK_OPTION);
            return;
```

```
        }
        if(usernameTxt.getText().contains("@") ||
usernameTxt.getText().contains(" ")){//format check
            JOptionPane.showMessageDialog(this, "The username can not
contain the @ symbol OR a space.","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        //checks email address
        if(!emailAddressTxt.getText().contains("@") ||
emailAddressTxt.getText().contains(" ")){
            JOptionPane.showMessageDialog(this, "The email address must
contain the @ symbol and no spaces.","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        //length check
        if(!Validation.isAppropriateLength(39,
emailAddressTxt.getText())){//email address must not be longer than 39
characters
            JOptionPane.showMessageDialog(this, "The email address must
not be longer than 39 characters.","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!new String(passwordField.getPassword()).equals(new String
(confirmPasswordField.getPassword())){//checks that both passwords
match - double entry verification validation
            JOptionPane.showMessageDialog(this, "The two passwords do
not match.","Invalid Input",JOptionPane.OK_OPTION);
            return;
        }
        if(new String(passwordField.getPassword()).length()<5 || new
String(passwordField.getPassword()).length()>20){//length check
            JOptionPane.showMessageDialog(this, "The password must be
between 5-20 characters in length (inclusive).","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(Validation.isPresent(firstNameTxt.getText())){//only performs
validation if first name is not empty
            if(!Validation.isAppropriateLength(35,
firstNameTxt.getText())){//length check - first name can't be more than
35 characters
                JOptionPane.showMessageDialog(this, "The first name can
not be longer than 35 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.consistsOnlyOfLetters(firstNameTxt.getText()
){
                JOptionPane.showMessageDialog(this, "The first name can
only contain characters that are in the english alphabet","Invalid
```

```
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(Validation.isPresent(surnameTxt.getText())){//only performs
validation if  surname is not empty
            if(!Validation.isAppropriateLength(40,
surnameTxt.getText())){//length check - surname can't be more than 40
characters
                JOptionPane.showMessageDialog(this, "The surname can not
be longer than 40 characters","Invalid Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.consistsOnlyOfLetters(surnameTxt.getText())){
                JOptionPane.showMessageDialog(this, "The surname can
only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(Validation.isPresent(postcodeTxt.getText())){//only performs
validation is postcode is not left empty
            if(!Validation.isValidPostcode(postcodeTxt.getText())){
                JOptionPane.showMessageDialog(this, "The postcode must
be between 6-8 characters long, contain exactly one space and have all
non-space characters capitalised","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(!Validation.isAppropriateLength(40,
addressLineOneTxt.getText())){//length check
            JOptionPane.showMessageDialog(this, "Each address line can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!Validation.isAppropriateLength(40,
addressLineTwoTxt.getText())){//length check
            JOptionPane.showMessageDialog(this, "Each address line can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!Validation.isAppropriateLength(40,
addressLineThreeTxt.getText())){//length check
            JOptionPane.showMessageDialog(this, "Each address line can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!Validation.isAppropriateLength(40,
addressLineFourTxt.getText())){//length check
```

```
            JOptionPane.showMessageDialog(this, "Each address line can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(!Validation.isAppropriateLength(32,
discordUsernameTxt.getText())){
            JOptionPane.showMessageDialog(this, "The Discord username
can not be longer than 32 characters","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
        if(Validation.isPresent(discordNumberTxt.getText())){
            if(!Validation.isInteger(discordNumberTxt.getText())){
                JOptionPane.showMessageDialog(this, "The Discord number
must be an integer","Invalid Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(4,
discordNumberTxt.getText())){
                JOptionPane.showMessageDialog(this, "The Discord number
must be no more than 4 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.rangeCheck(0, 9999,
Integer.parseInt(discordNumberTxt.getText()))){//range check
                JOptionPane.showMessageDialog(this, "The Discord number
must be between 0000-9999","Invalid Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(Validation.isPresent(dateOfBirthTxt.getText())){
            if(!Validation.formatCheckDateOfBirth(dateOfBirthTxt.getText
())){
                JOptionPane.showMessageDialog(this, "The date of birth
must be a valid date of birth in the form dd/mm/yyyy","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        if(Validation.isPresent(phoneNumberTxt.getText())){
            if(!Validation.isAppropriateLength(20,
phoneNumberTxt.getText())){
                JOptionPane.showMessageDialog(this, "Phone number can
not be longer than 20 characters.","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.onlyContainsPhoneNumberCharacters(phoneNumber
Txt.getText())){
                JOptionPane.showMessageDialog(this, "Phone number can
only contain plusses, spaces and integers.","Invalid
```

```
Input",JOptionPane.OK_OPTION);
                return;
            }
        }
        //lookup check to ensure that both email address and username
are not already taken
        try{
            if(ThreadClient.isUserNameAlreadyTaken(usernameTxt.getText()
)){
                JOptionPane.showMessageDialog(this, "The username is
already taken. Plesae try a different username.","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(ThreadClient.isEmailAddressAlreadyTaken(emailAddressTxt.g
etText())){
                JOptionPane.showMessageDialog(this, "The email is
already taken. Plesae try a different email address.","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
        }catch(Exception e){
            JOptionPane.showMessageDialog(this, "An error occured:
"+e.getMessage()+". This is almost certainly becuase something is wrong
with your internet connection or the server.","Network
Error",JOptionPane.OK_OPTION);
            return;
        }
        try{
            //creates the account once the validation has been performed
            ThreadClient.createAccount(new
String[]{usernameTxt.getText(),new
String(passwordField.getPassword()),firstNameTxt.getText(),surnameTxt.ge
tText(),postcodeTxt.getText(),addressLineOneTxt.getText(),addressLineTwo
Txt.getText(),addressLineThreeTxt.getText(),addressLineFourTxt.getText()
,discordUsernameTxt.getText(),discordNumberTxt.getText(),dateOfBirthTxt.
getText(),phoneNumberTxt.getText(),"false","false","1000","false"},email
AddressTxt.getText());
            JOptionPane.showMessageDialog(this,"Account succesfully
created.","Success", JOptionPane.OK_OPTION);
            LogIn log = new LogIn();//takes the user to the log in
screen once the account has been made
            log.setVisible(true);this.dispose();
            return;
        }catch(Exception e){
            JOptionPane.showMessageDialog(this, "An error occured:
"+e.getMessage()+". This is almost certainly becuase something is wrong
with your internet connection or the server.","Network
Error",JOptionPane.OK_OPTION);
            return;
        }
    }
```

```
    private void usernameTxtKeyTyped(java.awt.event.KeyEvent evt)
{

    }

    private void usernameTxtFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void emailAddressTxtFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void viewPassBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //allows the user to mask or unmask the password
        viewPass=!viewPass;
        if(viewPass==true){
            char c =0;
            passwordField.setEchoChar(c);
            viewPassBtn.setText("Hide");
        }else{
            passwordField.setEchoChar(DEFAULT_ECHO_CHAR);
            viewPassBtn.setText("View");
        }
    }

    private void
viewConfirmBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //allows the user to mask or unmask the password
        viewConfirmPass=!viewConfirmPass;
        if(viewConfirmPass==true){
            char c =0;
            confirmPasswordField.setEchoChar(c);
            viewConfirmBtn.setText("Hide");
        }else{
            confirmPasswordField.setEchoChar(DEFAULT_ECHO_CHAR);
            viewConfirmBtn.setText("View");
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
```

```
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(CreateAccount.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(CreateAccount.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(CreateAccount.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(CreateAccount.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new CreateAccount().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel addressFourLabel;
    private javax.swing.JTextField addressLineFourTxt;
    private javax.swing.JTextField addressLineOneTxt;
    private javax.swing.JTextField addressLineThreeTxt;
    private javax.swing.JTextField addressLineTwoTxt;
    private javax.swing.JLabel addressOneLabel;
    private javax.swing.JLabel addressThreeLabel;
    private javax.swing.JLabel addressTwoLabel;
    private javax.swing.JButton clearBtn;
    private javax.swing.JPasswordField confirmPasswordField;
    private javax.swing.JLabel confirmPasswordLabel;
    private javax.swing.JButton createAccountBtn;
    private javax.swing.JLabel dateOfBirthLabel;
    private javax.swing.JTextField dateOfBirthTxt;
    private javax.swing.JLabel discordNumberLabel;
    private javax.swing.JTextField discordNumberTxt;
```

```
    private javax.swing.JLabel discordUsernameLabel;
    private javax.swing.JTextField discordUsernameTxt;
    private javax.swing.JTextField emailAddressTxt;
    private javax.swing.JLabel firstNameLabel;
    private javax.swing.JTextField firstNameTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton logInBtn;
    private javax.swing.JPasswordField passwordField;
    private javax.swing.JLabel passwordLabel;
    private javax.swing.JLabel phoneNumberLabel;
    private javax.swing.JTextField phoneNumberTxt;
    private javax.swing.JLabel postcodeLabel;
    private javax.swing.JTextField postcodeTxt;
    private javax.swing.JLabel surnameLabel;
    private javax.swing.JTextField surnameTxt;
    private javax.swing.JTextField usernameTxt;
    private javax.swing.JButton viewConfirmBtn;
    private javax.swing.JButton viewPassBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import javax.swing.JOptionPane;
```

```
import philipm.cs5.software.development.server.system.UserInfo;

/**
 * Creates a form that allows a user to create an online game against a
friend.
 * @author mortimer
 */
public class CreateGameAgainstFriend extends javax.swing.JFrame {
    private SessionInfo session;
    /**
     * Creates new form CreateGameAgainstFriend
     */
    private CreateGameAgainstFriend() {
        initComponents();
    }
    /**
     * Creates new form CreateGameAgainstFriend
     * @param session The session information of the user
     */
    public CreateGameAgainstFriend(SessionInfo session){
        this.session=session;//stores session
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        backBtn = new javax.swing.JButton();
        jLabel2 = new javax.swing.JLabel();
        friendUserNameTxt = new javax.swing.JTextField();
        useCodeOrUsernameBtn = new javax.swing.JButton();
        fiveMinutesBtn = new javax.swing.JToggleButton();
        tenMinutesBtn = new javax.swing.JToggleButton();
        unlimitedBtn = new javax.swing.JToggleButton();
        jLabel3 = new javax.swing.JLabel();
        createGameBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Create Game Against Friend");

        jLabel1.setText("Create Online Game Against Friend");
```

```
        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        jLabel2.setText("Friend Username");

        useCodeOrUsernameBtn.setText("Create Code to Share Instead");
        useCodeOrUsernameBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                useCodeOrUsernameBtnActionPerformed(evt);
            }
        });

        fiveMinutesBtn.setText("Five Minutes");
        fiveMinutesBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                fiveMinutesBtnActionPerformed(evt);
            }
        });

        tenMinutesBtn.setText("Ten Minutes");
        tenMinutesBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                tenMinutesBtnActionPerformed(evt);
            }
        });

        unlimitedBtn.setText("Unlimited");
        unlimitedBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                unlimitedBtnActionPerformed(evt);
            }
        });

        jLabel3.setText("Time Control");

        createGameBtn.setText("Create Game");
        createGameBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
```

```
{
                createGameBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(187, 187, 187)
                        .addComponent(jLabel1))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(30, 30, 30)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel3)
                                .addGap(15, 15, 15)
                                .addComponent(fiveMinutesBtn)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED)
                                .addComponent(tenMinutesBtn)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED)
                                .addComponent(unlimitedBtn))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addComponent(jLabel2)
                                    .addComponent(backBtn))
                                .addGap(48, 48, 48)
                                .addComponent(friendUserNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 297,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                .addContainerGap(137, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addComponent(useCodeOrUsernameBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 357,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addGap(126, 126, 126))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addComponent(createGameBtn)
                        .addGap(90, 90, 90))))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(30, 30, 30)
                .addComponent(jLabel1)
                .addGap(49, 49, 49)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(friendUserNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(28, 28, 28)
                .addComponent(useCodeOrUsernameBtn)
                .addGap(58, 58, 58)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(fiveMinutesBtn)
                    .addComponent(tenMinutesBtn)
                    .addComponent(unlimitedBtn)
                    .addComponent(jLabel3))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 74, Short.MAX_VALUE)
                .addComponent(createGameBtn)
                .addGap(32, 32, 32)
                .addComponent(backBtn)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
```

```
        pack();
    }// </editor-fold>

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
       //takes user back to online menu
       OnlineMenu online = new OnlineMenu(session);
       online.setVisible(true);
       this.dispose();
    }

    private void
fiveMinutesBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only one time control can be choosen
        if(fiveMinutesBtn.isSelected()){
            tenMinutesBtn.setSelected(false);
            unlimitedBtn.setSelected(false);
        }
    }

    private void tenMinutesBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // ensures that only one time control can be choosen
        if(tenMinutesBtn.isSelected()){
            fiveMinutesBtn.setSelected(false);
            unlimitedBtn.setSelected(false);
        }
    }

    private void unlimitedBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // ensures that only one time control can be choosen
        if(unlimitedBtn.isSelected()){
            fiveMinutesBtn.setSelected(false);
            tenMinutesBtn.setSelected(false);
        }
    }

    private void
useCodeOrUsernameBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //updates the username box and button text if relevant
        if(useCodeOrUsernameBtn.getText().equals("Create Code to Share
Instead")){
            friendUserNameTxt.setEditable(false);
            friendUserNameTxt.setEnabled(false);
            useCodeOrUsernameBtn.setText("Input Username Instead");
        }else{
            friendUserNameTxt.setEnabled(true);
            friendUserNameTxt.setEditable(true);
            useCodeOrUsernameBtn.setText("Create Code to Share
Instead");
```

```
        }
    }

    private void createGameBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //creates a new game after performing relevant validation
        //gets time control to use
        int timeLeft=-100;
        if(fiveMinutesBtn.isSelected()){
            timeLeft=5*60;
        }else if(tenMinutesBtn.isSelected()){
            timeLeft=600;
        }else if(unlimitedBtn.isSelected()==false){
            JOptionPane.showMessageDialog(this,"Please select a time
control.","Error",JOptionPane.OK_OPTION);
            return;
        }
        String userName="";//stores the username of the opponent
        if(useCodeOrUsernameBtn.getText().equals("Create Code to Share
Instead")){
            userName=friendUserNameTxt.getText();
            if(!Validation.isPresent(userName) ||
userName.equals(session.getUsername())||
userName.equals(UserInfo.GUEST_ID)){//ensures that the username can't be
empty or be that of the current user
                JOptionPane.showMessageDialog(this,"Please input a valid
username.","Error",JOptionPane.OK_OPTION);
                return;
            }
            try{
                if(ThreadClient.isUserNameAlreadyTaken(userName)==false)
{//checks to see if username is taken
                    JOptionPane.showMessageDialog(this,"User not found.
Please enter  a valid username.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                if(ThreadClient.getOnlineGameData(userName)[2][0].equals
("")==false){//checks to see if friend is already playing three games
                    JOptionPane.showMessageDialog(this,"User selected is
currently playing three games. Users can not play more than three
games.","Error",JOptionPane.OK_OPTION);
                    return;
                }
            }catch(Exception e){
                JOptionPane.showMessageDialog(this, "There has been an
error when communicating with the
server.","Error",JOptionPane.OK_OPTION);
                return;
            }
        }
        try{
            String
code=ThreadClient.createGameAgainstFriend(session.getUsername(),
```

```
userName, timeLeft);//creates the game
            if(code.equals("-1")){
                JOptionPane.showMessageDialog(this, "There has been an
error when communicating with the
server.","Error",JOptionPane.OK_OPTION);
            }else if(userName.equals("")){
                JOptionPane.showMessageDialog(this,"Game created. Game
code "+code,"Game Created",JOptionPane.OK_OPTION);
            }else{
                JOptionPane.showMessageDialog(this,"Game created.","Game
Created",JOptionPane.OK_OPTION);
            }
        }catch(Exception e){
            JOptionPane.showMessageDialog(this, "There has been an error
when communicating with the
server.","Error",JOptionPane.OK_OPTION);
        }
        //takes user back to online menu
        OnlineMenu online = new OnlineMenu(session);
        online.setVisible(true);
        this.dispose();

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(CreateGameAgainstFriend.c
lass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(CreateGameAgainstFriend.c
lass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(CreateGameAgainstFriend.c
```

```
lass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(CreateGameAgainstFriend.c
lass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new CreateGameAgainstFriend().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JButton createGameBtn;
    private javax.swing.JToggleButton fiveMinutesBtn;
    private javax.swing.JTextField friendUserNameTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JToggleButton tenMinutesBtn;
    private javax.swing.JToggleButton unlimitedBtn;
    private javax.swing.JButton useCodeOrUsernameBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;
import philipm.cs5.software.development.server.system.UserInfo;

/**
 * Creates a form that allows the user to play an offline game against a
computer opponent
```

```
 * @author mortimer
 */
public class CreateOfflineGame extends javax.swing.JFrame {
    SessionInfo session;//stores the session information
    /**
     * Creates new form CreateOfflineGame
     */
    private CreateOfflineGame() {
        initComponents();
    }
    /**
     * Creates new form CreateOfflineGame
     * @param session The session information of the current user
     */
    public CreateOfflineGame(SessionInfo session){
        initComponents();
        this.session=session;
        gameTitleTxt.setText(suggestedGameTitle(session));//sets the
game title to a suggested title that has not yet been taken
    }
    /**
     * Suggests a suitable game id. This will be either Game 1, Game 2
or Game 3.
     * @param ses The session info object used to get the appropriate
username
     * @return The title
     */
    public static String suggestedGameTitle(SessionInfo ses){
        String title="Game ";int num=1;//stores the number to put at the
end of the title
        String titlesTaken[]=new String[3];int noOfGames=0;//stores all
game titles
        boolean isTaken=false;
        try{
            //gets the title of all current games
            FileReader read = new
FileReader(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+ses.getUserna
me()+".txt");
            BufferedReader buffRead = new BufferedReader(read);
            String lineRead;
            while((lineRead=buffRead.readLine())!=null){
                titlesTaken[noOfGames]=lineRead.split(UserInfo.SPLIT_ITE
M)[0];
                noOfGames++;
            }buffRead.close();read.close();
            //tries game1
            for(int i=0;i<noOfGames;i++){
                if(titlesTaken[i].equals("Game1")){
                    isTaken=true;
                    break;
                }
            }
            if(isTaken){
```

```
                isTaken=false;
                for(int i=0;i<noOfGames;i++){//tries Game2
                    if(titlesTaken[i].equals("Game2")){
                        isTaken=true;
                        break;
                    }
                }
                if(isTaken){
                    return "Game3";
                }else{
                    return "Game2";
                }
            }else{
                return "Game1";
            }
        }catch(IOException e){
            JOptionPane.showMessageDialog(null,"An unexpected error
occured "+e);
        }
        return title+String.valueOf(num);
    }
    /**
     * Checks to see if the game title chosen is valid (i.e. unique and
not containing any illegal characters)
     * @param ses The session information
     * @param title The game title
     * @return True if valid, otherwise false
     */
    public static boolean isGameTitleValid(SessionInfo ses,String
title){
        if(Validation.containsIllegalStrings(title)){//checks that the
title does not contain illegal characters
            return false;
        }
        //searches to ensure that game title is unique
        try{
            FileReader read = new
FileReader(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+ses.getUserna
me()+".txt");
            BufferedReader buffRead = new BufferedReader(read);
            String lineRead;
            while((lineRead=buffRead.readLine())!=null){
                if(lineRead.split(UserInfo.SPLIT_ITEM)[0].equals(title))
{
                    return false;
                }
            }buffRead.close();read.close();
        }catch(IOException e){
            JOptionPane.showMessageDialog(null,"An unexpected error
occured "+e);
        }
        return true;
    }
```

```
    /**
     * Adds new offline game to text file
     * @param gameTitle The game title
     * @param opponentId The opponent id
     * @param isUserWhite Whether the user plays as white
     * @param secondsWhiteAndBlackHave The time control setting. This
equals -100 if no time control
     * @param sess The session info
     */
    public static void addNewGameToFile(String gameTitle,String
opponentId, boolean isUserWhite,int secondsWhiteAndBlackHave,SessionInfo
sess){
        try{
            //game id,opponentID, isUserWhite, boardState(s),time game
has taken, time white has left, time black has left
            //checks to see if file is empty
            FileReader read = new
FileReader(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+sess.getUsern
ame()+".txt");
            BufferedReader buffRead = new BufferedReader(read);
            boolean isEmpty =
buffRead.readLine()==null;buffRead.close();read.close();
            //writes new record to text file
            FileWriter write = new
FileWriter(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+sess.getUsern
ame()+".txt",true);
            BufferedWriter buffWrite = new BufferedWriter(write);
            if(isEmpty==false){
                buffWrite.newLine();
            }
            buffWrite.write(gameTitle+UserInfo.SPLIT_ITEM+opponentId+Use
rInfo.SPLIT_ITEM+String.valueOf(isUserWhite)+UserInfo.SPLIT_ITEM+Offline
Play.NEW_BOARD+UserInfo.SPLIT_ITEM+"0"+UserInfo.SPLIT_ITEM+String.valueO
f(secondsWhiteAndBlackHave)+UserInfo.SPLIT_ITEM+String.valueOf(secondsWh
iteAndBlackHave));
            buffWrite.flush();write.flush();buffWrite.close();write.clos
e();
        }catch(IOException e){
            JOptionPane.showMessageDialog(null,"An unexpected error has
occured "+e,"Error",JOptionPane.OK_OPTION);
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {
```

```
        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        difficultyOneBtn = new javax.swing.JToggleButton();
        difficultyTwoBtn = new javax.swing.JToggleButton();
        difficultyThreeBtn = new javax.swing.JToggleButton();
        difficultyFourBtn = new javax.swing.JToggleButton();
        difficultyFiveBtn = new javax.swing.JToggleButton();
        jLabel2 = new javax.swing.JLabel();
        whiteBtn = new javax.swing.JToggleButton();
        randomBtn = new javax.swing.JToggleButton();
        blackBtn = new javax.swing.JToggleButton();
        backBtn = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
        createGameBtn = new javax.swing.JButton();
        jLabel4 = new javax.swing.JLabel();
        gameTitleTxt = new javax.swing.JTextField();
        jLabel5 = new javax.swing.JLabel();
        unlimitedTimeBtn = new javax.swing.JToggleButton();
        tenMinsBtn = new javax.swing.JToggleButton();
        fiveMinutesBtn = new javax.swing.JToggleButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Create New Game");

        jLabel1.setText("Computer Difficulty");

        difficultyOneBtn.setText("1 (Easiest)");
        difficultyOneBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                difficultyOneBtnActionPerformed(evt);
            }
        });

        difficultyTwoBtn.setText("2");
        difficultyTwoBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                difficultyTwoBtnActionPerformed(evt);
            }
        });

        difficultyThreeBtn.setText("3");
        difficultyThreeBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                difficultyThreeBtnActionPerformed(evt);
            }
```

```
        });

        difficultyFourBtn.setText("4");
        difficultyFourBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                difficultyFourBtnActionPerformed(evt);
            }
        });

        difficultyFiveBtn.setText("5 (Hardest)");
        difficultyFiveBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                difficultyFiveBtnActionPerformed(evt);
            }
        });

        jLabel2.setText("You play as");

        whiteBtn.setText("White");
        whiteBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                whiteBtnActionPerformed(evt);
            }
        });

        randomBtn.setText("Random");
        randomBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                randomBtnActionPerformed(evt);
            }
        });

        blackBtn.setText("Black");
        blackBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                blackBtnActionPerformed(evt);
            }
        });

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
```

```
        });

        jLabel3.setText("Create New Game");
        jLabel3.setToolTipText("");

        createGameBtn.setText("Create Game");
        createGameBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                createGameBtnActionPerformed(evt);
            }
        });

        jLabel4.setText("Game Title");

        jLabel5.setText("Time Control");

        unlimitedTimeBtn.setText("Unlimited");
        unlimitedTimeBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                unlimitedTimeBtnActionPerformed(evt);
            }
        });

        tenMinsBtn.setText("10 Minutes");
        tenMinsBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                tenMinsBtnActionPerformed(evt);
            }
        });

        fiveMinutesBtn.setText("5 Minutes");
        fiveMinutesBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                fiveMinutesBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
```

```
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(createGameBtn)
                .addGap(140, 140, 140))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                    .addComponent(jLabel2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                    .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                                .addGap(55, 55, 55)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(whiteBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED)
                                        .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                            .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                .addGap(7, 7, 7)
                                                .addComponent(randomBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                            .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                .addComponent(difficulty
TwoBtn)
                                                .addGap(0, 0, 0)
                                                .addComponent(difficulty
ThreeBtn)
                                                .addGap(1, 1, 1)
                                                .addComponent(difficulty
FourBtn)))
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED)
                                        .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                            .addComponent(difficultyFive
```

```
Btn, javax.swing.GroupLayout.DEFAULT_SIZE, 123, Short.MAX_VALUE)
                                                .addComponent(blackBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                                        .addComponent(difficultyOneBtn)
                                        .addComponent(gameTitleTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 314,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGroup(jPanel1Layout.createSequen
tialGroup()
                                            .addComponent(fiveMinutesBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 120,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addGap(18, 18, 18)
                                            .addComponent(tenMinsBtn)
                                            .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED)
                                            .addComponent(unlimitedTimeBtn,
javax.swing.GroupLayout.DEFAULT_SIZE, 105, Short.MAX_VALUE))))
                                .addComponent(backBtn)))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(299, 299, 299)
                            .addComponent(jLabel3))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addContainerGap()
                            .addComponent(jLabel4))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addContainerGap()
                            .addComponent(jLabel5)))
                    .addContainerGap(166, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel3)
                .addGap(24, 24, 24)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.TRAILING)
                        .addComponent(jLabel4)
                        .addComponent(gameTitleTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                        .addComponent(jLabel1)
                        .addComponent(difficultyOneBtn)
                        .addComponent(difficultyTwoBtn)
                        .addComponent(difficultyThreeBtn)
                        .addComponent(difficultyFourBtn)
```

```
                    .addComponent(difficultyFiveBtn))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.UNRELATED)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(whiteBtn)
                    .addComponent(randomBtn)
                    .addComponent(blackBtn))
                .addGap(29, 29, 29)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(unlimitedTimeBtn)
                    .addComponent(tenMinsBtn)
                    .addComponent(fiveMinutesBtn))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 13, Short.MAX_VALUE)
                .addComponent(createGameBtn)
                .addGap(1, 1, 1)
                .addComponent(backBtn)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void
difficultyOneBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only difficulty option can be selected
        if(difficultyOneBtn.isSelected()){
            difficultyFiveBtn.setSelected(false);
            difficultyFourBtn.setSelected(false);
            difficultyThreeBtn.setSelected(false);
            difficultyTwoBtn.setSelected(false);
        }
```

```
    }

    private void
difficultyTwoBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only difficulty option can be selected
        if(difficultyTwoBtn.isSelected()){
            difficultyFiveBtn.setSelected(false);
            difficultyFourBtn.setSelected(false);
            difficultyThreeBtn.setSelected(false);
            difficultyOneBtn.setSelected(false);
        }
    }

    private void
difficultyThreeBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only difficulty option can be selected
        if(difficultyThreeBtn.isSelected()){
            difficultyFiveBtn.setSelected(false);
            difficultyFourBtn.setSelected(false);
            difficultyOneBtn.setSelected(false);
            difficultyTwoBtn.setSelected(false);
        }
    }

    private void
difficultyFourBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only difficulty option can be selected
        if(difficultyFourBtn.isSelected()){
            difficultyFiveBtn.setSelected(false);
            difficultyOneBtn.setSelected(false);
            difficultyThreeBtn.setSelected(false);
            difficultyTwoBtn.setSelected(false);
        }
    }

    private void
difficultyFiveBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only difficulty option can be selected
        if(difficultyFiveBtn.isSelected()){
            difficultyOneBtn.setSelected(false);
            difficultyFourBtn.setSelected(false);
            difficultyThreeBtn.setSelected(false);
            difficultyTwoBtn.setSelected(false);
        }
    }

    private void whiteBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only one colour can be choosen
```

```
        if(whiteBtn.isSelected()){
            blackBtn.setSelected(false);
            randomBtn.setSelected(false);
        }
    }

    private void randomBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // ensures that only one colour can be choosen
        if(randomBtn.isSelected()){
            blackBtn.setSelected(false);
            whiteBtn.setSelected(false);
        }
    }

    private void blackBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only one colour can be choosen
        if(blackBtn.isSelected()){
            whiteBtn.setSelected(false);
            randomBtn.setSelected(false);
        }
    }

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes user back to offline menu
        OfflinePlay offline = new OfflinePlay(session);
        offline.setVisible(true);
        this.dispose();
    }

    private void createGameBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // creates a new game after checking that game settings have
actually been selected;
        //gets selected difficulty setting
        String computerDifficulty="";
        if(difficultyOneBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_ONE_COMP;
        }else if(difficultyTwoBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_TWO_COMP;
        }else if(difficultyThreeBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_THREE_COMP;
        }else if(difficultyFourBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_FOUR_COMP;
        }else if(difficultyFiveBtn.isSelected()){
            computerDifficulty=OfflinePlay.LEVEL_FIVE_COMP;
        }else{
            //error message if no difficulty setting selected
            JOptionPane.showMessageDialog(this,"Please select a
difficulty setting.","Error",JOptionPane.OK_OPTION);
            return;
```

```
        }
        String gameTitle=gameTitleTxt.getText();//stores the game title
        if(isGameTitleValid(session,
gameTitle)==false||Validation.isPresent(gameTitle)==false){
            JOptionPane.showMessageDialog(this,"Please choose a valid,
unique game title.","Error",JOptionPane.OK_OPTION);
            return;
        }
        boolean isUserWhite=false;//stores which colour the user will
play as
        if(whiteBtn.isSelected()){
            isUserWhite=true;
        }else if(blackBtn.isSelected()){
            isUserWhite=false;
        }else if(randomBtn.isSelected()){
            //randomly assings user a colour
            if(Math.random()>=0.5){
                isUserWhite=true;
            }else{
                isUserWhite=false;
            }
        }else{
            //error message if no colour chosen
            JOptionPane.showMessageDialog(this,"Please select a colour
to play as.","Error",JOptionPane.OK_OPTION);
            return;
        }
        int timeControl=-100;
        if(unlimitedTimeBtn.isSelected()){
            timeControl=-100;
        }else if(fiveMinutesBtn.isSelected()){
            timeControl=5*60;
        }else if(tenMinsBtn.isSelected()){
            timeControl=10*60;
        }else{
            JOptionPane.showMessageDialog(this,"Please select a time
control setting.","Error",JOptionPane.OK_OPTION);
            return;
        }
        addNewGameToFile(gameTitle,computerDifficulty,isUserWhite,timeCo
ntrol,session);//adds game to file
        //takes user to game
        ChessBoardState[]state={new
ChessBoardState(OfflinePlay.NEW_BOARD)};
        ChessBoardOffline board = new ChessBoardOffline(session,
computerDifficulty, new
ChessGame(state,computerDifficulty.equals(OfflinePlay.LEVEL_TWO_COMP)),
isUserWhite, 0, timeControl, timeControl, gameTitle);
        board.setVisible(true);
        this.dispose();
    }

    private void
```

```
unlimitedTimeBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures only one time control can be selected
        if(unlimitedTimeBtn.isSelected()){
            fiveMinutesBtn.setSelected(false);
            tenMinsBtn.setSelected(false);
        }
    }

    private void tenMinsBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // ensures only one time control can be selected
        if(tenMinsBtn.isSelected()){
            fiveMinutesBtn.setSelected(false);
            unlimitedTimeBtn.setSelected(false);
        }
    }

    private void
fiveMinutesBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures only one time control can be selected
        if(fiveMinutesBtn.isSelected()){
            tenMinsBtn.setSelected(false);
            unlimitedTimeBtn.setSelected(false);
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(CreateOfflineGame.class.g
etName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
```

```
            java.util.logging.Logger.getLogger(CreateOfflineGame.class.g
etName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(CreateOfflineGame.class.g
etName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(CreateOfflineGame.class.g
etName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new CreateOfflineGame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JToggleButton blackBtn;
    private javax.swing.JButton createGameBtn;
    private javax.swing.JToggleButton difficultyFiveBtn;
    private javax.swing.JToggleButton difficultyFourBtn;
    private javax.swing.JToggleButton difficultyOneBtn;
    private javax.swing.JToggleButton difficultyThreeBtn;
    private javax.swing.JToggleButton difficultyTwoBtn;
    private javax.swing.JToggleButton fiveMinutesBtn;
    private javax.swing.JTextField gameTitleTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JToggleButton randomBtn;
    private javax.swing.JToggleButton tenMinsBtn;
    private javax.swing.JToggleButton unlimitedTimeBtn;
    private javax.swing.JToggleButton whiteBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;
import philipm.cs5.software.development.server.system.UserInfo;

/**
 * Creates a form that allows the user to create an offline chess game
against a human opponent
 * @author mortimer
 */
public class CreateOfflineGameAgainstPerson extends javax.swing.JFrame {
    SessionInfo session;//stores the session information
    /**
     * Creates new form CreateOfflineGame
     */
    private CreateOfflineGameAgainstPerson() {
        initComponents();
    }
    /**
     * Creates new form CreateOfflineGame
     * @param session The session information of the current user
     */
    public CreateOfflineGameAgainstPerson(SessionInfo session){
        initComponents();
        opponentNameTxt.setText(OfflinePlay.HUMAN_OPPONENT);//sets text
field to default value
        this.session=session;
        gameTitleTxt.setText(CreateOfflineGame.suggestedGameTitle(sessio
n));//sets the title to a suggested valid game title
    }
    /**
     * Creates a new offline game which is added to the correct text
file
     * @param opponentId The opponent id
     * @param isUserWhite whether the user is the white player
     */
    private void addNewGameToFile(String opponentId, boolean
isUserWhite){
        try{
            //opponentID, isUserWhite, fen string (s),time game has
taken
            //checks to see if file is empty
            FileReader read = new
FileReader(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+session.getUs
ername()+".txt");
            BufferedReader buffRead = new BufferedReader(read);
            boolean isEmpty =
```

```
buffRead.readLine()==null;buffRead.close();read.close();
            //writes new record to text file
            FileWriter write = new
FileWriter(OfflinePlay.OFFLINE_GAMES_FOLDER+File.separator+session.getUs
ername()+".txt",true);
            BufferedWriter buffWrite = new BufferedWriter(write);
            if(isEmpty==false){
                buffWrite.newLine();
            }
            buffWrite.write(opponentId+UserInfo.SPLIT_ITEM+String.valueO
f(isUserWhite)+UserInfo.SPLIT_ITEM+OfflinePlay.NEW_BOARD+UserInfo.SPLIT_
ITEM+"0");
            buffWrite.flush();write.flush();buffWrite.close();write.clos
e();
        }catch(IOException e){
            JOptionPane.showMessageDialog(this,"An unexpected error has
occured "+e,"Error",JOptionPane.OK_OPTION);
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        whiteBtn = new javax.swing.JToggleButton();
        randomBtn = new javax.swing.JToggleButton();
        blackBtn = new javax.swing.JToggleButton();
        backBtn = new javax.swing.JButton();
        jLabel3 = new javax.swing.JLabel();
        createGameBtn = new javax.swing.JButton();
        opponentNameTxt = new javax.swing.JTextField();
        jLabel4 = new javax.swing.JLabel();
        gameTitleTxt = new javax.swing.JTextField();
        jLabel5 = new javax.swing.JLabel();
        fiveMinutesBtn = new javax.swing.JToggleButton();
        tenMinutesBtn = new javax.swing.JToggleButton();
        unlimitedBtn = new javax.swing.JToggleButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Create New Game");

        jLabel1.setText("Opponent Name:");
```

```
        jLabel2.setText("You play as");

        whiteBtn.setText("White");
        whiteBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                whiteBtnActionPerformed(evt);
            }
        });

        randomBtn.setText("Random");
        randomBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                randomBtnActionPerformed(evt);
            }
        });

        blackBtn.setText("Black");
        blackBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                blackBtnActionPerformed(evt);
            }
        });

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        jLabel3.setText("Create New Game");
        jLabel3.setToolTipText("");

        createGameBtn.setText("Create Game");
        createGameBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                createGameBtnActionPerformed(evt);
            }
        });

        jLabel4.setText("Game Title");

        jLabel5.setText("Time Control");

        fiveMinutesBtn.setText("5 Minutes");
```

```
        fiveMinutesBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                fiveMinutesBtnActionPerformed(evt);
            }
        });

        tenMinutesBtn.setText("10 Minutes");
        tenMinutesBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                tenMinutesBtnActionPerformed(evt);
            }
        });

        unlimitedBtn.setText("Unlimited");
        unlimitedBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                unlimitedBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addContainerGap(471, Short.MAX_VALUE)
                        .addComponent(createGameBtn)
                        .addGap(37, 37, 37))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(backBtn)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 122,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel5))
```

```
                                .addGap(44, 44, 44)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(whiteBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 107,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(27, 27, 27)
                                        .addComponent(randomBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 111,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(15, 15, 15)
                                        .addComponent(blackBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 115,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(fiveMinutesBtn)
                                        .addGap(27, 27, 27)
                                        .addComponent(tenMinutesBtn)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED)
                                        .addComponent(unlimitedBtn))))))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(299, 299, 299)
                            .addComponent(jLabel3))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addContainerGap()
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(jLabel1)
                                .addComponent(jLabel4))
                            .addGap(75, 75, 75)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(gameTitleTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 267,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(opponentNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 267,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                    .addGap(103, 103, 103))
            );
            jPanel1Layout.setVerticalGroup(
                jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jLabel3)
                    .addGap(22, 22, 22)
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
```

```
                    .addComponent(jLabel4)
                    .addComponent(gameTitleTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(opponentNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(24, 24, 24)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(whiteBtn)
                    .addComponent(randomBtn)
                    .addComponent(blackBtn))
                .addGap(22, 22, 22)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(fiveMinutesBtn)
                    .addComponent(tenMinutesBtn)
                    .addComponent(unlimitedBtn))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 24, Short.MAX_VALUE)
                .addComponent(createGameBtn)
                .addGap(5, 5, 5)
                .addComponent(backBtn)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>
```

```
    private void whiteBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only one colour can be choosen
        if(whiteBtn.isSelected()){
            blackBtn.setSelected(false);
            randomBtn.setSelected(false);
        }
    }

    private void randomBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // ensures that only one colour can be choosen
        if(randomBtn.isSelected()){
            blackBtn.setSelected(false);
            whiteBtn.setSelected(false);
        }
    }

    private void blackBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // ensures that only one colour can be choosen
        if(blackBtn.isSelected()){
            whiteBtn.setSelected(false);
            randomBtn.setSelected(false);
        }
    }

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes user back to offline menu
        OfflinePlay offline = new OfflinePlay(session);
        offline.setVisible(true);
        this.dispose();
    }

    private void createGameBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // creates a new game after checking that game settings have
actually been selected;
        //gets opponent's name
        String opponentName=opponentNameTxt.getText();
        //ensures that opponent name is not a computer id
        if(opponentName.equals(OfflinePlay.LEVEL_ONE_COMP)||opponentName
.equals(OfflinePlay.LEVEL_TWO_COMP)||opponentName.equals(OfflinePlay.LEV
EL_THREE_COMP)||opponentName.equals(OfflinePlay.LEVEL_FOUR_COMP)||oppone
ntName.equals(OfflinePlay.LEVEL_FIVE_COMP)||Validation.containsIllegalSt
rings(opponentName)){
            JOptionPane.showMessageDialog(this,"Invalid opponent name.
Please change the opponent name.","Error",JOptionPane.OK_OPTION);
            return;
        }
        boolean isUserWhite=false;//stores which colour the user will
```

```
play as
        if(whiteBtn.isSelected()){
            isUserWhite=true;
        }else if(blackBtn.isSelected()){
            isUserWhite=false;
        }else if(randomBtn.isSelected()){
            //randomly assings user a colour
            if(Math.random()>=0.5){
                isUserWhite=true;
            }else{
                isUserWhite=false;
            }
        }else{
            //error message if no colour chosen
            JOptionPane.showMessageDialog(this,"Please select a colour
to play as.","Error",JOptionPane.OK_OPTION);
            return;
        }
        String gameTitle=gameTitleTxt.getText();
        if(CreateOfflineGame.isGameTitleValid(session,
gameTitle)==false||Validation.isPresent(gameTitle)==false){
            JOptionPane.showMessageDialog(this,"Please select a valid
and unique title.","Error",JOptionPane.OK_OPTION);
            return;
        }
        int timeLeft=-100;
        if(unlimitedBtn.isSelected()){
            timeLeft=-100;
        }else if(fiveMinutesBtn.isSelected()){
            timeLeft=5*60;
        }else if(tenMinutesBtn.isSelected()){
            timeLeft =10*60;
        }else{
            JOptionPane.showMessageDialog(this,"Please select a time
control setting.","Error",JOptionPane.OK_OPTION);
            return;
        }
        CreateOfflineGame.addNewGameToFile(gameTitle, opponentName,
isUserWhite, timeLeft, session);//writes record to file
        //takes user to game
        ChessBoardState[]state={new
ChessBoardState(OfflinePlay.NEW_BOARD)};
        ChessBoardOffline board = new ChessBoardOffline(session,
opponentName, new ChessGame(state,false), isUserWhite, 0, timeLeft,
timeLeft, gameTitle);
        board.setVisible(true);
        this.dispose();
    }

    private void
fiveMinutesBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // makes sure only one time control can be selected
```

```
        if(fiveMinutesBtn.isSelected()){
            tenMinutesBtn.setSelected(false);
            unlimitedBtn.setSelected(false);
        }
    }

    private void tenMinutesBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // makes sure only one time control can be selected
        if(tenMinutesBtn.isSelected()){
            fiveMinutesBtn.setSelected(false);
            unlimitedBtn.setSelected(false);
        }
    }

    private void unlimitedBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // makes sure only one time control can be selected
        if(unlimitedBtn.isSelected()){
            fiveMinutesBtn.setSelected(false);
            tenMinutesBtn.setSelected(false);
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(CreateOfflineGameAgainstP
erson.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(CreateOfflineGameAgainstP
erson.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(CreateOfflineGameAgainstP
```

```
erson.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(CreateOfflineGameAgainstP
erson.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new CreateOfflineGameAgainstPerson().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JToggleButton blackBtn;
    private javax.swing.JButton createGameBtn;
    private javax.swing.JToggleButton fiveMinutesBtn;
    private javax.swing.JTextField gameTitleTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField opponentNameTxt;
    private javax.swing.JToggleButton randomBtn;
    private javax.swing.JToggleButton tenMinutesBtn;
    private javax.swing.JToggleButton unlimitedBtn;
    private javax.swing.JToggleButton whiteBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.Component;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.RenderingHints;
```

```
import java.awt.image.BufferedImage;
import javax.swing.ImageIcon;

/**
 * A custom image icon class that ensures the image icon of chess pieces
of high quality
 * @author mortimer
 */
public class CustomImageIconBoard extends ImageIcon {
    /**
     * Creates an ImageIcon using a buffered image. This image is then
rendered using the highest quality rendering settings
     * @param f The 56 by 56 pixel buffered image.
     */
    public CustomImageIconBoard(BufferedImage f) {
      super(f);
      //sets a high render quality of the ImageIcon
      BufferedImage i= new BufferedImage(56, 56,
BufferedImage.TYPE_INT_ARGB);
      Graphics2D g2d = (Graphics2D)i.getGraphics();
                g2d.setRenderingHint(RenderingHints.KEY_ALPHA_INTERPOLAT
ION, RenderingHints.VALUE_ALPHA_INTERPOLATION_QUALITY);
                g2d.setRenderingHint(RenderingHints.KEY_COLOR_RENDERING,
RenderingHints.VALUE_COLOR_RENDER_QUALITY);
                g2d.setRenderingHint(RenderingHints.KEY_DITHERING,
RenderingHints.VALUE_DITHER_ENABLE);
                g2d.setRenderingHint(RenderingHints.KEY_FRACTIONALMETRIC
S, RenderingHints.VALUE_FRACTIONALMETRICS_ON);
                g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BICUBIC);
                g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2d.setRenderingHint(RenderingHints.KEY_RENDERING,
RenderingHints.VALUE_RENDER_QUALITY);

      g2d.drawImage(getImage(), 0, 0, 56, 56, null);//draws the fifty
six by fifty six ImageIcon
      setImage(i);
    }
    /**
     * Creates a custom image icon that renders an image at the highes
possible quality.
     * @param f the 56 by 56 pixel image
     */
    public CustomImageIconBoard(Image f) {
      super(f);
      //sets a high render quality of the ImageIcon
      BufferedImage i= new BufferedImage(56, 56,
BufferedImage.TYPE_INT_ARGB);
      Graphics2D g2d = (Graphics2D)i.getGraphics();
                g2d.setRenderingHint(RenderingHints.KEY_ALPHA_INTERPOLAT
ION, RenderingHints.VALUE_ALPHA_INTERPOLATION_QUALITY);
                g2d.setRenderingHint(RenderingHints.KEY_COLOR_RENDERING,
```

```
RenderingHints.VALUE_COLOR_RENDER_QUALITY);
                g2d.setRenderingHint(RenderingHints.KEY_DITHERING,
RenderingHints.VALUE_DITHER_ENABLE);
                g2d.setRenderingHint(RenderingHints.KEY_FRACTIONALMETRIC
S, RenderingHints.VALUE_FRACTIONALMETRICS_ON);
                g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BICUBIC);
                g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2d.setRenderingHint(RenderingHints.KEY_RENDERING,
RenderingHints.VALUE_RENDER_QUALITY);

      g2d.drawImage(getImage(), 0, 0, 56, 56, null);
      setImage(i);
    }
    /**
     * Gets the icon's height.
     * @return The icon's height (this will be 56)
     */
    @Override
    public int getIconHeight() {
      return 56;
    }
    /**
     * Gets the icon's width.
     * @return The icon's width (this will be 56)
     */
    @Override
    public int getIconWidth() {
      return 56;
    }
    /**
     * Paints the icon
     * @param c The component
     * @param g The graphics
     * @param x The x value
     * @param y The y value
     */
    @Override
    public void paintIcon(Component c, Graphics g, int x, int y) {
      g.drawImage(getImage(), x, y, c);
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
```

```
 */
package philipm.cs5.software.development.user.system;

import java.awt.Component;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.RenderingHints;
import java.awt.image.BufferedImage;
import javax.swing.ImageIcon;

/**
 * A custom image icon class that ensures the image icon of the speaker
is of high quality
 * @author mortimer
 */
public class CustomImageIconBoardSpeaker extends ImageIcon {
    /**
     * Creates an image icon for the speaker which is of high visual
quality.
     * @param f The image of the speaker (which must be 15 by 15
pixels).
     */
    public CustomImageIconBoardSpeaker(Image f) {
      super(f);
      //sets a high render quality of the ImageIcon
      BufferedImage i= new BufferedImage(15, 15,
BufferedImage.TYPE_INT_ARGB);
      //sets the image to a high rendering quality
      Graphics2D g2d = (Graphics2D)i.getGraphics();
                g2d.setRenderingHint(RenderingHints.KEY_ALPHA_INTERPOLAT
ION, RenderingHints.VALUE_ALPHA_INTERPOLATION_QUALITY);
                g2d.setRenderingHint(RenderingHints.KEY_COLOR_RENDERING,
RenderingHints.VALUE_COLOR_RENDER_QUALITY);
                g2d.setRenderingHint(RenderingHints.KEY_DITHERING,
RenderingHints.VALUE_DITHER_ENABLE);
                g2d.setRenderingHint(RenderingHints.KEY_FRACTIONALMETRIC
S, RenderingHints.VALUE_FRACTIONALMETRICS_ON);
                g2d.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BICUBIC);
                g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
                g2d.setRenderingHint(RenderingHints.KEY_RENDERING,
RenderingHints.VALUE_RENDER_QUALITY);

      g2d.drawImage(getImage(), 0, 0, 15, 15, null);
      setImage(i);
    }
    /**
     * Gets the height of the icon.
     * @return The height of the icon  (which is always 15)
     */
    @Override
```

```
    public int getIconHeight() {
      return 15;
    }
    /**
     * Gets the width of the icon.
     * @return The width of the icon  (which is always 15)
     */
    @Override
    public int getIconWidth() {
      return 15;
    }
    /**
     * Paints the icon
     * @param c The component
     * @param g The graphics
     * @param x The x coordinate
     * @param y The y coordinate
     */
    @Override
    public void paintIcon(Component c, Graphics g, int x, int y) {
      g.drawImage(getImage(), x, y, c);
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import javax.swing.JTable;
import philipm.cs5.software.development.server.system.Fixture;

/**
 * Creates a form that allow users to view the fixtures and results of
the west cross chess club
 * @author mortimer
 */
public class FixturesAndResults extends javax.swing.JFrame {
    private SessionInfo session;//stores the session information
    private static final String[]COLUMN_NAMES=new
String[]{"White","Black","Result"};//stores the column names of the
table
    private String[][][]fixturesAndResults;//stores the fixtures and
results
    private int roundIndex;//stores the round that the user is currently
```

```
viewing
    private boolean haveFixturesBeenGenerated;//stores whether the
fixtures have actually been generated
    /**
     * Creates new form FixturesAndResults
     */
    private FixturesAndResults() {
        initComponents();
    }
    /**
     * Creates new form FixturesAndResults
     * @param session The session information for the current user
     * @throws Exception An exception may occur when retrieving the
fixtures and results from the server
     */
    public FixturesAndResults(SessionInfo session) throws Exception{
        initComponents();
        this.session=session;
        //sets various aspects of the table to make a more intuitve user
experience
        fixturesAndResultsTbl.getTableHeader().setReorderingAllowed(fals
e);
        fixturesAndResultsTbl.setSelectionMode(javax.swing.ListSelection
Model.SINGLE_SELECTION);
        fixturesAndResultsTbl.setDefaultEditor(Object.class, null);
        updateVariablesServer();//retrieves the fixtures and results
from the server
        updateTable();//updates the table based on the data retrieved
from the server
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        fixturesAndResultsTbl = new javax.swing.JTable();
        jLabel1 = new javax.swing.JLabel();
        backBtn = new javax.swing.JButton();
        roundLbl = new javax.swing.JLabel();
        prevRoundBtn = new javax.swing.JButton();
        nextRoundBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Fixtures and Results");
```

```
        fixturesAndResultsTbl.setModel(new
javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null},
                {null, null, null},
                {null, null, null},
                {null, null, null}
            },
            new String [] {
                "White", "Black", "Result"
            }
        ) {
            boolean[] canEdit = new boolean [] {
                true, false, false
            };

            public boolean isCellEditable(int rowIndex, int columnIndex)
{
                return canEdit [columnIndex];
            }
        });
        jScrollPane1.setViewportView(fixturesAndResultsTbl);
        if (fixturesAndResultsTbl.getColumnModel().getColumnCount() > 0)
{
            fixturesAndResultsTbl.getColumnModel().getColumn(0).setResiz
able(false);
            fixturesAndResultsTbl.getColumnModel().getColumn(1).setResiz
able(false);
        }

        jLabel1.setText("Fixtures and Results");

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        roundLbl.setText("Round 1");

        prevRoundBtn.setText("Previous Round");
        prevRoundBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                prevRoundBtnActionPerformed(evt);
            }
        });

        nextRoundBtn.setText("Next Round");
```

```
        nextRoundBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                nextRoundBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(438, 438, 438)
                        .addComponent(jLabel1))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(44, 44, 44)
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 997,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(70, 70, 70)
                        .addComponent(backBtn)
                        .addGap(235, 235, 235)
                        .addComponent(prevRoundBtn)
                        .addGap(69, 69, 69)
                        .addComponent(nextRoundBtn))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(461, 461, 461)
                        .addComponent(roundLbl)))
                .addContainerGap(60, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jLabel1)
                .addGap(3, 3, 3)
                .addComponent(roundLbl)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 454,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.BASELINE)
```

```
                    .addComponent(backBtn)
                    .addComponent(prevRoundBtn)
                    .addComponent(nextRoundBtn)))
        );

        pack();
    }// </editor-fold>

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes user back to the form that they came from
        if(session.getIsClubLeader()){//return leader back to leader
form
            WestCrossChessClubLeader leader = new
WestCrossChessClubLeader(session);
            leader.setVisible(true);
            this.dispose();return;
        }else{//returns member back to member form
            WestCrossChessClubMember member = new
WestCrossChessClubMember(session);
            member.setVisible(true);
            this.dispose();return;
        }
    }

    private void nextRoundBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //displays the next round to the user
        roundIndex++;
        updateTable();
    }

    private void prevRoundBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //displays the previous round to the user
        roundIndex--;
        updateTable();
    }
    /**
     * Requests data from the server and updates variables accordingly
     * @throws Exception An exception may occur when attempting to
retrieve information from the server
     */
    private void updateVariablesServer() throws Exception{
        fixturesAndResults= ThreadClient.getFixturesAndResults();
        haveFixturesBeenGenerated=fixturesAndResults.length!=0;
        roundIndex=0;
    }
    /**
     * Updates the table based on the data stored in the object
     */
    private void updateTable(){
        if(haveFixturesBeenGenerated){//displays the fixtures and
```

```
results
            roundLbl.setText("Round
"+String.valueOf(roundIndex+1));//dispalys which round is being viewed
            //enables/disables the buttons which take the user to the
next and previous round depending on which roudn is being accessed
            prevRoundBtn.setVisible(true);
            prevRoundBtn.setEnabled(roundIndex!=0);
            nextRoundBtn.setEnabled(roundIndex!=(fixturesAndResults.leng
th-1));
            nextRoundBtn.setVisible(true);
            //sets the fixtures and results table to be visible
            jScrollPane1.setEnabled(true);jScrollPane1.setVisible(true);
            fixturesAndResultsTbl.setEnabled(true);fixturesAndResultsTbl
.setVisible(true);
            //sets table
            String[][]tableData=new
String[fixturesAndResults[0][0].length][COLUMN_NAMES.length];
            //in the form: white, black, result
            for(int player=0;player<tableData.length;player++){
                tableData[player][0]=fixturesAndResults[roundIndex][0][p
layer];
                if(tableData[player][0].equals(Fixture.BY_NAME)){//repla
ces a bye player with an empty value
                    tableData[player][0]="";
                }
                tableData[player][1]=fixturesAndResults[roundIndex][1][p
layer];
                if(tableData[player][1].equals(Fixture.BY_NAME)){
                    tableData[player][1]="";
                }
                //converts the game state to a value that makes sense to
the member's viewing the table
                switch (fixturesAndResults[roundIndex][2][player]) {
                    case Fixture.BYE_GAME_STATE:
                        tableData[player][2]="Bye Game";
                        break;
                    case Fixture.GAME_DRAW:
                        tableData[player][2]="Draw";
                        break;
                    case Fixture.GAME_WHITE_WIN:
                        tableData[player][2]="White Win";
                        break;
                    case Fixture.GAME_BLACK_WIN:
                        tableData[player][2]="Black Win";
                        break;
                    default:
                        //game has not yet been played
                        tableData[player][2]="Game Not Yet Played";
                        break;
                }
            }
            //sets the table to the data generated
            JTable tab = new JTable(tableData, COLUMN_NAMES);
```

```
            fixturesAndResultsTbl.setModel(tab.getModel());
            return;
        }else{
            //displays that the fixtures have not been generated and
updates the form's components accordingly
            roundLbl.setText("The fixtures have not yet been
generated");
            prevRoundBtn.setEnabled(false);prevRoundBtn.setVisible(false
);
            nextRoundBtn.setEnabled(false);nextRoundBtn.setVisible(false
);
            fixturesAndResultsTbl.setEnabled(false);fixturesAndResultsTb
l.setVisible(false);
            jScrollPane1.setEnabled(false);jScrollPane1.setVisible(false
);
            return;
        }
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FixturesAndResults.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FixturesAndResults.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FixturesAndResults.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FixturesAndResults.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
```

```
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FixturesAndResults().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JTable fixturesAndResultsTbl;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton nextRoundBtn;
    private javax.swing.JButton prevRoundBtn;
    private javax.swing.JLabel roundLbl;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.IOException;
import javax.swing.DefaultCellEditor;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import philipm.cs5.software.development.server.system.Fixture;

/**
 * Form that allows the leader to edit and view fixtures/results.
 * @author mortimer
 */
public class FixturesAndResultsLeader extends javax.swing.JFrame  {
    private SessionInfo session;//stores the session information
    private static final String[]COLUMN_NAMES=new
String[]{"White","Black","Result"};//stores the name of the columns used
in the table
    private String[][][]fixturesAndResults;//stores the fixtures and
results
    private int roundIndex;//stores which round is currently being
viewed
```

```
    private boolean haveFixturesBeenGenerated;//stores whether the
fixtures have been generated yet
    //constants used to indicate the game result to the leader
    private static final String WHITE_WIN="White Win";private static
final String BLACK_WIN="Black Win";
    private static final String GAME_DRAWN="Draw";private static final
String GAME_NOT_YET_PLAYED="Game Not Yet Played";
    private static final String BYE_GAME="Bye Game";
    private static final String[]GAME_STATE_OPTIONS=new
String[]{GAME_DRAWN,WHITE_WIN,BLACK_WIN,GAME_NOT_YET_PLAYED,BYE_GAME};//
stores the options used by the JComboBox editor which allows the leader
to alter the game state
    /**
     * Creates new form FixturesAndResults
     */
    private FixturesAndResultsLeader() {
        initComponents();
    }
    /**
     * Creates new form FixturesAndResults
     * @param session The session information
     * @throws Exception An exception may occur when attempting to
retrieve the fixtures and results from the server
     */
    public FixturesAndResultsLeader(SessionInfo session) throws
Exception{
        initComponents();
        this.session=session;
        //sets various table characteristics for a better user
experience
        fixturesAndResultsTbl.getTableHeader().setReorderingAllowed(fals
e);
        fixturesAndResultsTbl.setSelectionMode(javax.swing.ListSelection
Model.SINGLE_SELECTION);
        fixturesAndResultsTbl.setDefaultEditor(Object.class, null);
        updateVariablesServer();//updates fixtures and results by
getting the data from the server
        updateTable();//updates the table based on the fixtures and
results
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
```

```
        fixturesAndResultsTbl = new javax.swing.JTable();
        jLabel1 = new javax.swing.JLabel();
        backBtn = new javax.swing.JButton();
        roundLbl = new javax.swing.JLabel();
        prevRoundBtn = new javax.swing.JButton();
        nextRoundBtn = new javax.swing.JButton();
        saveBtn = new javax.swing.JButton();
        startNewSeasonBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Fixtures and Results");

        fixturesAndResultsTbl.setModel(new
javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null},
                {null, null, null},
                {null, null, null},
                {null, null, null}
            },
            new String [] {
                "White", "Black", "Result"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.Object.class, java.lang.Object.class,
java.lang.String.class
            };
            boolean[] canEdit = new boolean [] {
                true, false, false
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex)
{
                return canEdit [columnIndex];
            }
        });
        jScrollPane1.setViewportView(fixturesAndResultsTbl);
        if (fixturesAndResultsTbl.getColumnModel().getColumnCount() > 0)
{
            fixturesAndResultsTbl.getColumnModel().getColumn(0).setResiz
able(false);
            fixturesAndResultsTbl.getColumnModel().getColumn(1).setResiz
able(false);
        }

        jLabel1.setText("Fixtures and Results");
```

```
        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        roundLbl.setText("Round 1");

        prevRoundBtn.setText("Previous Round");
        prevRoundBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                prevRoundBtnActionPerformed(evt);
            }
        });

        nextRoundBtn.setText("Next Round");
        nextRoundBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                nextRoundBtnActionPerformed(evt);
            }
        });

        saveBtn.setText("Save Results");
        saveBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                saveBtnActionPerformed(evt);
            }
        });

        startNewSeasonBtn.setText("Start New Season");
        startNewSeasonBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                startNewSeasonBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLa
```

```
yout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(438, 438, 438)
                        .addComponent(jLabel1))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(44, 44, 44)
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 997,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(70, 70, 70)
                        .addComponent(backBtn)
                        .addGap(235, 235, 235)
                        .addComponent(prevRoundBtn)
                        .addGap(69, 69, 69)
                        .addComponent(nextRoundBtn)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                        .addComponent(saveBtn)
                        .addGap(46, 46, 46)
                        .addComponent(startNewSeasonBtn))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(461, 461, 461)
                        .addComponent(roundLbl)))
                .addContainerGap(47, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jLabel1)
                .addGap(3, 3, 3)
                .addComponent(roundLbl)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 454,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)
                    .addComponent(saveBtn)
                    .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.BASELINE)
                        .addComponent(backBtn)
                        .addComponent(prevRoundBtn)
                        .addComponent(nextRoundBtn)
                        .addComponent(startNewSeasonBtn))))
        );

        pack();
    }// </editor-fold>
```

```
    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes user back to the form that they came from
        if(session.getIsClubLeader()){//takes leader back to leader form
            int confirm = JOptionPane.showConfirmDialog(this, "Would you
like to go back? Any unsaved changes (if there are any) will be lost.",
"Back?", JOptionPane.YES_NO_CANCEL_OPTION);
            if(confirm!=JOptionPane.YES_OPTION){
                return;
            }
            WestCrossChessClubLeader leader = new
WestCrossChessClubLeader(session);
            leader.setVisible(true);
            this.dispose();return;
        }else{
            WestCrossChessClubMember member = new
WestCrossChessClubMember(session);
            member.setVisible(true);
            this.dispose();return;
        }
    }

    private void nextRoundBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //stores any changes made to the fixtures / results and takes
the leader to the next round
        updateFixturesArray();
        roundIndex++;
        updateTable();
    }

    private void prevRoundBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //stores any changes made to the fixtures / results and takes
the leader to the previous round
        updateFixturesArray();
        roundIndex--;
        updateTable();
    }

    private void saveBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //saves the changes (if any) and sends the updated
fixtures/results to the server
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to save the fixtures/results? This means that any changes
you have made will be saved.", "Save?",
JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm!=JOptionPane.YES_OPTION){
            return;
        }
        // saves the fixtures in their current form to the server
```

```
        updateFixturesArray();
        try{
            ThreadClient.updateFixturesAndResults(fixturesAndResults);
        }catch(IOException e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
        }
    }

    private void
startNewSeasonBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //seeks user confirmation and then starts a new season
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to start a new season? Doing so will mean that all
previous data (including fixtures, results and the league table) will be
lost.", "New Season?", JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm!=JOptionPane.YES_OPTION){
            return;
        }
        try{
            ThreadClient.createNewSeason();//creates the new season
            updateVariablesServer();//updates the variables and table
based on the new data
            updateTable();
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
        }

    }
    /**
     * Updates the fixtures array game results based on the JTable. This
only updates the local array.
     */
    private void updateFixturesArray(){
        //loops through all results in the current round and updates the
fixture/results array.
        //this essentially results in any changes to the game states
being saved to the array.
        String result;
        for(int
match=0;match<fixturesAndResults[roundIndex][0].length;match++){
            result=fixturesAndResultsTbl.getValueAt(match,
2).toString();
            //converts result to computer representation of game state
            switch (result) {
                case BLACK_WIN:
                    fixturesAndResults[roundIndex][2][match]=Fixture.GAM
```

```
E_BLACK_WIN;
                    break;
                case WHITE_WIN:
                    fixturesAndResults[roundIndex][2][match]=Fixture.GAM
E_WHITE_WIN;
                    break;
                case GAME_DRAWN:
                    fixturesAndResults[roundIndex][2][match]=Fixture.GAM
E_DRAW;
                    break;
                case GAME_NOT_YET_PLAYED:
                    fixturesAndResults[roundIndex][2][match]=Fixture.GAM
E_NOT_YET_PLAYED;
                    break;
                default:
                    //if game is a bye
                    fixturesAndResults[roundIndex][2][match]=Fixture.BYE
_GAME_STATE;
                    break;
            }
        }
    }
    /**
     * Requests data from the server and updates variables accordingly
     * @throws Exception An exception may occur when attempting to
retrieve information from the server
     */
    private void updateVariablesServer() throws Exception{
        fixturesAndResults= ThreadClient.getFixturesAndResults();//gets
fixtures/results from the server
        haveFixturesBeenGenerated=fixturesAndResults.length!=0;//determi
nes whether the fixtuers have actually been generated. A fixtures array
of length 0 indicates that the fxitures have not been generated.
        roundIndex=0;//sets the round index to 0
    }
    /**
     * Updates the table based on the data stored in the object
     */
    private void updateTable(){
        if(haveFixturesBeenGenerated){
            //displays the fixtyres/results of the current round in the
table
            roundLbl.setText("Round
"+String.valueOf(roundIndex+1));//displays which round number is being
viewed
            //enables/disables the previous and next round buttons based
on whether the previous/next round exists
            prevRoundBtn.setVisible(true);
            prevRoundBtn.setEnabled(roundIndex!=0);
            nextRoundBtn.setEnabled(roundIndex!=(fixturesAndResults.leng
th-1));
            nextRoundBtn.setVisible(true);
            //displays the table
```

```
            jScrollPane1.setEnabled(true);jScrollPane1.setVisible(true);
            fixturesAndResultsTbl.setEnabled(true);fixturesAndResultsTbl
.setVisible(true);
            //uses the fixtures and results array to generate the table
data
            String[][]tableData=new
String[fixturesAndResults[0][0].length][COLUMN_NAMES.length];
            //in the form: white, black, result
            for(int player=0;player<tableData.length;player++){
                tableData[player][0]=fixturesAndResults[roundIndex][0][p
layer];
                if(tableData[player][0].equals(Fixture.BY_NAME)){//bye
player names are replaced with an empty string
                    tableData[player][0]="";
                }
                tableData[player][1]=fixturesAndResults[roundIndex][1][p
layer];
                if(tableData[player][1].equals(Fixture.BY_NAME)){
                    tableData[player][1]="";
                }
                //converts the game state from a computer form to a form
that is understandable by humans
                switch (fixturesAndResults[roundIndex][2][player]) {
                    case Fixture.BYE_GAME_STATE:
                        tableData[player][2]=BYE_GAME;
                        break;
                    case Fixture.GAME_DRAW:
                        tableData[player][2]=GAME_DRAWN;
                        break;
                    case Fixture.GAME_WHITE_WIN:
                        tableData[player][2]=WHITE_WIN;
                        break;
                    case Fixture.GAME_BLACK_WIN:
                        tableData[player][2]=BLACK_WIN;
                        break;
                    default:
                        //game has not yet been played
                        tableData[player][2]=GAME_NOT_YET_PLAYED;
                        break;
                }
            }
            JComboBox <String>game = new
JComboBox<String>(GAME_STATE_OPTIONS);//creates a new ComboBox that
allows the game results to be edited by the leader
            DefaultTableModel model = new DefaultTableModel(tableData,
COLUMN_NAMES){//creates a new table model using the data generated
                /**
                 * Determines whether the cell is editable. Only game
results can be edited. Only game results that contain two human players
can be edited.
                 * @param row The row number of the cell
                 * @param column The column number of the cell
                 * @return Whether the cell is editable
```

```
                 */
                @Override
                public boolean isCellEditable(int row, int column){
                    //a game result can only be altered if the game is
against two human players
                    return !(column==2 && (tableData[row][1].equals("")
|| tableData[row][0].equals("")));
                }
            };
            fixturesAndResultsTbl.setModel(model);//sets the table to
the generated model
            fixturesAndResultsTbl.getColumnModel().getColumn(2).setCellE
ditor(new DefaultCellEditor(game));//sets it so that the game results
are edited using a JComboBox. This avoids the need to validate inputs as
only valid game states can be selected.
            return;
        }else{
            //updates the form to display to the user that the fixtures
have not yet been generated
            roundLbl.setText("The fixtures have not yet been
generated");
            saveBtn.setVisible(false);saveBtn.setEnabled(false);
            prevRoundBtn.setEnabled(false);prevRoundBtn.setVisible(false
);
            nextRoundBtn.setEnabled(false);nextRoundBtn.setVisible(false
);
            fixturesAndResultsTbl.setEnabled(false);fixturesAndResultsTb
l.setVisible(false);
            jScrollPane1.setEnabled(false);jScrollPane1.setVisible(false
);
            return;
        }
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
```

```
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(FixturesAndResultsLeader.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(FixturesAndResultsLeader.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(FixturesAndResultsLeader.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(FixturesAndResultsLeader.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new FixturesAndResultsLeader().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JTable fixturesAndResultsTbl;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton nextRoundBtn;
    private javax.swing.JButton prevRoundBtn;
    private javax.swing.JLabel roundLbl;
    private javax.swing.JButton saveBtn;
    private javax.swing.JButton startNewSeasonBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;


/**
 * Creates a thread which gets the computer move in the chess game. This
```

```
is done as its own thread to allow the form to render whilst the
computer move is retrieved.
 * @author mortimer
 */
public class GetComputerMove extends Thread{
    ChessBoardOffline b;//stores the chess board form
    ChessGame game;//stores the chess game
    int lvl;//stores the computer level
    /**
     * Creates a new object that gets and make the computer move in an
offline chess game.
     * @param gam The chess game object for the game in question
     * @param board The chess board form for which the move is being
retrieved.
     * @param lvl The difficulty level of the computer
     */
    public GetComputerMove(ChessGame gam, ChessBoardOffline board,int
lvl){
        //sets the parameters as object variables
        b=board;game=gam;this.lvl=lvl;
    }
    @Override
    /**
     * Gets the computer move and adds it to the chess board form.
     */
       public void run() {
           try{
            //gets move and updates board
            game.addState(game.getComputerMove(lvl));
            b.currentIndex++;
            b.updateBoard();
            b.playMoveMadeSound();
            b.checkForTerminalState();
           }catch(Exception e){//when checkmate is reached, this method
may still be called

           }
       }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.Serializable;
```

```
import philipm.cs5.software.development.server.system.UserInfo;

/**
 * This class is used to create a package to send to the server. It
creates an object which essentially stores an array. Element 0 of this
 * array is a command which indicates what the server has to do. The
command is a constant and stored in this class.
 * @author mortimer
 */
public class InputPackages implements Serializable {
    private String[]packageToSend;//this stores the package to send
    //these are the constant which are inserted at the nought element of
the array. These commands indicate which task needs to be performed
    public static final String
GET_USER_INFO_WITH_USERNAME_AND_PASSWORD="UserInfWithPassAndUserName9123
";
    public static final String
GET_USER_INFO_WITH_EMAIL_AND_PASSWORD="UserInfWithPassAndEmail3468";
    public static final String CREATE_ACCOUNT="cReateAccount234s";
    public static final String
GET_ONLINE_GAMES_FOR_USER="getOnlinegamedetailsforuser";
    public static final String
CREATE_RANDOM_GAME="newRandomGameRequest189249";//constant that stores
request to create a new random online game
    public static final String
DELETE_GAME="deletegameRequest1224";//stores the constant that signals a
request to delete a game
    public static final String
UPDATED_ONLINE_GAME_RECORD="updateOnlinerecordRequestConst23";//requests
to update online record
    public static final String
CREATE_NEW_GAME_AGAINST_FRIEND="createNewGameAgainstFriendReq3924";
    public static final String
JOIN_GAME_USING_GAME_CODE="joindWithCode9123";
    public static final String GET_GAME_RECORD="getGameRecordWithCode";
    public static final String
GET_CLUB_CONTACT_DETAILS="getClubConatctddata35";
    public static final String
GET_WHETHER_USER_HAS_REQUESTED_TO_JOIN_CLUB="hasUserrequestedtojoinclun1
4";
    public static final String
CHANGE_STATUS_REQUEST_TO_JOIN_CLUB="changeREquestStatsu12984";
    public static final String
REMOVE_USER_FROM_CHESS_CLUB="removeUserFromChessClub124";
    public static final String GET_LEAGUE_TABLE="getELaegy";
    public static final String GET_ALL_CLUB_MEMBERS="getAllMembers";
    public static final String
GET_FIXTURES_AND_RESULTS="fixturesAndRes";
    public static final String
GET_PEOPLE_ADDED_MIDWAY_THROUGH_SEASON="getMidwaypead";
    public static final String
UPDATED_FIXTURES_AND_RESULTS="updateFixtsAndres";
    private String[][][]fixturesAndResults; //stores the fixtures and
```

```
results if that is what is being sent to the server
    public static final String CREATE_NEW_SEASON="newSeasonas";
    public static final String SET_CONTACT_DETAILS="setContactdakl";
    public static final String SET_USER_AS_LEADER="setLEaderaf";
    public static final String GET_ALL_REQUESTS="alLRequessf";
    public static final String GET_EMAIL_OF_USER="getEmailUseraf";
    public static final String
REMOVE_REQUEST_JOIN_CLUB="setUserRequstClubFalse";
    public static final String ADD_USER_TO_CLUB="addUserToClub";
    public static final String
GET_CLUB_MEMBER_DETAILS_ALL_MEMBERS="getAllClubmeiaefd";
    public static final String
UPDATE_USER_INFO_FILE_RECORD="updahusasf";
    public static final String REMOVE_NON_MEMBER_USER="sausdf";
    public static final String GET_ALL_NON_MEMBERS="getAllNonMebrsfd";
    /**
     * Creates an input package to get user info by providing log in
details
     * @param userName The username
     * @param password The password
     */
    public void createLogInRequestForWithUserNameAndPassword(String
userName,String password){
        packageToSend= new String[3];
        packageToSend[0]=GET_USER_INFO_WITH_USERNAME_AND_PASSWORD;packag
eToSend[1]=userName;packageToSend[2]=password;
    }
    /**
     * Creates an InputPackage object
     */
    public InputPackages(){

    }
    /**
     * Creates package that requests the fixtures and results
     */
    public void getFixturesAndResults(){
        packageToSend=new
String[1];packageToSend[0]=GET_FIXTURES_AND_RESULTS;
    }
    /**
     * Creates package that requests the the list of people added midway
through the season.
     */
    public void getPeopleAddedMidwayThroughSeason(){
        packageToSend=new
String[1];packageToSend[0]=GET_PEOPLE_ADDED_MIDWAY_THROUGH_SEASON;
    }
    /**
     * Creates package that requests the fixtures and results be updated
to the inputted fixture and results.
     * @param fixturesAndResults The new fixtures and results.
     */
```

```
    public void
updateFixturesAndResults(String[][][]fixturesAndResults){
        packageToSend=new String[]{UPDATED_FIXTURES_AND_RESULTS};
        this.fixturesAndResults=fixturesAndResults;
    }
    /**
     * Returns the fixtures and results array stored in the package
     * @return The fixtures and results array
     */
    public String[][][]getFixturesAndResultsArray(){
        return this.fixturesAndResults;
    }
    /**
     * Creates a package that requests to add a specified user to the
chess club.
     * @param userName The username of the user to add.
     */
    public void addUserToClubAsMember(String userName){
        packageToSend=new String[]{ADD_USER_TO_CLUB,userName};
    }
    /**
     * Creates package that requests that a new season is started on the
server.
     */
    public void startNewSeason(){
        packageToSend=new String[]{CREATE_NEW_SEASON};
    }
    /**
     * Creates a package that requests the server alter the club contact
details.
     * @param phoneNumber The new phone number
     * @param emailAddress The new email address
     */
    public void setContactDetails(String phoneNumber, String
emailAddress){
        packageToSend=new
String[]{SET_CONTACT_DETAILS,phoneNumber,emailAddress};
    }
    /**
     * Creates package that requests for the server to make the chosen
user the club leader
     * @param userName The name of the user
     */
    public void setClubLeader(String userName){
        packageToSend=new String[]{SET_USER_AS_LEADER,userName};
    }
    /**
     * Creates a package that requests the user details of all users who
have requested to join the club.
     */
    public void getAllRequestsToJoinClub(){
        packageToSend=new String[]{GET_ALL_REQUESTS};
    }
```

```
    /**
     * Creates a package that requests the email address of a chosen
user
     * @param userName The username of the chosen user
     */
    public void getEmailAddressOfUser(String userName){
        packageToSend=new String[]{GET_EMAIL_OF_USER,userName};
    }
    /**
     * Creates a package that requests that the server give the client
the user details of all non members.
     */
    public void getAllNonMembers(){
        packageToSend=new String[]{GET_ALL_NON_MEMBERS};
    }
    /**
     * Creates a new input package which requests all online game data
on a user
     * @param userName The username
     */
    public void createRequestForAllGameDataForAUser(String userName){
        packageToSend=new String[2];
        packageToSend[1]=userName;
        packageToSend[0]=GET_ONLINE_GAMES_FOR_USER;
    }
    /**
     * Creates a package that tells the server to withdraw the request
to join the club from a given user.
     * @param userName The username of the user to withdraw the request
from.
     */
    public void withdrawRequestToJoinClub(String userName){
        packageToSend=new String[]{REMOVE_REQUEST_JOIN_CLUB,userName};
    }
    /**
     * Creates package that requests the user information on all club
members.
     */
    public void getAllClubMembers(){
        packageToSend=new
String[1];packageToSend[0]=GET_ALL_CLUB_MEMBERS;
    }
    /**
     * Creates an input package to get user info by providing log in
details
     * @param emailAddress The email Address
     * @param password The password
     */
    public void createLogInRequestForWithEmailAndPassword(String
emailAddress,String password){
        packageToSend= new String[3];
        packageToSend[0]=GET_USER_INFO_WITH_EMAIL_AND_PASSWORD;packageTo
Send[1]=emailAddress;packageToSend[2]=password;
```

```
    }
    /**
     * Creates a new package that requests for an online game to be
created
     * @param userName The username of the user who wishes to play a new
random online game
     */
    public void createRandomGamePackage(String userName){
        packageToSend=new String[2];
        packageToSend[0]=CREATE_RANDOM_GAME;packageToSend[1]=userName;
    }
    /**
     * Creates a package that requests all data on club members.
     */
    public void getAllClubMembersInfo(){
        packageToSend=new String[]{GET_CLUB_MEMBER_DETAILS_ALL_MEMBERS};
    }
    /**
     * Creates a package to tell server to update a user record.
     * @param newRecord The new record
     */
    public void updateUserRecord(String[]newRecord){
        packageToSend=new String[UserInfo.WIDTH_OF_USER_DETAILS_FILE+1];
        packageToSend[0]=UPDATE_USER_INFO_FILE_RECORD;
        for(int i=0;i<UserInfo.WIDTH_OF_USER_DETAILS_FILE;i++){
            packageToSend[i+1]=newRecord[i];
        }
    }
    /**
     * Gets the package array
     * @return The string array storing the action
     */
    public String[]getPackageToSend(){
        return packageToSend;
    }
    /**
     * Tells server to remove non club member from system.
     * @param userName The name of the user account to delete
     */
    public void removeNonMemberUserFromRecord(String userName){
        packageToSend=new String[]{REMOVE_NON_MEMBER_USER,userName};
    }
    /**
     * Returns the action name
     * @return The nought element of the package to send array is
returned
     */
    public String getActionName(){
        return packageToSend[0];
    }
    /**
     * Creates a package that requests for the league table
     */
```

```
    public void getLeagueTableRequest(){
        packageToSend=new String[1];
        packageToSend[0]=GET_LEAGUE_TABLE;
    }
    /**
     * Creates a package that requests the server to add a new user
record.
     * @param userRecord The user record array
     * @param userName The username of the user
     * @param emailAddress The email address
     */
    public void createAccountPacket(String []userRecord,String userName,
String emailAddress){
        packageToSend=new String[userRecord.length+3];
        packageToSend[0]=CREATE_ACCOUNT;
        for(int i=0;i<userRecord.length;i++){
            packageToSend[i+1]=userRecord[i];
        }
        packageToSend[userRecord.length+1]=userName;
        packageToSend[userRecord.length+2]=emailAddress;
    }
    /**
     * Returns the user record. Note: this only works if the command is
actually for creating an account.
     * @return The user record.
     */
    public String[]getUserRecod(){
        String[] rec = new String [UserInfo.WIDTH_OF_USER_DETAILS_FILE];
        for(int i=1;i<UserInfo.WIDTH_OF_USER_DETAILS_FILE+1;i++){
            rec[i-1]=packageToSend[i];
        }
        return rec;
    }
    /**
     * Returns the user name. Must be called only when the input package
is to create an account.
     * @return The username
     */
    public String getUserName(){
        return packageToSend[UserInfo.WIDTH_OF_USER_DETAILS_FILE+1];
    }
    /**
     * Returns the email address. Must be called only when the input
package is to create an account.
     * @return The email address
     */
    public String getEmail(){
        return packageToSend[UserInfo.WIDTH_OF_USER_DETAILS_FILE+2];
    }
    /**
     * Creates a copy of the current input package
     * @return A copy of the package
     */
```

```
    public InputPackages getCopyOfInputPackage(){
        String oldArray[]=this.getPackageToSend();
        String newArray[]=new String[oldArray.length];//copies previous
array
        for(int i=0;i<newArray.length;i++){
            newArray[i]=oldArray[i];
        }
        return new InputPackages(newArray);
    }
    private InputPackages(String[]messageArray){
        packageToSend=messageArray;
    }
    /**
     * Creates a packet used to request that a game is deleted
     * @param gameId The game id of the game to be deleted
     * @param gameResult The game result: 1 signifies a win for player
one. 0.5 indicates a draw and 0 indicates a player two win
     */
    public void createRequestToDeleteGame(String gameId, double
gameResult){
        packageToSend=new
String[3];packageToSend[0]=DELETE_GAME;packageToSend[1]=gameId;packageTo
Send[2]=String.valueOf(gameResult);
    }
    /**
     * Updates the online record
     * @param newOnlineGameRecord The updated record
     */
    public void createRequestToUpdateOnlineGame(String
[]newOnlineGameRecord){
        packageToSend=new String[1+newOnlineGameRecord.length];
        packageToSend[0]=UPDATED_ONLINE_GAME_RECORD;
        for(int i=0;i<newOnlineGameRecord.length;i++){
            packageToSend[i+1]=newOnlineGameRecord[i];
        }
    }
    /**
     * Creates package that tells the server to tell the client whether
the chosen user has requested to join the club.
     * @param userName The name of the user
     */
    public void getWhetherUserHasRequestedToJoinClub(String userName){
        packageToSend=new
String[2];packageToSend[0]=GET_WHETHER_USER_HAS_REQUESTED_TO_JOIN_CLUB;p
ackageToSend[1]=userName;
    }
    /**
     * Creates a package that requests that the server changes the
request status of a user.
     * @param userName The name of the user
     */
    public void changeRequestToJoinClubStatus(String userName){
        packageToSend=new
```

```
String[2];packageToSend[0]=CHANGE_STATUS_REQUEST_TO_JOIN_CLUB;packageToS
end[1]=userName;
    }
    /**
     * Creates a package that requests that the chosen user is removed
from the chess club.
     * @param userName The name of the user
     */
    public void removeUserFromChessClub(String userName){
        packageToSend=new
String[2];packageToSend[0]=REMOVE_USER_FROM_CHESS_CLUB;packageToSend[1]=
userName;
    }
    /**
     * Creates a request to get the contact info of the chess club
     */
    public void requestClubContactInfo(){
        packageToSend=new
String[1];packageToSend[0]=GET_CLUB_CONTACT_DETAILS;
    }
    /**
     * Creates a new request to play an online game against a friend
     * @param userNamePlayerOne the username of the first player
     * @param userNamePlayerTwo The second username or "" if the game is
to be played by sharing a code
     * @param timeLeftForBoth The time left for both players
     */
    public void createNewOnlineGameWithFriend(String userNamePlayerOne,
String userNamePlayerTwo,String timeLeftForBoth){
        packageToSend=new
String[4];packageToSend[0]=CREATE_NEW_GAME_AGAINST_FRIEND;packageToSend[
1]=userNamePlayerOne;
        packageToSend[2]=userNamePlayerTwo;packageToSend[3]=timeLeftForB
oth;
    }
    /**
     * Attempts to join a game using the provided game code
     * @param gameCode The game code
     * @param userName The username of the player
     */
    public void joinGameUsingGameCode(int gameCode,String userName){
        packageToSend=new
String[3];packageToSend[0]=JOIN_GAME_USING_GAME_CODE;packageToSend[1]=St
ring.valueOf(gameCode);packageToSend[2]=userName;
    }
    /**
     * Creates a request to get game data
     * @param gameCode The game id
     */
    public void getGameRecord(int gameCode){
        packageToSend=new
String[2];packageToSend[0]=GET_GAME_RECORD;packageToSend[1]=String.value
Of(gameCode);
```

```
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import javax.swing.JTable;


/**
 * This form is used to display the league table in the West Cross Chess
Club
 * @author mortimer
 */
public class LeagueTable extends javax.swing.JFrame {
    private SessionInfo session;//stores the session information
    private static final String[]COLUMN_NAMES=new
String[]{"Position","Username","Wins","Draws","Losses","Points"};//store
s the name of the columns for the league table
    /**
     * Creates new form LeagueTable
     */
    private LeagueTable() {
        initComponents();
    }
    /**
     * Creates a form to view the league table. This form is created
when the fixtures for the club have been generated.
     * @param session The session info
     * @param leagueTableData The league table data
     */
    public LeagueTable(SessionInfo session,String[][]leagueTableData){
        this.session= session;
        initComponents();
        //sets various details of the table to allow for a clean and
intuitive user experience
        leagueTable.getTableHeader().setReorderingAllowed(false);
        leagueTable.setSelectionMode(javax.swing.ListSelectionModel.SING
LE_SELECTION);
        leagueTable.setDefaultEditor(Object.class, null);
        setTable(leagueTableData);//sets the league table based on the
league table data
    }
    /**
```

```
     * Creates a form to view the league table. This form is created
when the fixtures for the club have not yet been generated.
     * @param session The session info
     * @param allClubMembers The league table data
     */
    public LeagueTable(SessionInfo session,String[]allClubMembers){
        this.session= session;
        initComponents();
        //sets various details of the table to allow for a clean and
intuitive user experience
        leagueTable.getTableHeader().setReorderingAllowed(false);
        leagueTable.setSelectionMode(javax.swing.ListSelectionModel.SING
LE_SELECTION);
        leagueTable.setDefaultEditor(Object.class, null);
        setTable(allClubMembers);//sets the table
    }
    /**
     * Sets the league table based on the league table data
     * @param leagueTableData The league table data
     */
    private void setTable(String[][]leagueTableData){
        String[][]leagueDataFormatted=new
String[leagueTableData.length][6];//stores the formatted data, which is
used to display the table
        int pos=1; //stores the position
        String prevPoints="";//stores the points tally of the player
directly above a certain user
        for(int player=0;player<leagueTableData.length;player++){//loops
through every player in the league table and calculates their position
in the table. Players who have the same number of points are assigned
the same position in the table
            if(player==0){
                prevPoints=String.valueOf(leagueTableData[player][4]);
                //copies various attributes from the league table data
in the appropriate format
                leagueDataFormatted[player][0]=String.valueOf(pos);
                leagueDataFormatted[player][1]=leagueTableData[player][0
];
                leagueDataFormatted[player][2]=leagueTableData[player][1
];leagueDataFormatted[player][3]=leagueTableData[player][2];
                leagueDataFormatted[player][4]=leagueTableData[player][3
];leagueDataFormatted[player][5]=leagueTableData[player][4];
            }else{
                if(!prevPoints.equals(leagueTableData[player][4])){
                    pos++;//increments the position if the points of
this player are not the same as the points of the previous player
                }
                prevPoints=String.valueOf(leagueTableData[player][4]);
                leagueDataFormatted[player][0]=String.valueOf(pos);
                //copies various attributes from the league table data
in the appropriate format
                leagueDataFormatted[player][1]=leagueTableData[player][0
];
```

```
                leagueDataFormatted[player][2]=leagueTableData[player][1
];leagueDataFormatted[player][3]=leagueTableData[player][2];
                leagueDataFormatted[player][4]=leagueTableData[player][3
];leagueDataFormatted[player][5]=leagueTableData[player][4];
            }
        }
        JTable tab = new JTable(leagueDataFormatted, COLUMN_NAMES);
        leagueTable.setModel(tab.getModel());//sets the league table to
the formatted data
    }
    /**
     * Sets the league table based on the members of the club
     * @param allClubMembers The club members
     */
    private void setTable(String[]allClubMembers){
        //this method is called when the fixtures have not yet been
generated. Hence, the table just displays all users as being in first
place and having
        //0 points
        String [][]tableData=new String[allClubMembers.length][6];
        for(int i=0;i<tableData.length;i++){
            tableData[i][0]="1";tableData[i][1]=allClubMembers[i];
            tableData[i][2]="0";tableData[i][3]="0";tableData[i][4]="0";
            tableData[i][5]="0";
        }
        JTable tab = new JTable(tableData, COLUMN_NAMES);
        leagueTable.setModel(tab.getModel());//sets the table to the
formatted value
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        backBtn = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        leagueTable = new javax.swing.JTable();
        jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("League Table");

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        leagueTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        jScrollPane1.setViewportView(leagueTable);

        jLabel1.setText("League Table");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 673, Short.MAX_VALUE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(35, 35, 35)
                                .addComponent(backBtn))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(325, 325, 325)
                                .addComponent(jLabel1)))
                        .addGap(0, 0, Short.MAX_VALUE)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 480,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(1, 1, 1)
                .addComponent(backBtn)
                .addGap(24, 24, 24))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user back to the chess club form they came from
        if(session.getIsClubLeader()){
            //takes the user back to the club leader form
            WestCrossChessClubLeader leader = new
WestCrossChessClubLeader(session);
            leader.setVisible(true);this.dispose();
        }else{
            //takes the user back to the member form
            WestCrossChessClubMember memberForm = new
WestCrossChessClubMember(session);
            memberForm.setVisible(true);this.dispose();
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
```

```
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(LeagueTable.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(LeagueTable.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(LeagueTable.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(LeagueTable.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LeagueTable().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable leagueTable;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
```

```
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;
import philipm.cs5.software.development.server.system.UserInfo;

/**
 * Creates a form that allows the user to log in with their log in
credentials
 * @author mortimer
 */
public class LogIn extends javax.swing.JFrame {
    private boolean showPassword=false;//stores whether the password
should be masked or not
    private final char DEFAULT_ECHO_CHAR;//stores tge default character
used to mask the password
    private static final String
TERMS_AND_CONS_FILE="haveTermsAndConditionsBeenAccepted.txt";//stores
the text file location that stores whether the terms and conditions have
been agreed to
    /**
     * Creates new form LogIn
     */
    public LogIn() {
        initComponents();
        try{
            //checks to see if terms and conditions have already been
agreed to
            FileReader read = new FileReader(TERMS_AND_CONS_FILE);
            BufferedReader buffReader = new BufferedReader(read);
            boolean termsAccepted=buffReader.readLine().equals("true");
            buffReader.close();read.close();
            if(!termsAccepted){
                //displays the terms and conditions of using the
software
                JOptionPane.showMessageDialog(this, "This program makes
use of the Stockfish 12 chess engine. This program is licensed to use
Stockfish 12 under the GNU General Public License Version 3 (GPLv3)."
                +System.lineSeparator()+"A copy of this license can be
found in the directory for this program titled 'GNU.txt'. The source
code to Stockfish 12 may be found at: "
                +System.lineSeparator()+"https://github.com/official-
stockfish/Stockfish or https://stockfishchess.org/download/ or
https://stockfishchess.org/"
                        +System.lineSeparator()+"Additionally, the
```

```
source code can be found in the program directory in a zip file titled
'Stockfish-sf_12.zip'"
                +System.lineSeparator()+"This message will not be
displayed again. Please do NOT alter any text files in the program
repository."+
                        System.lineSeparator()+"For any further
assistance, please contact philipmortimer99@aol.com","Message"
                        +System.lineSeparator()+"Furthermore, when
creating an account, all data (apart from your password) can be viewed
and altered by"
                        +System.lineSeparator()+"the leader of the West
Cross Chess Club. By using this software, you consent that you are ok
with this."
                        ,JOptionPane.OK_OPTION);
            }
            //sets it so that this message is not displayed again
            FileWriter write= new FileWriter(TERMS_AND_CONS_FILE);
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write("true");
            buffWrite.flush();write.flush();buffWrite.close();write.clos
e();
        }catch(IOException e){
            JOptionPane.showMessageDialog(this,"An unexpected error
occurred: "+e,"Error",JOptionPane.OK_OPTION);
        }
        DEFAULT_ECHO_CHAR=passwordTxt.getEchoChar();
        this.getRootPane().setDefaultButton(logInBtn);//sets the log in
button as the default button
    }

    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        usernameOrEmailAddressTxt = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        passwordTxt = new javax.swing.JPasswordField();
        showOrHidePasswordBtn = new javax.swing.JButton();
        changeInternetSettingsBtn = new javax.swing.JButton();
        logInBtn = new javax.swing.JButton();
        exitBtn = new javax.swing.JButton();
        guestLogInBtn = new javax.swing.JButton();
        createAccountBtn = new javax.swing.JButton();
```

```
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Log In");

        jLabel1.setText("Username / Email Address");

        jLabel2.setText("Password");

        showOrHidePasswordBtn.setText("Show Password");
        showOrHidePasswordBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                showOrHidePasswordBtnActionPerformed(evt);
            }
        });

        changeInternetSettingsBtn.setText("Change Online Settings");
        changeInternetSettingsBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                changeInternetSettingsBtnActionPerformed(evt);
            }
        });

        logInBtn.setText("Log In");
        logInBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                logInBtnActionPerformed(evt);
            }
        });

        exitBtn.setText("Exit");
        exitBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                exitBtnActionPerformed(evt);
            }
        });

        guestLogInBtn.setText("Log In As Guest");
        guestLogInBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                guestLogInBtnActionPerformed(evt);
            }
        });

        createAccountBtn.setText("Create Account");
```

```
        createAccountBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                createAccountBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING, false)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel1)
                            .addComponent(jLabel2))
                        .addGap(43, 43, 43)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(usernameOrEmailAddressTxt)
                            .addComponent(passwordTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 309, Short.MAX_VALUE))
                        .addGap(84, 84, 84))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(29, 29, 29)
                        .addComponent(exitBtn)
                        .addGap(31, 31, 31)
                        .addComponent(changeInternetSettingsBtn)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, 31, Short.MAX_VALUE)
                        .addComponent(createAccountBtn)
                        .addGap(18, 18, 18)
                        .addComponent(guestLogInBtn)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)))
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addComponent(showOrHidePasswordBtn)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addComponent(logInBtn)
                        .addGap(18, 18, 18)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
```

```
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(45, 45, 45)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(usernameOrEmailAddressTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(45, 45, 45)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel2)
                    .addGroup(jPanel1Layout.createParallelGroup(javax.sw
ing.GroupLayout.Alignment.BASELINE)
                        .addComponent(passwordTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(showOrHidePasswordBtn)))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 165, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(changeInternetSettingsBtn)
                    .addComponent(logInBtn)
                    .addComponent(exitBtn)
                    .addComponent(guestLogInBtn)
                    .addComponent(createAccountBtn))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>
/**
```

```
 * Ensures the password field is masked or not depending on the option
selected.
 * @param evt The button being pressed
 */
    private void
showOrHidePasswordBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //hides or makes the password visible
        showPassword=!showPassword;
        if(showPassword){
            //makes the password visible as plain text
            showOrHidePasswordBtn.setText("Hide Password");
            char c=0;
            passwordTxt.setEchoChar(c);
        }else{
            //masks the password
            showOrHidePasswordBtn.setText("Show Password");
            passwordTxt.setEchoChar(DEFAULT_ECHO_CHAR);
        }
    }
    /**
     * Exits the system after asking for user confirmation.
     * @param evt The exit button being pressed by the user
     */
    private void exitBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to exit?", "Exit?", JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            System.exit(0);
        }
    }
    /**
     * Takes the user to a form that allows them to edit their online
settings.
     * @param evt The user pressing the appropriate button
     */
    private void
changeInternetSettingsBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        ChangeInternetSettings change = new ChangeInternetSettings();
        change.setVisible(true);
        this.dispose();
    }

    private void logInBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //logs user in via server
        //validates input somewhat before querying server
        if(usernameOrEmailAddressTxt.getText().equals(UserInfo.GUEST_ID)
||
usernameOrEmailAddressTxt.getText().equals(UserInfo.GUEST_ID_EMAIL)){//b
anned inputs reserved for guest ids
```

```
            JOptionPane.showMessageDialog(this,"You can't log in with
this ID.","Error",JOptionPane.OK_OPTION);
            return;
        }
        if(usernameOrEmailAddressTxt.getText().contains("@")){//for
email address log in
            try {
                //checks to see if email address exists
                if(ThreadClient.isEmailAddressAlreadyTaken(usernameOrEma
ilAddressTxt.getText())==false){
                    JOptionPane.showMessageDialog(this, "The email
address was not found.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                //attempts to log user in
                UserInfo user =
ThreadClient.logInWithEmailAddress(usernameOrEmailAddressTxt.getText(),
new String(passwordTxt.getPassword()));
                if(user.getIsPasswordCorrect()==false){//if password
does not match email
                    JOptionPane.showMessageDialog(this, "The password is
incorrect.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                //takes user to menu if log in was succesful
                Menu menu = new Menu(new SessionInfo(user.getUserName(),
user.getIsClubMember(),
user.getIsClubLeader()));menu.setVisible(true);this.dispose();return;//t
akes user to menu with appropriate session information
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(this, "A network error has
occured. This may be due to your network settings or the server.
"+ex,"Error",JOptionPane.OK_OPTION);
                return;
            }
        }else{//for username
            try{
                //attempts to log user in with username
                UserInfo user =
ThreadClient.logInWithUserName(usernameOrEmailAddressTxt.getText(), new
String(passwordTxt.getPassword()));//gets user info from server
                if(user.getUserFound()==false){//displays appropraite
message if the username was not found
                    JOptionPane.showMessageDialog(this, "The username
was not found.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                if(user.getIsPasswordCorrect()==false){//displays
appropraite message if the password entered was wrong
                    JOptionPane.showMessageDialog(this, "The password is
incorrect.","Error",JOptionPane.OK_OPTION);
                    return;
                }
```

```
                Menu menu = new Menu(new SessionInfo(user.getUserName(),
user.getIsClubMember(),
user.getIsClubLeader()));menu.setVisible(true);this.dispose();return;//t
akes user to menu with appropriate session information
                //takes user to menu having created session info object
to store relevant data
            }catch(Exception e){
                JOptionPane.showMessageDialog(this, "A network error has
occured. This may be due to your network settings or the server.
"+e,"Error",JOptionPane.OK_OPTION);
                return;
            }
        }
    }

    private void
createAccountBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes user to form which allows them to create an account
        CreateAccount account = new CreateAccount();//loads the create
account form
        account.setVisible(true);
        this.dispose();
    }

    private void guestLogInBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        Menu menu = new
Menu(SessionInfo.createGuestSession());menu.setVisible(true);this.dispos
e();
        //logs user in as guest and takes them to main menu
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
```

```
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(LogIn.class.getName()).lo
g(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(LogIn.class.getName()).lo
g(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(LogIn.class.getName()).lo
g(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(LogIn.class.getName()).lo
g(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LogIn().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton changeInternetSettingsBtn;
    private javax.swing.JButton createAccountBtn;
    private javax.swing.JButton exitBtn;
    private javax.swing.JButton guestLogInBtn;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton logInBtn;
    private javax.swing.JPasswordField passwordTxt;
    private javax.swing.JButton showOrHidePasswordBtn;
    private javax.swing.JTextField usernameOrEmailAddressTxt;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.Color;
import javax.swing.JOptionPane;
```

```
import philipm.cs5.software.development.server.system.UserInfo;


/**
 * This form allows the chess club leader to view and edit the details
of chess club members. It also allows the leader to remove users from
the chess club.
 * @author mortimer
 */
public class MemberDetails extends javax.swing.JFrame {

    private SessionInfo session;//stores the session information
    private String[][]clubMembersInfo;//stores the data of club members
the club
    private String[]emails;//stores the email addresses of the user
    private int clubMembersIndex=0;//stores the index that indicates
which club member's info is currently being viewed
    private boolean isUserInfoBeingEdited=false;//stores whether the
information of the current user is being edited
    /**
     * Creates new form MemberDetails
     * @param session The session information
     * @throws java.lang.Exception An exception may occur when
attempting to load info.
     */
    public MemberDetails(SessionInfo session) throws Exception{
        initComponents();
        this.session=session;
        clubMembersInfo=ThreadClient.getAllClubMemberUserInfo();//gets
the data stored in the server on club members
        emails = new String[clubMembersInfo.length];//gets the email
addressesof club members
        //sets the look and feel of the form
        removeBtn.setBackground(Color.red);
        prevBtn.setEnabled(false);nextBtn.setEnabled(clubMembersIndex!=c
lubMembersInfo.length-1);
        for(int req=0;req<emails.length;req++){
            emails[req]=ThreadClient.getEmailAddressOfUser(clubMembersIn
fo[req][0]);
        }
        loadFormBasedOnIndex();//loads the appropraite user to the form
    }
    /**
     * Creates new form MemberDetails
     */
    private MemberDetails(){
        initComponents();
    }
    /**
     * Loads user info onto the form based on the index (i.e. based on
which user is currently being viewed).
     */
    private void loadFormBasedOnIndex(){
```

```
        clearForm();//clears the form
        //determines whether the buttons to naviagte between users
should be enabled or not.
        prevBtn.setEnabled(true);nextBtn.setEnabled(true);
        if(clubMembersIndex==0){
            prevBtn.setEnabled(false);
        }
        if(clubMembersIndex==clubMembersInfo.length-1){
            nextBtn.setEnabled(false);
        }
        //ensuers that the leader can't remove themselves from the club
        removeBtn.setEnabled(true);
        if(clubMembersInfo[clubMembersIndex][0].equals(session.getUserna
me())){
            removeBtn.setEnabled(false);
        }
        //Username, password, first name, surname, postcode, addressline
one, address line two, address line three, address line four, discord
username, discord number, date of birth, phone number, isClubLeader,
isClubMember, ELO, hasUserRequestedToJoinClub
        //sets the value of the various fields based on the user
informtion provided
        usernameTxt.setText(clubMembersInfo[clubMembersIndex][0]);
        emailAddressTxt.setText(emails[clubMembersIndex]);
        firstNameTxt.setText(clubMembersInfo[clubMembersIndex][2]);surna
meTxt.setText(clubMembersInfo[clubMembersIndex][3]);
        postcodeTxt.setText(clubMembersInfo[clubMembersIndex][4]);addres
sLineOneTxt.setText(clubMembersInfo[clubMembersIndex][5]);
        addressLineTwoTxt.setText(clubMembersInfo[clubMembersIndex][6]);
addressLineThreeTxt.setText(clubMembersInfo[clubMembersIndex][7]);
        addressLineFourTxt.setText(clubMembersInfo[clubMembersIndex][8])
;discordUsernameTxt.setText(clubMembersInfo[clubMembersIndex][9]);
        discordNumberTxt.setText(clubMembersInfo[clubMembersIndex][10]);
dateOfBirthTxt.setText(clubMembersInfo[clubMembersIndex][11]);
        phoneNumberTxt.setText(clubMembersInfo[clubMembersIndex][12]);el
oTxt.setText(clubMembersInfo[clubMembersIndex][15]);
    }
    /**
     * Sets all fields on the form empty
     */
    private void clearForm(){
        usernameTxt.setText("");emailAddressTxt.setText("");
        firstNameTxt.setText("");surnameTxt.setText("");postcodeTxt.setT
ext("");
        eloTxt.setText("");addressLineOneTxt.setText("");addressLineTwoT
xt.setText("");
        addressLineThreeTxt.setText("");addressLineFourTxt.setText("");
        discordUsernameTxt.setText("");discordNumberTxt.setText("");
        dateOfBirthTxt.setText("");phoneNumberTxt.setText("");
    }
    /**
     * This method is called from within the constructor to initialize
the form.
```

```
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        usernameTxt = new javax.swing.JTextField();
        emailAddressTxt = new javax.swing.JTextField();
        firstNameTxt = new javax.swing.JTextField();
        surnameTxt = new javax.swing.JTextField();
        postcodeTxt = new javax.swing.JTextField();
        addressLineOneTxt = new javax.swing.JTextField();
        addressLineTwoTxt = new javax.swing.JTextField();
        addressLineThreeTxt = new javax.swing.JTextField();
        addressLineFourTxt = new javax.swing.JTextField();
        discordUsernameTxt = new javax.swing.JTextField();
        discordNumberTxt = new javax.swing.JTextField();
        dateOfBirthTxt = new javax.swing.JTextField();
        phoneNumberTxt = new javax.swing.JTextField();
        backBtn = new javax.swing.JButton();
        editBtn = new javax.swing.JButton();
        passwordLabel = new javax.swing.JLabel();
        confirmPasswordLabel = new javax.swing.JLabel();
        firstNameLabel = new javax.swing.JLabel();
        surnameLabel = new javax.swing.JLabel();
        postcodeLabel = new javax.swing.JLabel();
        addressOneLabel = new javax.swing.JLabel();
        addressTwoLabel = new javax.swing.JLabel();
        addressThreeLabel = new javax.swing.JLabel();
        addressFourLabel = new javax.swing.JLabel();
        discordUsernameLabel = new javax.swing.JLabel();
        discordNumberLabel = new javax.swing.JLabel();
        dateOfBirthLabel = new javax.swing.JLabel();
        phoneNumberLabel = new javax.swing.JLabel();
        jLabel20 = new javax.swing.JLabel();
        eloTxt = new javax.swing.JTextField();
```

```
        removeBtn = new javax.swing.JButton();
        nextBtn = new javax.swing.JButton();
        prevBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Member Details");

        jLabel1.setText("Username");

        jLabel2.setText("Email Address");

        jLabel5.setText("First Name");

        jLabel6.setText("Surname");

        jLabel7.setText("Postcode");

        jLabel8.setText("Address Line 1");

        jLabel9.setText("Address Line 2");

        jLabel10.setText("Address Line 3");

        jLabel11.setText("Address Line 4");

        jLabel12.setText("Discord Username");

        jLabel13.setText("Discord Number");

        jLabel14.setText("Date of Birth");

        jLabel15.setText("Phone Number");

        usernameTxt.setEditable(false);
        usernameTxt.setFocusCycleRoot(true);
        usernameTxt.addFocusListener(new java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                usernameTxtFocusLost(evt);
            }
        });
        usernameTxt.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyTyped(java.awt.event.KeyEvent evt) {
                usernameTxtKeyTyped(evt);
            }
        });

        emailAddressTxt.setEditable(false);
        emailAddressTxt.setFocusCycleRoot(true);
        emailAddressTxt.addFocusListener(new
java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                emailAddressTxtFocusLost(evt);
```

```
            }
        });

        firstNameTxt.setEditable(false);
        firstNameTxt.setFocusCycleRoot(true);

        surnameTxt.setEditable(false);

        postcodeTxt.setEditable(false);

        addressLineOneTxt.setEditable(false);

        addressLineTwoTxt.setEditable(false);

        addressLineThreeTxt.setEditable(false);

        addressLineFourTxt.setEditable(false);

        discordUsernameTxt.setEditable(false);

        discordNumberTxt.setEditable(false);

        dateOfBirthTxt.setEditable(false);
        dateOfBirthTxt.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                dateOfBirthTxtActionPerformed(evt);
            }
        });

        phoneNumberTxt.setEditable(false);

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        editBtn.setText("Edit Details");
        editBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                editBtnActionPerformed(evt);
            }
        });

        passwordLabel.setForeground(new java.awt.Color(255, 51, 51));

        confirmPasswordLabel.setForeground(new java.awt.Color(255, 51,
51));
```

```
        firstNameLabel.setForeground(new java.awt.Color(255, 51, 51));

        surnameLabel.setForeground(new java.awt.Color(255, 51, 51));

        postcodeLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressOneLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressTwoLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressThreeLabel.setForeground(new java.awt.Color(255, 51,
51));

        addressFourLabel.setForeground(new java.awt.Color(255, 51, 51));

        discordUsernameLabel.setForeground(new java.awt.Color(255, 51,
51));

        discordNumberLabel.setForeground(new java.awt.Color(255, 51,
51));

        dateOfBirthLabel.setForeground(new java.awt.Color(255, 51, 51));

        phoneNumberLabel.setForeground(new java.awt.Color(255, 51, 51));

        jLabel20.setText("Elo");

        eloTxt.setEditable(false);

        removeBtn.setText("Remove Member");
        removeBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                removeBtnActionPerformed(evt);
            }
        });

        nextBtn.setText("Next Member");
        nextBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                nextBtnActionPerformed(evt);
            }
        });

        prevBtn.setText("Previous Member");
        prevBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                prevBtnActionPerformed(evt);
            }
```

```
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addContainerGap()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel1)
                                        .addGap(125, 125, 125)
                                        .addComponent(usernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel2)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(emailAddressTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(36, 36, 36)
                                .addComponent(backBtn)))
                        .addGap(0, 0, Short.MAX_VALUE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel5)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(firstNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                                .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                    .addComponent(jLabel6)
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addComponent(surnameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addComponent(jLabel7)
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addComponent(postcodeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addComponent(jLabel20)
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addComponent(eloTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(87, 87, 87)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                                .addComponent(passwordLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 70, Short.MAX_VALUE)
                                .addComponent(confirmPasswordLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                            .addComponent(prevBtn)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addGap(54, 54, 54)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(firstNameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(surnameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(postcodeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addGap(18, 18, 18)
                                    .addComponent(nextBtn)))))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED)
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addComponent(jLabel9)
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addComponent(addressLineTwoTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addComponent(jLabel8)
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addComponent(addressLineOneTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(jLabel10)
                                        .addComponent(jLabel11)
                                        .addComponent(jLabel12)
                                        .addComponent(jLabel14)
                                        .addComponent(jLabel13))
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                        .addComponent(discordNumberTxt)
                                        .addComponent(discordUsernameTxt)
                                        .addComponent(addressLineFourTxt)
```

```
                                        .addComponent(addressLineThreeTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 255, Short.MAX_VALUE)))
                                .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                    .addComponent(jLabel15)
                                    .addGap(29, 29, 29)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(dateOfBirthTxt,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                            .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(addressOneLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressTwoLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressThreeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressFourLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(discordUsernameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(discordNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(dateOfBirthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(phoneNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                            .addGap(0, 0, Short.MAX_VALUE)
                            .addComponent(removeBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 162,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(30, 30, 30)
                            .addComponent(editBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 162,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(35, 35, 35)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(60, 60, 60)
                                .addComponent(addressOneLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressTwoLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressThreeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressFourLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(discordUsernameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(26, 26, 26)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addGap(0, 0, Short.MAX_VALUE)
                                        .addComponent(confirmPasswordLab
el, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(19, 19, 19)
                                        .addComponent(firstNameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(surnameLabel,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addGroup(jPanel1Layout.createSequen
tialGroup()
                                            .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(jLabel1)
                                                .addComponent(usernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                            .addGap(18, 18, 18)
                                            .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(jLabel2)
                                                .addComponent(emailAddressTx
t, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                            .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                                .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                    .addGroup(jPanel1Layout.
createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                                        .addGroup(jPanel1Lay
out.createSequentialGroup()
                                                            .addGap(18, 18,
18)
                                                            .addComponent(pa
sswordLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                                        .addGroup(jPanel1Lay
out.createSequentialGroup()
                                                            .addGap(24, 24,
24)
                                                            .addGroup(jPanel
1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                                                .addComponen
t(firstNameTxt, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                                .addComponen
t(jLabel5))))
                                                    .addGap(27, 27, 27)
                                                    .addComponent(surnameTxt
, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                                .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                    .addPreferredGap(javax.s
wing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                                                        .addComponent(jLabel6)))
                                                .addGap(30, 30, 30)
                                                .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                                    .addComponent(postcodeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                    .addComponent(jLabel7))
                                                .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED, 14, Short.MAX_VALUE)))))
                            .addGap(18, 18, 18)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addComponent(discordNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addComponent(postcodeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGap(18, 18, 18)
                                    .addComponent(dateOfBirthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(18, 18, 18)
                                    .addComponent(phoneNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGroup(jPanel1Layout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel20)
                                    .addComponent(eloTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(26, 26, 26)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel8)
                                .addComponent(addressLineOneTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(18, 18, 18)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel9)
                                .addComponent(addressLineTwoTxt,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel10)
                            .addComponent(addressLineThreeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel11)
                            .addComponent(addressLineFourTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel12)
                            .addComponent(discordUsernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel13)
                            .addComponent(discordNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel14)
                            .addComponent(dateOfBirthTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel15)
                            .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
```

```
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(20, 20, 20)
                        .addComponent(backBtn)
                        .addContainerGap(39, Short.MAX_VALUE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(editBtn)
                            .addComponent(removeBtn)
                            .addComponent(nextBtn)
                            .addComponent(prevBtn))
                        .addGap(28, 28, 28))))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void
dateOfBirthTxtActionPerformed(java.awt.event.ActionEvent evt)
{

    }

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //goes back to the leader form
        WestCrossChessClubLeader leader = new
WestCrossChessClubLeader(session);
        leader.setVisible(true);this.dispose();
    }

    private void editBtnActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        //either saves changes made to a user's profile or enters
editing mode for a given user
        if(isUserInfoBeingEdited){
            //saves changes to user to server
            //checks to see that changes are valid
            //presence check
            if(Validation.containsIllegalStrings(usernameTxt.getText()+e
mailAddressTxt.getText()+firstNameTxt.getText()+surnameTxt.getText()+pos
tcodeTxt.getText()+addressLineOneTxt.getText()
                    +addressLineTwoTxt.getText()+addressLineThreeTxt.get
Text()+addressLineFourTxt.getText()+discordNumberTxt.getText()+discordUs
ernameTxt.getText()+dateOfBirthTxt.getText()+phoneNumberTxt.getText())){
//performs a look up check to see whether any field contains an illegal
character
                JOptionPane.showMessageDialog(this, "One or more fields
contain an illegal character sequence. Here is a list of illegal
caharacter sequences: "+Validation.getListOfIllegalStrings(),"Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(Validation.isPresent(firstNameTxt.getText())){//only
performs validation if first name is not empty
                if(!Validation.isAppropriateLength(35,
firstNameTxt.getText())){//length check - first name can't be more than
35 characters
                    JOptionPane.showMessageDialog(this, "The first name
can not be longer than 35 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.consistsOnlyOfLetters(firstNameTxt.getTex
t())){
                    JOptionPane.showMessageDialog(this, "The first name
can only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(surnameTxt.getText())){//only
performs validation if  surname is not empty
                if(!Validation.isAppropriateLength(40,
surnameTxt.getText())){//length check - surname can't be more than 40
characters
                    JOptionPane.showMessageDialog(this, "The surname can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.consistsOnlyOfLetters(surnameTxt.getText(
)){
                    JOptionPane.showMessageDialog(this, "The surname can
only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
```

```
                    return;
                }
            }
            if(Validation.isPresent(postcodeTxt.getText())){//only
performs validation is postcode is not left empty
                if(!Validation.isValidPostcode(postcodeTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The postcode
must be between 6-8 characters long, contain exactly one space and have
all non-space characters capitalised","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(!Validation.isAppropriateLength(40,
addressLineOneTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineTwoTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineThreeTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineFourTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(32,
discordUsernameTxt.getText())){
                JOptionPane.showMessageDialog(this, "The Discord
username can not be longer than 32 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(Validation.isPresent(discordNumberTxt.getText())){
                if(!Validation.isInteger(discordNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The Discord
number must be an integer","Invalid Input",JOptionPane.OK_OPTION);
                    return;
```

```
                }
                if(!Validation.isAppropriateLength(4,
discordNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The Discord
number must be no more than 4 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.rangeCheck(0, 9999,
Integer.parseInt(discordNumberTxt.getText())){//range check
                    JOptionPane.showMessageDialog(this, "The Discord
number must be between 0000-9999","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(dateOfBirthTxt.getText())){
                if(!Validation.formatCheckDateOfBirth(dateOfBirthTxt.get
Text())){
                    JOptionPane.showMessageDialog(this, "The date of
birth must be a valid date of birth in the form dd/mm/yyyy","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(phoneNumberTxt.getText())){
                if(!Validation.isAppropriateLength(20,
phoneNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "Phone number
can not be longer than 20 characters.","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.onlyContainsPhoneNumberCharacters(phoneNu
mberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "Phone number
can only contain plusses, spaces and integers.","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            //saves changes having validated them
//Username 0, password 1, first name 2, surname 3, postcode 4,
addressline one 5, address line two 6, address line three 7, address
line four 8, discord username 9, discord number 10, date of birth 11,
phone number 12, isClubLeader 13, isClubMember 14, ELO 15,
hasUserRequestedToJoinClub 16
            String[]newRecord=new
String[]{usernameTxt.getText(),clubMembersInfo[clubMembersIndex][1],firs
tNameTxt.getText(),surnameTxt.getText(),postcodeTxt.getText(),addressLin
eOneTxt.getText(),addressLineTwoTxt.getText(),addressLineThreeTxt.getTex
t(),addressLineFourTxt.getText(),discordNumberTxt.getText(),discordNumbe
rTxt.getText(),dateOfBirthTxt.getText(),phoneNumberTxt.getText(),clubMem
```

```
bersInfo[clubMembersIndex][13],clubMembersInfo[clubMembersIndex][14],clu
bMembersInfo[clubMembersIndex][15],clubMembersInfo[clubMembersIndex][16]
};
            for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                clubMembersInfo[clubMembersIndex][x]=newRecord[x];
            }
            try{
                ThreadClient.updateUserInfo(newRecord);//changes are
saved
            }catch(Exception e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;
            }
            isUserInfoBeingEdited=false;
            //updates components and exits editing mode
            prevBtn.setEnabled(true);nextBtn.setEnabled(true);
            removeBtn.setEnabled(true);
            firstNameTxt.setEditable(false);surnameTxt.setEditable(false
);
            postcodeTxt.setEditable(false);
            addressLineOneTxt.setEditable(false);addressLineTwoTxt.setEd
itable(false);addressLineThreeTxt.setEditable(false);
            addressLineFourTxt.setEditable(false);discordUsernameTxt.set
Editable(false);discordNumberTxt.setEditable(false);
            dateOfBirthTxt.setEditable(false);phoneNumberTxt.setEditable
(false);
            loadFormBasedOnIndex();//updates form values
            editBtn.setText("Edit Details");
            removeBtn.setText("Remove Member");
            removeBtn.setBackground(Color.red);
        }else{
            //puts form in editing mode - chaning looks and feel of
components
            editBtn.setText("Save Changes");
            removeBtn.setText("Cancel Changes");
            isUserInfoBeingEdited=true;
            prevBtn.setEnabled(false);nextBtn.setEnabled(false);
            removeBtn.setEnabled(true);removeBtn.setBackground(nextBtn.g
etBackground());
            //sets fields to be editable
            firstNameTxt.setEditable(true);surnameTxt.setEditable(true);
            postcodeTxt.setEditable(true);
            addressLineOneTxt.setEditable(true);addressLineTwoTxt.setEdi
table(true);addressLineThreeTxt.setEditable(true);
            addressLineFourTxt.setEditable(true);discordUsernameTxt.setE
ditable(true);discordNumberTxt.setEditable(true);
            dateOfBirthTxt.setEditable(true);phoneNumberTxt.setEditable(
true);
        }
    }
```

```
    private void usernameTxtKeyTyped(java.awt.event.KeyEvent evt)
{

    }

    private void usernameTxtFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void emailAddressTxtFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void removeBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //removes members or cancels changes made when editing
        if(isUserInfoBeingEdited){
            //cancles changes made whilst editing and reverst form to
the state it was in before editing mode was entered
            isUserInfoBeingEdited=false;
            prevBtn.setEnabled(true);nextBtn.setEnabled(true);
            removeBtn.setEnabled(true);
            firstNameTxt.setEditable(false);surnameTxt.setEditable(false
);
            postcodeTxt.setEditable(false);
            addressLineOneTxt.setEditable(false);addressLineTwoTxt.setEd
itable(false);addressLineThreeTxt.setEditable(false);
            addressLineFourTxt.setEditable(false);discordUsernameTxt.set
Editable(false);discordNumberTxt.setEditable(false);
            dateOfBirthTxt.setEditable(false);phoneNumberTxt.setEditable
(false);
            loadFormBasedOnIndex();
            editBtn.setText("Edit Details");
            removeBtn.setText("Remove Member");
            removeBtn.setBackground(Color.red);
            return;
        }else{
            //removes a member from the chess club after asking for
confirmation
            int confirm = JOptionPane.showConfirmDialog(this, "Are you
sure you would like to remove the member from the Chess Club?",
"Remove?", JOptionPane.YES_NO_CANCEL_OPTION);
            if(confirm==JOptionPane.YES_OPTION){
                try{
                    ThreadClient.removeUserFromChessClub(clubMembersInfo
[clubMembersIndex][0]);//removes user
                }catch(Exception e){
                    JOptionPane.showMessageDialog(this,"Error when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
                    Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;
```

```
                }
                //removes user info from array
                String[][]newClubMemberInfo=new
String[clubMembersInfo.length-1][UserInfo.WIDTH_OF_USER_DETAILS_FILE];
                for(int i=0;i<clubMembersIndex;i++){
                    for(int
x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                        newClubMemberInfo[i][x]=clubMembersInfo[i][x];
                    }
                }
                for(int
i=clubMembersIndex+1;i<clubMembersInfo.length;i++){
                    for(int
x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                        newClubMemberInfo[i-1][x]=clubMembersInfo[i][x];
                    }
                }
                clubMembersInfo=newClubMemberInfo;
                clubMembersIndex=0;
                loadFormBasedOnIndex();//loads updated form without
removed member
            }
        }
    }

    private void prevBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the leader to the previous user
        clubMembersIndex--;
        loadFormBasedOnIndex();
    }

    private void nextBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the leader to the next user
        clubMembersIndex++;
        loadFormBasedOnIndex();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
```

```
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(MemberDetails.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(MemberDetails.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(MemberDetails.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(MemberDetails.class.getNa
me()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MemberDetails().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel addressFourLabel;
    private javax.swing.JTextField addressLineFourTxt;
    private javax.swing.JTextField addressLineOneTxt;
    private javax.swing.JTextField addressLineThreeTxt;
    private javax.swing.JTextField addressLineTwoTxt;
    private javax.swing.JLabel addressOneLabel;
    private javax.swing.JLabel addressThreeLabel;
    private javax.swing.JLabel addressTwoLabel;
    private javax.swing.JButton backBtn;
    private javax.swing.JLabel confirmPasswordLabel;
    private javax.swing.JLabel dateOfBirthLabel;
    private javax.swing.JTextField dateOfBirthTxt;
    private javax.swing.JLabel discordNumberLabel;
    private javax.swing.JTextField discordNumberTxt;
    private javax.swing.JLabel discordUsernameLabel;
```

```
    private javax.swing.JTextField discordUsernameTxt;
    private javax.swing.JButton editBtn;
    private javax.swing.JTextField eloTxt;
    private javax.swing.JTextField emailAddressTxt;
    private javax.swing.JLabel firstNameLabel;
    private javax.swing.JTextField firstNameTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton nextBtn;
    private javax.swing.JLabel passwordLabel;
    private javax.swing.JLabel phoneNumberLabel;
    private javax.swing.JTextField phoneNumberTxt;
    private javax.swing.JLabel postcodeLabel;
    private javax.swing.JTextField postcodeTxt;
    private javax.swing.JButton prevBtn;
    private javax.swing.JButton removeBtn;
    private javax.swing.JLabel surnameLabel;
    private javax.swing.JTextField surnameTxt;
    private javax.swing.JTextField usernameTxt;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import javax.swing.JOptionPane;

/**
 * Creates a menu form that allows a user to access various parts of the
system
 * @author mortimer
```

```
 */
public class Menu extends javax.swing.JFrame {
    SessionInfo session;//the session information of the current user
    /**
     * Creates new form Menu
     */
    private Menu() {
        initComponents();
    }
    /**
     * Creates new form Menu
     * @param sess The session information of the current user
     */
    public Menu(SessionInfo sess){//overloading is used to store session
info and run the form
        session =sess;
        initComponents();
        //sets the username label depending on the session information
        if(sess.getIsGuest()){
            logInInfoLbl.setText("You are currently logged in as a
Guest.");
            onlineBtn.setEnabled(false);
        }else{
            logInInfoLbl.setText("Username: "+sess.getUsername());
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        chessClubBtn = new javax.swing.JButton();
        onlineBtn = new javax.swing.JButton();
        offlineBtn = new javax.swing.JButton();
        logOutBtn = new javax.swing.JButton();
        Exit = new javax.swing.JButton();
        logInInfoLbl = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Menu");

        jLabel1.setText("West Cross Chess Club");
        jLabel1.setToolTipText("");
```

```
        chessClubBtn.setText("West Cross Chess Club");
        chessClubBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                chessClubBtnActionPerformed(evt);
            }
        });

        onlineBtn.setText("Online");
        onlineBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                onlineBtnActionPerformed(evt);
            }
        });

        offlineBtn.setText("Offline");
        offlineBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                offlineBtnActionPerformed(evt);
            }
        });

        logOutBtn.setText("Log Out");
        logOutBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                logOutBtnActionPerformed(evt);
            }
        });

        Exit.setText("Exit");
        Exit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                ExitActionPerformed(evt);
            }
        });

        logInInfoLbl.setText("Username:");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
```

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addGap(293, 293, 293))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(43, 43, 43)
                .addComponent(logOutBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(Exit)
                .addGap(88, 88, 88))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(logInInfoLbl,
javax.swing.GroupLayout.PREFERRED_SIZE, 219,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(286, 286, 286)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(chessClubBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(onlineBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(offlineBtn,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
                .addContainerGap(271, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(29, 29, 29)
                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 32,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(6, 6, 6)
                .addComponent(logInInfoLbl)
                .addGap(26, 26, 26)
                .addComponent(chessClubBtn)
                .addGap(51, 51, 51)
                .addComponent(onlineBtn)
                .addGap(56, 56, 56)
                .addComponent(offlineBtn)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 109, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(logOutBtn)
                    .addComponent(Exit))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void ExitActionPerformed(java.awt.event.ActionEvent evt)
{
        //exits sytem upon user confirmation
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to exit?", "Exit?", JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            System.exit(0);
        }
    }

    private void logOutBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // logs user out after confirmation
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to log out?", "Log out?",
JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            LogIn log = new LogIn();
            log.setVisible(true);
            this.dispose();
        }
    }

    private void offlineBtnActionPerformed(java.awt.event.ActionEvent
evt) {
```

```
        // takes user to form for offline games
        OfflinePlay offline = new OfflinePlay(session);
        offline.setVisible(true);
        this.dispose();
    }

    private void onlineBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //takes user to online game form
        OnlineMenu online = new OnlineMenu(session);
        online.setVisible(true);
        this.dispose();
    }

    private void chessClubBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // takes the user to the west cross chess club menu. This form
differs depending on what type of user is attempting to access the form
        if(session.getIsGuest()){//takes the user to the chess club form
for guests
            WestCrossChessClubGuest chessClubGuest = new
WestCrossChessClubGuest(session);
            chessClubGuest.setVisible(true);
            this.dispose();
        }else if(session.getIsClubMember()==false){//takes the user to
the chess club form for non-member users
            WestCrossChessClubUser chessClubUser = new
WestCrossChessClubUser(session);
            chessClubUser.setVisible(true);
            this.dispose();
        }else if(session.getIsClubMember() &&
session.getIsClubLeader()==false){//takes the user to the chess club
form for member users
            WestCrossChessClubMember chessClubMember = new
WestCrossChessClubMember(session);
            chessClubMember.setVisible(true);
            this.dispose();
        }else{//takes the user to the chess club form for the chess club
leader
            WestCrossChessClubLeader chessClubLeader = new
WestCrossChessClubLeader(session);
            chessClubLeader.setVisible(true);
            this.dispose();
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
```

```
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(Menu.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(Menu.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(Menu.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(Menu.class.getName()).log
(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Menu().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton Exit;
    private javax.swing.JButton chessClubBtn;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JLabel logInInfoLbl;
    private javax.swing.JButton logOutBtn;
    private javax.swing.JButton offlineBtn;
    private javax.swing.JButton onlineBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.Color;
import javax.swing.JOptionPane;
import philipm.cs5.software.development.server.system.UserInfo;


/**
 * Creates a form that allows the leader to view the details of non
chess club members. The leader can remove these users from the system as
well as add them to the chess club.
 * @author mortimer
 */
public class NonMemberDetails extends javax.swing.JFrame {
    private SessionInfo session;//stores the session information
    private String[][]nonMemberInfo;//stores the data of non-members
    private String[]emails;//stores the email addresses of the users
    private int nonMemberIndex=0;//stores the index which represents
which user is currently being viewed
    private boolean isUserInfoBeingEdited=false;//stores whether the
user data is currently being edited
    /**
     * Creates new form NonMemberDetails
     * @param session The session information
     * @throws java.lang.Exception An exception may occur when
attempting to load info.
     */
    public NonMemberDetails(SessionInfo session) throws Exception{
        initComponents();
        this.session=session;//stores the session information
        //sets look and feel of form
        noUsersLbl.setVisible(false);
        nonMemberInfo=ThreadClient.getAllUsersWhoAreNotMembers(); //gets
the data from the server
        emails = new String[nonMemberInfo.length];//stores the email
address of all users
        removeBtn.setBackground(Color.red);
        prevBtn.setEnabled(false);nextBtn.setEnabled(nonMemberIndex!=non
MemberInfo.length-1);
        //gets email address of users
        for(int req=0;req<emails.length;req++){
            emails[req]=ThreadClient.getEmailAddressOfUser(nonMemberInfo
[req][0]);
        }
        loadFormBasedOnIndex();//loads first user to form if appropriate
    }
    /**
```

```
     * Creates new form NonMemberDetails
     */
    private NonMemberDetails(){
        initComponents();
    }
    /**
     * Loads user info onto the form based on the index
     */
    private void loadFormBasedOnIndex(){
        clearForm();//clears the form
        if(nonMemberInfo.length==0){//if there are no users who are not
members, the form is updated to reflect this
            noUsersLbl.setVisible(true);
            addUserBtn.setEnabled(false);
            prevBtn.setEnabled(false);nextBtn.setEnabled(false);
            removeBtn.setEnabled(false);editBtn.setEnabled(false);
            return;
        }
        //enables or disables the buttons that allow navigation between
users based on the user currently being viewed.
        prevBtn.setEnabled(true);nextBtn.setEnabled(true);
        addUserBtn.setEnabled(true);
        if(nonMemberIndex==0){
            prevBtn.setEnabled(false);
        }
        if(nonMemberIndex==nonMemberInfo.length-1){
            nextBtn.setEnabled(false);
        }
        //ensuers that the leader can't remove themselves
        removeBtn.setEnabled(true);
        if(nonMemberInfo[nonMemberIndex][0].equals(session.getUsername()
)){
            removeBtn.setEnabled(false);
        }
        //Username, password, first name, surname, postcode, addressline
one, address line two, address line three, address line four, discord
username, discord number, date of birth, phone number, isClubLeader,
isClubMember, ELO, hasUserRequestedToJoinClub
        //sets the form fields with the appropriate values
        usernameTxt.setText(nonMemberInfo[nonMemberIndex][0]);
        emailAddressTxt.setText(emails[nonMemberIndex]);
        firstNameTxt.setText(nonMemberInfo[nonMemberIndex][2]);surnameTx
t.setText(nonMemberInfo[nonMemberIndex][3]);
        postcodeTxt.setText(nonMemberInfo[nonMemberIndex][4]);addressLin
eOneTxt.setText(nonMemberInfo[nonMemberIndex][5]);
        addressLineTwoTxt.setText(nonMemberInfo[nonMemberIndex][6]);addr
essLineThreeTxt.setText(nonMemberInfo[nonMemberIndex][7]);
        addressLineFourTxt.setText(nonMemberInfo[nonMemberIndex][8]);dis
cordUsernameTxt.setText(nonMemberInfo[nonMemberIndex][9]);
        discordNumberTxt.setText(nonMemberInfo[nonMemberIndex][10]);date
OfBirthTxt.setText(nonMemberInfo[nonMemberIndex][11]);
        phoneNumberTxt.setText(nonMemberInfo[nonMemberIndex][12]);eloTxt
.setText(nonMemberInfo[nonMemberIndex][15]);
```

```
    }
    /**
     * Sets all fields on the form empty
     */
    private void clearForm(){
        usernameTxt.setText("");emailAddressTxt.setText("");
        firstNameTxt.setText("");surnameTxt.setText("");postcodeTxt.setT
ext("");
        eloTxt.setText("");addressLineOneTxt.setText("");addressLineTwoT
xt.setText("");
        addressLineThreeTxt.setText("");addressLineFourTxt.setText("");
        discordUsernameTxt.setText("");discordNumberTxt.setText("");
        dateOfBirthTxt.setText("");phoneNumberTxt.setText("");
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        usernameTxt = new javax.swing.JTextField();
        emailAddressTxt = new javax.swing.JTextField();
        firstNameTxt = new javax.swing.JTextField();
        surnameTxt = new javax.swing.JTextField();
        postcodeTxt = new javax.swing.JTextField();
        addressLineOneTxt = new javax.swing.JTextField();
        addressLineTwoTxt = new javax.swing.JTextField();
        addressLineThreeTxt = new javax.swing.JTextField();
        addressLineFourTxt = new javax.swing.JTextField();
        discordUsernameTxt = new javax.swing.JTextField();
        discordNumberTxt = new javax.swing.JTextField();
        dateOfBirthTxt = new javax.swing.JTextField();
        phoneNumberTxt = new javax.swing.JTextField();
```

```
        backBtn = new javax.swing.JButton();
        editBtn = new javax.swing.JButton();
        passwordLabel = new javax.swing.JLabel();
        confirmPasswordLabel = new javax.swing.JLabel();
        firstNameLabel = new javax.swing.JLabel();
        surnameLabel = new javax.swing.JLabel();
        postcodeLabel = new javax.swing.JLabel();
        addressOneLabel = new javax.swing.JLabel();
        addressTwoLabel = new javax.swing.JLabel();
        addressThreeLabel = new javax.swing.JLabel();
        addressFourLabel = new javax.swing.JLabel();
        discordUsernameLabel = new javax.swing.JLabel();
        discordNumberLabel = new javax.swing.JLabel();
        dateOfBirthLabel = new javax.swing.JLabel();
        phoneNumberLabel = new javax.swing.JLabel();
        jLabel20 = new javax.swing.JLabel();
        eloTxt = new javax.swing.JTextField();
        removeBtn = new javax.swing.JButton();
        nextBtn = new javax.swing.JButton();
        prevBtn = new javax.swing.JButton();
        addUserBtn = new javax.swing.JButton();
        noUsersLbl = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Non Member Details");

        jLabel1.setText("Username");

        jLabel2.setText("Email Address");

        jLabel5.setText("First Name");

        jLabel6.setText("Surname");

        jLabel7.setText("Postcode");

        jLabel8.setText("Address Line 1");

        jLabel9.setText("Address Line 2");

        jLabel10.setText("Address Line 3");

        jLabel11.setText("Address Line 4");

        jLabel12.setText("Discord Username");

        jLabel13.setText("Discord Number");

        jLabel14.setText("Date of Birth");

        jLabel15.setText("Phone Number");
```

```
        usernameTxt.setEditable(false);
        usernameTxt.setFocusCycleRoot(true);
        usernameTxt.addFocusListener(new java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                usernameTxtFocusLost(evt);
            }
        });
        usernameTxt.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyTyped(java.awt.event.KeyEvent evt) {
                usernameTxtKeyTyped(evt);
            }
        });

        emailAddressTxt.setEditable(false);
        emailAddressTxt.setFocusCycleRoot(true);
        emailAddressTxt.addFocusListener(new
java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                emailAddressTxtFocusLost(evt);
            }
        });

        firstNameTxt.setEditable(false);
        firstNameTxt.setFocusCycleRoot(true);

        surnameTxt.setEditable(false);

        postcodeTxt.setEditable(false);

        addressLineOneTxt.setEditable(false);

        addressLineTwoTxt.setEditable(false);

        addressLineThreeTxt.setEditable(false);

        addressLineFourTxt.setEditable(false);

        discordUsernameTxt.setEditable(false);

        discordNumberTxt.setEditable(false);

        dateOfBirthTxt.setEditable(false);
        dateOfBirthTxt.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                dateOfBirthTxtActionPerformed(evt);
            }
        });

        phoneNumberTxt.setEditable(false);

        backBtn.setText("Back");
```

```
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        editBtn.setText("Edit Details");
        editBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                editBtnActionPerformed(evt);
            }
        });

        passwordLabel.setForeground(new java.awt.Color(255, 51, 51));

        confirmPasswordLabel.setForeground(new java.awt.Color(255, 51,
51));

        firstNameLabel.setForeground(new java.awt.Color(255, 51, 51));

        surnameLabel.setForeground(new java.awt.Color(255, 51, 51));

        postcodeLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressOneLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressTwoLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressThreeLabel.setForeground(new java.awt.Color(255, 51,
51));

        addressFourLabel.setForeground(new java.awt.Color(255, 51, 51));

        discordUsernameLabel.setForeground(new java.awt.Color(255, 51,
51));

        discordNumberLabel.setForeground(new java.awt.Color(255, 51,
51));

        dateOfBirthLabel.setForeground(new java.awt.Color(255, 51, 51));

        phoneNumberLabel.setForeground(new java.awt.Color(255, 51, 51));

        jLabel20.setText("Elo");

        eloTxt.setEditable(false);

        removeBtn.setText("Delete User");
        removeBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
```

```
{
                removeBtnActionPerformed(evt);
            }
        });

        nextBtn.setText("Next Member");
        nextBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                nextBtnActionPerformed(evt);
            }
        });

        prevBtn.setText("Previous Member");
        prevBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                prevBtnActionPerformed(evt);
            }
        });

        addUserBtn.setText("Add User To Club");
        addUserBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                addUserBtnActionPerformed(evt);
            }
        });

        noUsersLbl.setText("There are no users who are not members.");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addContainerGap()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel1)
                                        .addGap(125, 125, 125)
```

```
                                        .addComponent(usernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel2)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(emailAddressTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                                .addGap(0, 2, Short.MAX_VALUE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addContainerGap()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel5)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(firstNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(javax.swing.GroupLayout.Al
ignment.TRAILING, jPanel1Layout.createSequentialGroup()
                                        .addComponent(jLabel6)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(surnameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel7)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(postcodeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel20)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(eloTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE))))
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addGap(36, 36, 36)
                                    .addComponent(backBtn)
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                    .addComponent(addUserBtn)
                                    .addGap(101, 101, 101)))
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addGap(87, 87, 87)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                        .addComponent(passwordLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 70, Short.MAX_VALUE)
                                        .addComponent(confirmPasswordLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED)
                                    .addComponent(prevBtn)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                        .addGroup(jPanel1Layout.createSequen
tialGroup()
                                            .addGap(54, 54, 54)
                                            .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                                .addComponent(firstNameLabel
, javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addComponent(surnameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                .addComponent(postcodeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                                        .addGroup(jPanel1Layout.createSequen
tialGroup()
                                            .addGap(18, 18, 18)
                                            .addComponent(nextBtn)))))
                            .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED))
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
```

```
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(noUsersLbl)
                        .addGap(41, 41, 41)))
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel9)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(addressLineTwoTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel8)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(addressLineOneTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel10)
                                    .addComponent(jLabel11)
                                    .addComponent(jLabel12)
                                    .addComponent(jLabel14)
                                    .addComponent(jLabel13))
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                    .addComponent(discordNumberTxt)
                                    .addComponent(discordUsernameTxt)
                                    .addComponent(addressLineFourTxt)
                                    .addComponent(addressLineThreeTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 255, Short.MAX_VALUE)))
                            .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                .addComponent(jLabel15)
                                .addGap(29, 29, 29)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(dateOfBirthTxt,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(addressOneLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(addressTwoLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(addressThreeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(addressFourLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(discordUsernameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(discordNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(dateOfBirthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(phoneNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(removeBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 162,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(30, 30, 30)
                        .addComponent(editBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 162,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(35, 35, 35)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
```

```
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(60, 60, 60)
                                .addComponent(addressOneLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressTwoLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressThreeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressFourLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(discordUsernameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(26, 26, 26)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addGap(0, 0, Short.MAX_VALUE)
                                        .addComponent(confirmPasswordLab
el, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(19, 19, 19)
                                        .addComponent(firstNameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(surnameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                            .addComponent(jLabel1)
                                            .addComponent(usernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addGap(18, 18, 18)
                                        .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                            .addComponent(jLabel2)
                                            .addComponent(emailAddressTx
t, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                        .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                            .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                .addGroup(jPanel1Layout.
createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                                    .addGroup(jPanel1Lay
out.createSequentialGroup()
                                                        .addGap(18, 18,
18)
                                                        .addComponent(pa
sswordLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                                    .addGroup(jPanel1Lay
out.createSequentialGroup()
                                                        .addGap(24, 24,
24)
                                                        .addGroup(jPanel
1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                                            .addComponen
t(firstNameTxt, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                                            .addComponen
t(jLabel5))))
                                                .addGap(27, 27, 27)
                                                .addComponent(surnameTxt
, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                            .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                .addPreferredGap(javax.s
wing.LayoutStyle.ComponentPlacement.RELATED)
                                                .addComponent(jLabel6)))
                                        .addGap(30, 30, 30)
                                        .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                            .addComponent(postcodeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                            .addComponent(jLabel7))
                                        .addPreferredGap(javax.swing.Lay
```

```
outStyle.ComponentPlacement.RELATED, 14, Short.MAX_VALUE)))))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(discordNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(postcodeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGroup(jPanel1Layout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel20)
                                .addComponent(eloTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(18, 18, 18)
                                .addComponent(dateOfBirthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(phoneNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(44, 44, 44)
                                .addComponent(noUsersLbl))))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(26, 26, 26)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel8)
                            .addComponent(addressLineOneTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel9)
                            .addComponent(addressLineTwoTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
```

```
                                .addComponent(jLabel10)
                                .addComponent(addressLineThreeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(18, 18, 18)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel11)
                                .addComponent(addressLineFourTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(18, 18, 18)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel12)
                                .addComponent(discordUsernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(18, 18, 18)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel13)
                                .addComponent(discordNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(18, 18, 18)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel14)
                                .addComponent(dateOfBirthTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(18, 18, 18)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel15)
                                .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(20, 20, 20)
                            .addComponent(backBtn)
                            .addContainerGap(39, Short.MAX_VALUE))
                        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
```

```
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(editBtn)
                            .addComponent(removeBtn)
                            .addComponent(nextBtn)
                            .addComponent(prevBtn)
                            .addComponent(addUserBtn))
                        .addGap(28, 28, 28))))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void
dateOfBirthTxtActionPerformed(java.awt.event.ActionEvent evt)
{

    }

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //goes back to the leader form
        WestCrossChessClubLeader leader = new
WestCrossChessClubLeader(session);
        leader.setVisible(true);this.dispose();
    }

    private void editBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        if(isUserInfoBeingEdited){
            //saves changes to user to server
            //checks to see that changes are valid
            //presence check
            if(Validation.containsIllegalStrings(usernameTxt.getText()+e
```

```
mailAddressTxt.getText()+firstNameTxt.getText()+surnameTxt.getText()+pos
tcodeTxt.getText()+addressLineOneTxt.getText()
                    +addressLineTwoTxt.getText()+addressLineThreeTxt.get
Text()+addressLineFourTxt.getText()+discordNumberTxt.getText()+discordUs
ernameTxt.getText()+dateOfBirthTxt.getText()+phoneNumberTxt.getText())){
//performs a look up check to see whether any field contains an illegal
character
                JOptionPane.showMessageDialog(this, "One or more fields
contain an illegal character sequence. Here is a list of illegal
caharacter sequences: "+Validation.getListOfIllegalStrings(),"Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(Validation.isPresent(firstNameTxt.getText())){//only
performs validation if first name is not empty
                if(!Validation.isAppropriateLength(35,
firstNameTxt.getText())){//length check - first name can't be more than
35 characters
                    JOptionPane.showMessageDialog(this, "The first name
can not be longer than 35 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.consistsOnlyOfLetters(firstNameTxt.getTex
t())){
                    JOptionPane.showMessageDialog(this, "The first name
can only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(surnameTxt.getText())){//only
performs validation if  surname is not empty
                if(!Validation.isAppropriateLength(40,
surnameTxt.getText())){//length check - surname can't be more than 40
characters
                    JOptionPane.showMessageDialog(this, "The surname can
not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.consistsOnlyOfLetters(surnameTxt.getText(
))){
                    JOptionPane.showMessageDialog(this, "The surname can
only contain characters that are in the english alphabet","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(postcodeTxt.getText())){//only
performs validation is postcode is not left empty
                if(!Validation.isValidPostcode(postcodeTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The postcode
```

```
must be between 6-8 characters long, contain exactly one space and have
all non-space characters capitalised","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(!Validation.isAppropriateLength(40,
addressLineOneTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineTwoTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineThreeTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(40,
addressLineFourTxt.getText())){//length check
                JOptionPane.showMessageDialog(this, "Each address line
can not be longer than 40 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(!Validation.isAppropriateLength(32,
discordUsernameTxt.getText())){
                JOptionPane.showMessageDialog(this, "The Discord
username can not be longer than 32 characters","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            if(Validation.isPresent(discordNumberTxt.getText())){
                if(!Validation.isInteger(discordNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The Discord
number must be an integer","Invalid Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.isAppropriateLength(4,
discordNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "The Discord
number must be no more than 4 characters","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
```

```
                }
                if(!Validation.rangeCheck(0, 9999,
Integer.parseInt(discordNumberTxt.getText())){//range check
                    JOptionPane.showMessageDialog(this, "The Discord
number must be between 0000-9999","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(dateOfBirthTxt.getText())){
                if(!Validation.formatCheckDateOfBirth(dateOfBirthTxt.get
Text())){
                    JOptionPane.showMessageDialog(this, "The date of
birth must be a valid date of birth in the form dd/mm/yyyy","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            if(Validation.isPresent(phoneNumberTxt.getText())){
                if(!Validation.isAppropriateLength(20,
phoneNumberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "Phone number
can not be longer than 20 characters.","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
                if(!Validation.onlyContainsPhoneNumberCharacters(phoneNu
mberTxt.getText())){
                    JOptionPane.showMessageDialog(this, "Phone number
can only contain plusses, spaces and integers.","Invalid
Input",JOptionPane.OK_OPTION);
                    return;
                }
            }
            //saves changes having validated them
//Username 0, password 1, first name 2, surname 3, postcode 4,
addressline one 5, address line two 6, address line three 7, address
line four 8, discord username 9, discord number 10, date of birth 11,
phone number 12, isClubLeader 13, isClubMember 14, ELO 15,
hasUserRequestedToJoinClub 16
            String[]newRecord=new
String[]{usernameTxt.getText(),nonMemberInfo[nonMemberIndex][1],firstNam
eTxt.getText(),surnameTxt.getText(),postcodeTxt.getText(),addressLineOne
Txt.getText(),addressLineTwoTxt.getText(),addressLineThreeTxt.getText(),
addressLineFourTxt.getText(),discordNumberTxt.getText(),discordNumberTxt
.getText(),dateOfBirthTxt.getText(),phoneNumberTxt.getText(),nonMemberIn
fo[nonMemberIndex][13],nonMemberInfo[nonMemberIndex][14],nonMemberInfo[n
onMemberIndex][15],nonMemberInfo[nonMemberIndex][16]};
            for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                nonMemberInfo[nonMemberIndex][x]=newRecord[x];
            }
            try{
                ThreadClient.updateUserInfo(newRecord);//saves changes
```

```
made
            }catch(Exception e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;
            }
            //updates look and feel of form and exits editing mode
            isUserInfoBeingEdited=false;
            //updates components
            prevBtn.setEnabled(true);nextBtn.setEnabled(true);
            removeBtn.setEnabled(true);addUserBtn.setEnabled(true);
            firstNameTxt.setEditable(false);surnameTxt.setEditable(false
);
            postcodeTxt.setEditable(false);
            addressLineOneTxt.setEditable(false);addressLineTwoTxt.setEd
itable(false);addressLineThreeTxt.setEditable(false);
            addressLineFourTxt.setEditable(false);discordUsernameTxt.set
Editable(false);discordNumberTxt.setEditable(false);
            dateOfBirthTxt.setEditable(false);phoneNumberTxt.setEditable
(false);
            loadFormBasedOnIndex();//updates form
            editBtn.setText("Edit Details");
            removeBtn.setText("Delete User");
            removeBtn.setBackground(Color.red);
        }else{
            //puts form in editing mode - updating various components
and variables accordingly.
            editBtn.setText("Save Changes");
            removeBtn.setText("Cancel Changes");
            isUserInfoBeingEdited=true;
            prevBtn.setEnabled(false);nextBtn.setEnabled(false);
            addUserBtn.setEnabled(false);
            removeBtn.setEnabled(true);removeBtn.setBackground(nextBtn.g
etBackground());
            //sets fields to be editable
            firstNameTxt.setEditable(true);surnameTxt.setEditable(true);
            postcodeTxt.setEditable(true);
            addressLineOneTxt.setEditable(true);addressLineTwoTxt.setEdi
table(true);addressLineThreeTxt.setEditable(true);
            addressLineFourTxt.setEditable(true);discordUsernameTxt.setE
ditable(true);discordNumberTxt.setEditable(true);
            dateOfBirthTxt.setEditable(true);phoneNumberTxt.setEditable(
true);
        }
    }

    private void usernameTxtKeyTyped(java.awt.event.KeyEvent evt)
{

    }

    private void usernameTxtFocusLost(java.awt.event.FocusEvent evt)
```

```
{

    }

    private void emailAddressTxtFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void removeBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //removes members or cancels changes made when editing
        if(isUserInfoBeingEdited){
            //exits editing mode and cancels and changes made whilst
editing the user
            //updates look and feel of form to how it was before editing
started
            isUserInfoBeingEdited=false;
            prevBtn.setEnabled(true);nextBtn.setEnabled(true);
            removeBtn.setEnabled(true);
            firstNameTxt.setEditable(false);surnameTxt.setEditable(false
);
            postcodeTxt.setEditable(false);
            addressLineOneTxt.setEditable(false);addressLineTwoTxt.setEd
itable(false);addressLineThreeTxt.setEditable(false);
            addressLineFourTxt.setEditable(false);discordUsernameTxt.set
Editable(false);discordNumberTxt.setEditable(false);
            dateOfBirthTxt.setEditable(false);phoneNumberTxt.setEditable
(false);
            loadFormBasedOnIndex();
            editBtn.setText("Edit Details");
            removeBtn.setText("Delete User");
            removeBtn.setBackground(Color.red);
            addUserBtn.setEnabled(true);
            return;
        }else{
            //removes the user from the entire system
            //asks for confirmation to delete user
            int confirm = JOptionPane.showConfirmDialog(this, "Are you
sure you would like to delete this account", "Delete?",
JOptionPane.YES_NO_CANCEL_OPTION);
            if(confirm==JOptionPane.YES_OPTION){
                try{
                    ThreadClient.deleteUserInfoOfNonMemberUserFromServer
(nonMemberInfo[nonMemberIndex][0]);//deletes the account
                }catch(Exception e){
                    JOptionPane.showMessageDialog(this,"Error when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
                    Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;
                }
                removeCurrentIndexUserFromArrayAndSetIndexToZero();//upd
ates the array without the user
```

```
                loadFormBasedOnIndex();//loads the updated form
            }
        }
    }

    private void prevBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes the leader to the previous user
        nonMemberIndex--;
        loadFormBasedOnIndex();
    }

    private void nextBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes the leader to the next user
        nonMemberIndex++;
        loadFormBasedOnIndex();
    }

    private void addUserBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // adds the user to the club
        //asks for confirmation before adding the user to the club
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to add this user to the club?", "Add?",
JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            //adds user to the chess club
            try{
                ThreadClient.addUserToClubAsMember(usernameTxt.getText()
);//adds the user to the club
            }catch(Exception e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;

            }
            removeCurrentIndexUserFromArrayAndSetIndexToZero();//removes
the user from the array
            loadFormBasedOnIndex();//updates the form without the user
        }
    }
    /**
     * Removes the user info of the user at the current index from the
array and form. The index is then set to zero.
     */
    private void removeCurrentIndexUserFromArrayAndSetIndexToZero(){
        //removes user info from array
        String[][]newClubMemberInfo=new String[nonMemberInfo.length-
1][UserInfo.WIDTH_OF_USER_DETAILS_FILE];
        for(int i=0;i<nonMemberIndex;i++){
            for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
```

```
                newClubMemberInfo[i][x]=nonMemberInfo[i][x];
            }
        }
        for(int i=nonMemberIndex+1;i<nonMemberInfo.length;i++){
            for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                newClubMemberInfo[i-1][x]=nonMemberInfo[i][x];
            }
        }
        nonMemberInfo=newClubMemberInfo;
        nonMemberIndex=0;//sets index to zero
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(NonMemberDetails.class.ge
tName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(NonMemberDetails.class.ge
tName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(NonMemberDetails.class.ge
tName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(NonMemberDetails.class.ge
tName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
```

```
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new NonMemberDetails().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton addUserBtn;
    private javax.swing.JLabel addressFourLabel;
    private javax.swing.JTextField addressLineFourTxt;
    private javax.swing.JTextField addressLineOneTxt;
    private javax.swing.JTextField addressLineThreeTxt;
    private javax.swing.JTextField addressLineTwoTxt;
    private javax.swing.JLabel addressOneLabel;
    private javax.swing.JLabel addressThreeLabel;
    private javax.swing.JLabel addressTwoLabel;
    private javax.swing.JButton backBtn;
    private javax.swing.JLabel confirmPasswordLabel;
    private javax.swing.JLabel dateOfBirthLabel;
    private javax.swing.JTextField dateOfBirthTxt;
    private javax.swing.JLabel discordNumberLabel;
    private javax.swing.JTextField discordNumberTxt;
    private javax.swing.JLabel discordUsernameLabel;
    private javax.swing.JTextField discordUsernameTxt;
    private javax.swing.JButton editBtn;
    private javax.swing.JTextField eloTxt;
    private javax.swing.JTextField emailAddressTxt;
    private javax.swing.JLabel firstNameLabel;
    private javax.swing.JTextField firstNameTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
```

```
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton nextBtn;
    private javax.swing.JLabel noUsersLbl;
    private javax.swing.JLabel passwordLabel;
    private javax.swing.JLabel phoneNumberLabel;
    private javax.swing.JTextField phoneNumberTxt;
    private javax.swing.JLabel postcodeLabel;
    private javax.swing.JTextField postcodeTxt;
    private javax.swing.JButton prevBtn;
    private javax.swing.JButton removeBtn;
    private javax.swing.JLabel surnameLabel;
    private javax.swing.JTextField surnameTxt;
    private javax.swing.JTextField usernameTxt;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;
import philipm.cs5.software.development.server.system.UserInfo;



/**
 * This form is used to display all offline games and takes user to
other forms which allow for the creation of offline games
 * @author mortimer
 */
public class OfflinePlay extends javax.swing.JFrame {
    SessionInfo sess;//stores the session information
    public static final String
OFFLINE_GAMES_FOLDER="offlinegames";//stores the destination of the
folder that stores all offline games
    private String[][]gameData=new String[3][4];//stores data of ongoing
offline games
    public static final String SPLIT_FEN_STRING="fensplit*";//this is
```

```
used to split between different FEN's in a given record
    public static final String
NEW_BOARD="rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0
1";//stores the FEN state for a new game
    public static final String LEVEL_ONE_COMP = "Level One
Computer";//stores id's of various opponents
    public static final String LEVEL_TWO_COMP="Level Two Computer";
    public static final String LEVEL_THREE_COMP="Level Three Computer";
    public static final String LEVEL_FOUR_COMP="Level Four Computer";
    public static final String LEVEL_FIVE_COMP="Level Five Computer";
    public static final String HUMAN_OPPONENT="Person";
    //contents of one record storing game:
    //game id,opponentID, isUserWhite, boardState(s),time game has
taken, time white has left, time black has left
    /**
     * Creates new form OfflinePlay
     */
    private OfflinePlay() {
        initComponents();
    }
    /**
     * Creates new form OfflinePlay
     * @param sess The session information of the current user
     */
    public OfflinePlay(SessionInfo sess){
        this.sess=sess;//stores the session
        initComponents();
        updateTable();//updates the table based on the contents of the
offline games text file
        //sets various form attirbutes for clean visuals
        deleteGameBtn.setEnabled(false);playGameBtn.setEnabled(false);
        offlineGamesTbl.getTableHeader().setReorderingAllowed(false);//p
revents columns from being moved by the user
        offlineGamesTbl.setSelectionMode(javax.swing.ListSelectionModel.
SINGLE_SELECTION);//ensures only one row can be selected - making it
visually easier for the user to use
    }
    /**
     * Updates the table displaying the three stored games. Also updates
the 2d array storing the current data for these three games.
     */
    private void updateTable(){
        try{
            File file = new
File(OFFLINE_GAMES_FOLDER+File.separator+sess.getUsername()+".txt");
            file.createNewFile();//this creates a new text file if one
does not already exist
        }catch(IOException ex){
            JOptionPane.showMessageDialog(this,"An unexpected error
occured "+ex,"Error",JOptionPane.OK_OPTION);
        }
        try{
            //gets game data from text file and loads it global 2d array
```

```
            FileReader read = new
FileReader(OFFLINE_GAMES_FOLDER+File.separator+sess.getUsername()+".txt"
);
            BufferedReader buffRead = new BufferedReader(read);
            String lineRead;String lineArr[];
            gameData=new String[3][7];
            int in=0;
            //loads game data into array
            while((lineRead=buffRead.readLine())!=null){
                lineArr = lineRead.split(UserInfo.SPLIT_ITEM);
                for(int i=0;i<lineArr.length;i++){
                    gameData[in][i]=lineArr[i];
                }
                in++;
            }
            buffRead.close();read.close();
        }catch(IOException ex){
            JOptionPane.showMessageDialog(this,"An unexpected error
occured "+ex,"Error",JOptionPane.OK_OPTION);
        }
        //updates table contents to reflect gameData array
        String[]games;//stores all games that have been played so far
        for(int y=0;y<gameData.length;y++){
            if(gameData[y][0]==null){
                offlineGamesTbl.setValueAt("", y, 0);//sets game title
as empty
                offlineGamesTbl.setValueAt("",y,1);//sets opponent empty
                offlineGamesTbl.setValueAt("",y,2);//sets move number
empty
            }else{
                offlineGamesTbl.setValueAt(gameData[y][0], y, 0);//sets
game title
                offlineGamesTbl.setValueAt(gameData[y][1],y,1);//sets
opponent id
                games=gameData[y][3].split(SPLIT_FEN_STRING);
                offlineGamesTbl.setValueAt(games[games.length-1].split("
")[5],y,2);//sets move number
            }
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
```

```
        newGamePersonBtn = new javax.swing.JButton();
        newGameCompBtn = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        offlineGamesTbl = new javax.swing.JTable();
        menuBtn = new javax.swing.JButton();
        playGameBtn = new javax.swing.JButton();
        deleteGameBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Offline");

        newGamePersonBtn.setText("New Game Against Person");
        newGamePersonBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                newGamePersonBtnActionPerformed(evt);
            }
        });

        newGameCompBtn.setText("New Game Against Computer");
        newGameCompBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                newGameCompBtnActionPerformed(evt);
            }
        });

        jLabel1.setText("Offline Play");

        offlineGamesTbl.setModel(new
javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null},
                {null, null, null},
                {null, null, null}
            },
            new String [] {
                "Game Title", "Opponent", "Moves Made"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class,
java.lang.String.class
            };
            boolean[] canEdit = new boolean [] {
                false, false, false
            };

            public Class getColumnClass(int columnIndex) {
```

```
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex)
{
                return canEdit [columnIndex];
            }
        });
        offlineGamesTbl.addFocusListener(new
java.awt.event.FocusAdapter() {
            public void focusGained(java.awt.event.FocusEvent evt) {
                offlineGamesTblFocusGained(evt);
            }
            public void focusLost(java.awt.event.FocusEvent evt) {
                offlineGamesTblFocusLost(evt);
            }
        });
        jScrollPane1.setViewportView(offlineGamesTbl);
        if (offlineGamesTbl.getColumnModel().getColumnCount() > 0) {
            offlineGamesTbl.getColumnModel().getColumn(0).setResizable(f
alse);
            offlineGamesTbl.getColumnModel().getColumn(1).setResizable(f
alse);
            offlineGamesTbl.getColumnModel().getColumn(2).setResizable(f
alse);
        }

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(239, 239, 239)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(newGamePersonBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(newGameCompBtn,
javax.swing.GroupLayout.DEFAULT_SIZE, 367, Short.MAX_VALUE)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(373, 373, 373)
                        .addComponent(jLabel1)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGap(0, 121, Short.MAX_VALUE)
```

```
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 651,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(84, 84, 84))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(45, 45, 45)
                .addComponent(newGamePersonBtn)
                .addGap(58, 58, 58)
                .addComponent(newGameCompBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 110, Short.MAX_VALUE)
                .addContainerGap())
        );

        menuBtn.setText("Menu");
        menuBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                menuBtnActionPerformed(evt);
            }
        });

        playGameBtn.setText("Play Game");
        playGameBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                playGameBtnActionPerformed(evt);
            }
        });

        deleteGameBtn.setText("Delete Game");
        deleteGameBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                deleteGameBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
```

```
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(layout.createSequentialGroup()
                .addGap(59, 59, 59)
                .addComponent(menuBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(deleteGameBtn)
                .addGap(37, 37, 37)
                .addComponent(playGameBtn)
                .addGap(71, 71, 71))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, 17, Short.MAX_VALUE)
                        .addComponent(menuBtn)
                        .addContainerGap())
                    .addGroup(layout.createSequentialGroup()
                        .addGap(1, 1, 1)
                        .addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.BASELINE)
                            .addComponent(playGameBtn)
                            .addComponent(deleteGameBtn))
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT
_SIZE, Short.MAX_VALUE))))
        );

        pack();
    }// </editor-fold>

    private void menuBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user back the main menu
        Menu men = new Menu(sess);
        men.setVisible(true);
        this.dispose();
    }

    private void
newGamePersonBtnActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        //checks to see if a new game can be created
        if(gameData[2][1]!=null){
            JOptionPane.showMessageDialog(this, "Only three games can be
played at any one time. Please delete a game to create a new one.","Too
many games",JOptionPane.OK_OPTION);
            return;
        }
        //takes user to form where they can create a new offline game
        CreateOfflineGameAgainstPerson off = new
CreateOfflineGameAgainstPerson(sess);
        off.setVisible(true);this.dispose();
    }

    private void
newGameCompBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //checks to see if a new game can be created
        if(gameData[2][0]!=null){
            JOptionPane.showMessageDialog(this, "Only three games can be
played at any one time. Please delete a game to create a new one.","Too
many games",JOptionPane.OK_OPTION);
            return;
        }
        //takes user to form where they can create a new offline game
        CreateOfflineGame off = new CreateOfflineGame(sess);
        off.setVisible(true);this.dispose();
    }

    private void offlineGamesTblFocusGained(java.awt.event.FocusEvent
evt) {
        //enables buttons once a row has been selected
        deleteGameBtn.setEnabled(true);playGameBtn.setEnabled(true);
    }

    private void offlineGamesTblFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void playGameBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //takes user to selected game, if it is a valid game
        if(offlineGamesTbl.getSelectedRow()==-1 ||
gameData[offlineGamesTbl.getSelectedRow()][0]==null){//checks to make
sure a game is actually selected
            JOptionPane.showMessageDialog(this, "Please select a valid
game to play.","No Game Selected",JOptionPane.OK_OPTION);
            return;
        }
        String
fen[]=gameData[offlineGamesTbl.getSelectedRow()][3].split(SPLIT_FEN_STRI
NG);//stores all fen strings for games so far
        ChessBoardState states[]=new
```

```
ChessBoardState[fen.length];//stores all boards played so far
        for(int i=0;i<states.length;i++){//loads all states
            states[i]=new ChessBoardState(fen[i]);
        }
        //creates new chess board
        ChessBoardOffline board = new ChessBoardOffline(sess,
gameData[offlineGamesTbl.getSelectedRow()][1], new
ChessGame(states,gameData[offlineGamesTbl.getSelectedRow()][1].equals(LE
VEL_TWO_COMP)),
gameData[offlineGamesTbl.getSelectedRow()][2].equals("true"),
Integer.parseInt(gameData[offlineGamesTbl.getSelectedRow()][4]),Integer.
parseInt(gameData[offlineGamesTbl.getSelectedRow()][5]),Integer.parseInt
(gameData[offlineGamesTbl.getSelectedRow()][6]),gameData[offlineGamesTbl
.getSelectedRow()][0]);
        board.setVisible(true);this.dispose();

    }

    private void deleteGameBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //deletes game stored if it is a valid game to delete
        if(offlineGamesTbl.getSelectedRow()==-1 ||
gameData[offlineGamesTbl.getSelectedRow()][0]==null){//checks to make
sure a game is actually selected
            JOptionPane.showMessageDialog(this, "Please select a valid
game to delete.","No Game Selected",JOptionPane.OK_OPTION);
            return;
        }
        //seeks user confirmation before deleting game
        if(JOptionPane.showConfirmDialog(this, "Are you sure you would
like to delete this game?", "Delete?",
JOptionPane.YES_NO_CANCEL_OPTION)!=JOptionPane.YES_OPTION){
            return;
        }
        //deletes game from text file and array and uptables table
accordingly
        try{
            //writes old data without the game deleted to the offline
game file
            FileWriter write = new
FileWriter(OFFLINE_GAMES_FOLDER+File.separator+sess.getUsername()+".txt"
,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            boolean isFirstWrite=true;
            String line;//stores line of text to write
            //writes new data to file
            for(int y =0;y<gameData.length;y++){
                if(y!=offlineGamesTbl.getSelectedRow()){
                    if(gameData[y][0]!=null){
                        line=gameData[y][0];
                        for(int i=1;i<gameData[0].length;i++){//creates
text to write to file
                            line =line+
```

```
UserInfo.SPLIT_ITEM+gameData[y][i];
                        }
                        if(isFirstWrite==false){
                            buffWrite.newLine();
                        }else{
                            isFirstWrite=false;
                        }
                        buffWrite.write(line);
                    }
                }
            }
            buffWrite.flush();write.flush();buffWrite.close();write.clos
e();//closes and flushes writers
        }catch(IOException ex){
            JOptionPane.showMessageDialog(this,"An unexpected error
occured "+ex,"Error",JOptionPane.OK_OPTION);
        }
        updateTable();//updates table and gameData array
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(OfflinePlay.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(OfflinePlay.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(OfflinePlay.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(OfflinePlay.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new OfflinePlay().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton deleteGameBtn;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton menuBtn;
    private javax.swing.JButton newGameCompBtn;
    private javax.swing.JButton newGamePersonBtn;
    private javax.swing.JTable offlineGamesTbl;
    private javax.swing.JButton playGameBtn;
    // End of variables declaration
```

```
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;
import philipm.cs5.software.development.server.system.UserInfo;



/**
 * This form is used to display all offline games and takes user to
other forms which allow for the creation of offline games
 * @author mortimer
 */
public class OfflinePlay extends javax.swing.JFrame {
    SessionInfo sess;//stores the session information
    public static final String
OFFLINE_GAMES_FOLDER="offlinegames";//stores the destination of the
folder that stores all offline games
    private String[][]gameData=new String[3][4];//stores data of ongoing
offline games
    public static final String SPLIT_FEN_STRING="fensplit*";//this is
used to split between different FEN's in a given record
    public static final String
NEW_BOARD="rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0
1";//stores the FEN state for a new game
    public static final String LEVEL_ONE_COMP = "Level One
Computer";//stores id's of various opponents
    public static final String LEVEL_TWO_COMP="Level Two Computer";
    public static final String LEVEL_THREE_COMP="Level Three Computer";
    public static final String LEVEL_FOUR_COMP="Level Four Computer";
    public static final String LEVEL_FIVE_COMP="Level Five Computer";
    public static final String HUMAN_OPPONENT="Person";
    //contents of one record storing game:
    //game id,opponentID, isUserWhite, boardState(s),time game has
taken, time white has left, time black has left
    /**
```

```
     * Creates new form OfflinePlay
     */
    private OfflinePlay() {
        initComponents();
    }
    /**
     * Creates new form OfflinePlay
     * @param sess The session information of the current user
     */
    public OfflinePlay(SessionInfo sess){
        this.sess=sess;//stores the session
        initComponents();
        updateTable();//updates the table based on the contents of the
offline games text file
        //sets various form attirbutes for clean visuals
        deleteGameBtn.setEnabled(false);playGameBtn.setEnabled(false);
        offlineGamesTbl.getTableHeader().setReorderingAllowed(false);//p
revents columns from being moved by the user
        offlineGamesTbl.setSelectionMode(javax.swing.ListSelectionModel.
SINGLE_SELECTION);//ensures only one row can be selected - making it
visually easier for the user to use
    }
    /**
     * Updates the table displaying the three stored games. Also updates
the 2d array storing the current data for these three games.
     */
    private void updateTable(){
        try{
            File file = new
File(OFFLINE_GAMES_FOLDER+File.separator+sess.getUsername()+".txt");
            file.createNewFile();//this creates a new text file if one
does not already exist
        }catch(IOException ex){
            JOptionPane.showMessageDialog(this,"An unexpected error
occured "+ex,"Error",JOptionPane.OK_OPTION);
        }
        try{
            //gets game data from text file and loads it global 2d array
            FileReader read = new
FileReader(OFFLINE_GAMES_FOLDER+File.separator+sess.getUsername()+".txt"
);
            BufferedReader buffRead = new BufferedReader(read);
            String lineRead;String lineArr[];
            gameData=new String[3][7];
            int in=0;
            //loads game data into array
            while((lineRead=buffRead.readLine())!=null){
                lineArr = lineRead.split(UserInfo.SPLIT_ITEM);
                for(int i=0;i<lineArr.length;i++){
                    gameData[in][i]=lineArr[i];
                }
                in++;
            }
```

```
            buffRead.close();read.close();
        }catch(IOException ex){
            JOptionPane.showMessageDialog(this,"An unexpected error
occured "+ex,"Error",JOptionPane.OK_OPTION);
        }
        //updates table contents to reflect gameData array
        String[]games;//stores all games that have been played so far
        for(int y=0;y<gameData.length;y++){
            if(gameData[y][0]==null){
                offlineGamesTbl.setValueAt("", y, 0);//sets game title
as empty
                offlineGamesTbl.setValueAt("",y,1);//sets opponent empty
                offlineGamesTbl.setValueAt("",y,2);//sets move number
empty
            }else{
                offlineGamesTbl.setValueAt(gameData[y][0], y, 0);//sets
game title
                offlineGamesTbl.setValueAt(gameData[y][1],y,1);//sets
opponent id
                games=gameData[y][3].split(SPLIT_FEN_STRING);
                offlineGamesTbl.setValueAt(games[games.length-1].split("
")[5],y,2);//sets move number
            }
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        newGamePersonBtn = new javax.swing.JButton();
        newGameCompBtn = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        offlineGamesTbl = new javax.swing.JTable();
        menuBtn = new javax.swing.JButton();
        playGameBtn = new javax.swing.JButton();
        deleteGameBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Offline");

        newGamePersonBtn.setText("New Game Against Person");
        newGamePersonBtn.addActionListener(new
```

```
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                newGamePersonBtnActionPerformed(evt);
            }
        });

        newGameCompBtn.setText("New Game Against Computer");
        newGameCompBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                newGameCompBtnActionPerformed(evt);
            }
        });

        jLabel1.setText("Offline Play");

        offlineGamesTbl.setModel(new
javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null},
                {null, null, null},
                {null, null, null}
            },
            new String [] {
                "Game Title", "Opponent", "Moves Made"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class,
java.lang.String.class
            };
            boolean[] canEdit = new boolean [] {
                false, false, false
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex)
{
                return canEdit [columnIndex];
            }
        });
        offlineGamesTbl.addFocusListener(new
java.awt.event.FocusAdapter() {
            public void focusGained(java.awt.event.FocusEvent evt) {
                offlineGamesTblFocusGained(evt);
            }
            public void focusLost(java.awt.event.FocusEvent evt) {
                offlineGamesTblFocusLost(evt);
```

```
            }
        });
        jScrollPane1.setViewportView(offlineGamesTbl);
        if (offlineGamesTbl.getColumnModel().getColumnCount() > 0) {
            offlineGamesTbl.getColumnModel().getColumn(0).setResizable(f
alse);
            offlineGamesTbl.getColumnModel().getColumn(1).setResizable(f
alse);
            offlineGamesTbl.getColumnModel().getColumn(2).setResizable(f
alse);
        }

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(239, 239, 239)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(newGamePersonBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(newGameCompBtn,
javax.swing.GroupLayout.DEFAULT_SIZE, 367, Short.MAX_VALUE)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(373, 373, 373)
                        .addComponent(jLabel1)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGap(0, 121, Short.MAX_VALUE)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 651,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(84, 84, 84))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(45, 45, 45)
                .addComponent(newGamePersonBtn)
                .addGap(58, 58, 58)
                .addComponent(newGameCompBtn)
```

```
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 110, Short.MAX_VALUE)
                .addContainerGap())
        );

        menuBtn.setText("Menu");
        menuBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                menuBtnActionPerformed(evt);
            }
        });

        playGameBtn.setText("Play Game");
        playGameBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                playGameBtnActionPerformed(evt);
            }
        });

        deleteGameBtn.setText("Delete Game");
        deleteGameBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                deleteGameBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGroup(layout.createSequentialGroup()
                .addGap(59, 59, 59)
                .addComponent(menuBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(deleteGameBtn)
                .addGap(37, 37, 37)
                .addComponent(playGameBtn)
                .addGap(71, 71, 71))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
```

```
.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLa
yout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, 17, Short.MAX_VALUE)
                        .addComponent(menuBtn)
                        .addContainerGap())
                    .addGroup(layout.createSequentialGroup()
                        .addGap(1, 1, 1)
                        .addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.BASELINE)
                            .addComponent(playGameBtn)
                            .addComponent(deleteGameBtn))
                        .addContainerGap(javax.swing.GroupLayout.DEFAULT
_SIZE, Short.MAX_VALUE))))
        );

        pack();
    }// </editor-fold>

    private void menuBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user back the main menu
        Menu men = new Menu(sess);
        men.setVisible(true);
        this.dispose();
    }

    private void
newGamePersonBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //checks to see if a new game can be created
        if(gameData[2][1]!=null){
            JOptionPane.showMessageDialog(this, "Only three games can be
played at any one time. Please delete a game to create a new one.","Too
many games",JOptionPane.OK_OPTION);
            return;
        }
        //takes user to form where they can create a new offline game
        CreateOfflineGameAgainstPerson off = new
CreateOfflineGameAgainstPerson(sess);
        off.setVisible(true);this.dispose();
    }

    private void
newGameCompBtnActionPerformed(java.awt.event.ActionEvent evt)
```

```
{
        //checks to see if a new game can be created
        if(gameData[2][0]!=null){
            JOptionPane.showMessageDialog(this, "Only three games can be
played at any one time. Please delete a game to create a new one.","Too
many games",JOptionPane.OK_OPTION);
            return;
        }
        //takes user to form where they can create a new offline game
        CreateOfflineGame off = new CreateOfflineGame(sess);
        off.setVisible(true);this.dispose();
    }

    private void offlineGamesTblFocusGained(java.awt.event.FocusEvent
evt) {
        //enables buttons once a row has been selected
        deleteGameBtn.setEnabled(true);playGameBtn.setEnabled(true);
    }

    private void offlineGamesTblFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void playGameBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //takes user to selected game, if it is a valid game
        if(offlineGamesTbl.getSelectedRow()==-1 ||
gameData[offlineGamesTbl.getSelectedRow()][0]==null){//checks to make
sure a game is actually selected
            JOptionPane.showMessageDialog(this, "Please select a valid
game to play.","No Game Selected",JOptionPane.OK_OPTION);
            return;
        }
        String
fen[]=gameData[offlineGamesTbl.getSelectedRow()][3].split(SPLIT_FEN_STRI
NG);//stores all fen strings for games so far
        ChessBoardState states[]=new
ChessBoardState[fen.length];//stores all boards played so far
        for(int i=0;i<states.length;i++){//loads all states
            states[i]=new ChessBoardState(fen[i]);
        }
        //creates new chess board
        ChessBoardOffline board = new ChessBoardOffline(sess,
gameData[offlineGamesTbl.getSelectedRow()][1], new
ChessGame(states,gameData[offlineGamesTbl.getSelectedRow()][1].equals(LE
VEL_TWO_COMP)),
gameData[offlineGamesTbl.getSelectedRow()][2].equals("true"),
Integer.parseInt(gameData[offlineGamesTbl.getSelectedRow()][4]),Integer.
parseInt(gameData[offlineGamesTbl.getSelectedRow()][5]),Integer.parseInt
(gameData[offlineGamesTbl.getSelectedRow()][6]),gameData[offlineGamesTbl
.getSelectedRow()][0]);
        board.setVisible(true);this.dispose();
```

```
    }

    private void deleteGameBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //deletes game stored if it is a valid game to delete
        if(offlineGamesTbl.getSelectedRow()==-1 ||
gameData[offlineGamesTbl.getSelectedRow()][0]==null){//checks to make
sure a game is actually selected
            JOptionPane.showMessageDialog(this, "Please select a valid
game to delete.","No Game Selected",JOptionPane.OK_OPTION);
            return;
        }
        //seeks user confirmation before deleting game
        if(JOptionPane.showConfirmDialog(this, "Are you sure you would
like to delete this game?", "Delete?",
JOptionPane.YES_NO_CANCEL_OPTION)!=JOptionPane.YES_OPTION){
            return;
        }
        //deletes game from text file and array and uptables table
accordingly
        try{
            //writes old data without the game deleted to the offline
game file
            FileWriter write = new
FileWriter(OFFLINE_GAMES_FOLDER+File.separator+sess.getUsername()+".txt"
,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            boolean isFirstWrite=true;
            String line;//stores line of text to write
            //writes new data to file
            for(int y =0;y<gameData.length;y++){
                if(y!=offlineGamesTbl.getSelectedRow()){
                    if(gameData[y][0]!=null){
                        line=gameData[y][0];
                        for(int i=1;i<gameData[0].length;i++){//creates
text to write to file
                            line =line+
UserInfo.SPLIT_ITEM+gameData[y][i];
                        }
                        if(isFirstWrite==false){
                            buffWrite.newLine();
                        }else{
                            isFirstWrite=false;
                        }
                        buffWrite.write(line);
                    }
                }
            }
            buffWrite.flush();write.flush();buffWrite.close();write.clos
e();//closes and flushes writers
        }catch(IOException ex){
            JOptionPane.showMessageDialog(this,"An unexpected error
```

```
occured "+ex,"Error",JOptionPane.OK_OPTION);
        }
        updateTable();//updates table and gameData array
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(OfflinePlay.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(OfflinePlay.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(OfflinePlay.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(OfflinePlay.class.getName
()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
```

```
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new OfflinePlay().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton deleteGameBtn;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton menuBtn;
    private javax.swing.JButton newGameCompBtn;
    private javax.swing.JButton newGamePersonBtn;
    private javax.swing.JTable offlineGamesTbl;
    private javax.swing.JButton playGameBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import javax.swing.JOptionPane;
import javax.swing.Timer;

/**
 * Creates a new form that displays the online menu.
 * @author mortimer
 */
public class OnlineMenu extends javax.swing.JFrame {
    private SessionInfo session;//stores the session information
    private String gameData[][]=new
String[3][WIDTH_OF_ONLINE_RECORD];//stores game data array
    public static final int WIDTH_OF_ONLINE_RECORD=14;//stores the width
of an online game record
    private  TableRenderer renderer;//renders the online games table
    private Timer updateTableTimer;//used to periodically update the
online table by requesting data from the server
    /**
     * Creates new form OnlineMenu
     */
    private OnlineMenu() {
        initComponents();
    }
    /**
     * Creates new online form
     * @param sess The session information
     */
    public OnlineMenu(SessionInfo sess){
        session=sess;
        initComponents();
        //intilaises various visual aspects of the form
        playBtn.setEnabled(false);deleteBtn.setEnabled(false);
        onlineGamesTbl.getTableHeader().setReorderingAllowed(false);//pr
events columns from being moved by the user
        onlineGamesTbl.setSelectionMode(javax.swing.ListSelectionModel.S
INGLE_SELECTION);//ensures only one row can be selected - making it
visually easier for the user to use
        renderer=new TableRenderer(this);//creates a custom table
renderer
        onlineGamesTbl.getColumnModel().getColumn(0).setCellRenderer(ren
derer);
        onlineGamesTbl.getColumnModel().getColumn(1).setCellRenderer(ren
derer);
        onlineGamesTbl.getColumnModel().getColumn(2).setCellRenderer(ren
derer);
        updateTable();//updates the table by getting the online records
from the server
        //creates a timer that periodically updates the online game
table with data from the server
        updateTableTimer=new Timer(10000, new ActionListener(){
            /**
             * Updates the table using the timer every x number of
milliseconds.
```

```
             * @param e The ActionEvent from the timer indicating that
the table should be updated
             */
            @Override
            public void actionPerformed(ActionEvent e) {
                updateTable();//updates thte table
            }
        });updateTableTimer.start();//starts the timer
    }
    /**
     * Sets the play button as enabled or disabled
     * @param enable Whether or not the play button should be enabled
     */
    public void setPlayButtonEnabled(boolean enable){
        playBtn.setEnabled(enable);
    }
    /**
     * Sets the delete button as enabled or disabled
     * @param enable Whether or not the delete button should be enabled
     */
    public void setDeleteButtonEnabled(boolean enable){
        deleteBtn.setEnabled(enable);
    }
    /**
     * Connects to server and attempts to update table of all games
     */
    private void updateTable(){
        try{
            String[]games;
            //sets table to empty first
            for(int y=0;y<3;y++){
                for(int x=0;x<3;x++){
                    onlineGamesTbl.setValueAt("",y,x);
                }
            }
            //stores whether a given row should be highlighted based on
whether a game actually exists in that row of the table
            boolean isRowOneHighlighted=true;
            boolean isRowTwoHighlighted=true;
            boolean isRowThreeHighlighted=true;
            gameData=ThreadClient.getOnlineGameData(session.getUsername(
));//gets game data from server
            //updates table based on game data array
            for(int y=0;y<gameData.length;y++){
                onlineGamesTbl.setValueAt(gameData[y][0],y,0);//sets
game code
                //sets the username of the opponent
                if(session.getUsername().equals(gameData[y][1])){
                    onlineGamesTbl.setValueAt(gameData[y][2],y,1);
                }else{
                    onlineGamesTbl.setValueAt(gameData[y][1],y,1);
                }
                games=gameData[y][3].split(OfflinePlay.SPLIT_FEN_STRING)
```

```
;
                if(String.valueOf(onlineGamesTbl.getValueAt(y,
1)).equals("")){//if the game has not yet been created, the field is set
as empty and the row is rendered as a different colour
                    switch (y) {
                        case 0:
                            isRowOneHighlighted=false;
                            break;
                        case 1:
                            isRowTwoHighlighted=false;
                            break;
                        default:
                            isRowThreeHighlighted=false;
                            break;
                    }
                    onlineGamesTbl.setValueAt("",y,2);
                }else{
                    onlineGamesTbl.setValueAt(games[games.length-
1].split(" ")[5],y,2);//gets the move number
                }
            }
            renderer.updateRowColour(isRowOneHighlighted,
isRowTwoHighlighted, isRowThreeHighlighted);//updates highlighter
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occured when
retriving game data from the server. Please check your network settings.
Error Details: "+e.getMessage(),"Error",JOptionPane.OK_OPTION);
            Menu menu = new Menu(session);
            menu.setVisible(true);
            this.dispose();return;//displays error message and takes
user back to menu
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        gameAgainstPersonBtn = new javax.swing.JButton();
        gameAgainstPersonBtn1 = new javax.swing.JButton();
        playGameAgainstFriendEnterCodeBtn = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        onlineGamesTbl = new javax.swing.JTable();
        menuBtn = new javax.swing.JButton();
```

```
        playBtn = new javax.swing.JButton();
        deleteBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Title");

        jLabel1.setText("Online Play");

        gameAgainstPersonBtn.setText("New Game Against Random Player");
        gameAgainstPersonBtn.setToolTipText("");
        gameAgainstPersonBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                gameAgainstPersonBtnActionPerformed(evt);
            }
        });

        gameAgainstPersonBtn1.setText("Create Game Against Friend");
        gameAgainstPersonBtn1.setToolTipText("");
        gameAgainstPersonBtn1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                gameAgainstPersonBtn1ActionPerformed(evt);
            }
        });

        playGameAgainstFriendEnterCodeBtn.setText("Play Against Friend -
Enter Game Code");
        playGameAgainstFriendEnterCodeBtn.setToolTipText("");
        playGameAgainstFriendEnterCodeBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                playGameAgainstFriendEnterCodeBtnActionPerformed(evt);
            }
        });

        onlineGamesTbl.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null},
                {null, null, null},
                {null, null, null}
            },
            new String [] {
                "Game Code", "Opponent", "Move Number"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.String.class, java.lang.String.class,
java.lang.String.class
```

```
            };
            boolean[] canEdit = new boolean [] {
                false, false, false
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }

            public boolean isCellEditable(int rowIndex, int columnIndex)
{
                return canEdit [columnIndex];
            }
        });
        onlineGamesTbl.addFocusListener(new
java.awt.event.FocusAdapter() {
            public void focusGained(java.awt.event.FocusEvent evt) {
                onlineGamesTblFocusGained(evt);
            }
            public void focusLost(java.awt.event.FocusEvent evt) {
                onlineGamesTblFocusLost(evt);
            }
        });
        jScrollPane1.setViewportView(onlineGamesTbl);
        if (onlineGamesTbl.getColumnModel().getColumnCount() > 0) {
            onlineGamesTbl.getColumnModel().getColumn(0).setResizable(fa
lse);
            onlineGamesTbl.getColumnModel().getColumn(1).setResizable(fa
lse);
            onlineGamesTbl.getColumnModel().getColumn(2).setResizable(fa
lse);
        }

        menuBtn.setText("Menu");
        menuBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                menuBtnActionPerformed(evt);
            }
        });

        playBtn.setText("Play");
        playBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                playBtnActionPerformed(evt);
            }
        });

        deleteBtn.setText("Delete");
        deleteBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
```

```
{
                deleteBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap(141, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(gameAgainstPersonBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(gameAgainstPersonBtn1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(playGameAgainstFriendEnterCode
Btn, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(304, 304, 304))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 651,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(101, 101, 101))))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(391, 391, 391)
                .addComponent(jLabel1)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(29, 29, 29)
                .addComponent(menuBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(deleteBtn)
                .addGap(18, 18, 18)
                .addComponent(playBtn)
                .addGap(50, 50, 50))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
```

```
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(18, 18, 18)
                .addComponent(gameAgainstPersonBtn)
                .addGap(18, 18, 18)
                .addComponent(gameAgainstPersonBtn1)
                .addGap(18, 18, 18)
                .addComponent(playGameAgainstFriendEnterCodeBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 65, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 114,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(53, 53, 53)
                        .addComponent(menuBtn)
                        .addGap(15, 15, 15))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(playBtn)
                            .addComponent(deleteBtn))
                        .addGap(39, 39, 39))))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void onlineGamesTblFocusGained(java.awt.event.FocusEvent
evt) {
        //enables play and delete button once a row has been selected
```

```
        playBtn.setEnabled(true);deleteBtn.setEnabled(true);
    }

    private void onlineGamesTblFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void menuBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes user back to menu
        Menu menu = new Menu(session);
        menu.setVisible(true);
        this.dispose();
    }

    private void
gameAgainstPersonBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //checks to ensure that the user is in two or fewer games
currently
        try{
            Integer.parseInt(String.valueOf(onlineGamesTbl.getValueAt(2,
0)));
            JOptionPane.showMessageDialog(this, "You can only play up to
three online games at any one time.","Error",JOptionPane.OK_OPTION);
            return;
        }catch(NumberFormatException e){
        }
        try{
            ThreadClient.createNewRandomOnlineGame(session.getUsername()
);//creates a new random online game
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occured when
communicating with the server. Please check your network settings. Error
Details: "+e,"Error",JOptionPane.OK_OPTION);
        }
        updateTable();//updates the table to reflect the change
    }

    private void deleteBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        //attempts to delete a game that can be deleted.
        //only games where the opponent has net yet joined can be
deleted
        if(onlineGamesTbl.getSelectedRow()>=0 &&
onlineGamesTbl.getSelectedRow()<=2){
            if(onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow()
,1).equals("")==false){
                JOptionPane.showMessageDialog(this,"You can only delete
games that an opponent has not yet joined. Please go into the game and
resign if you wish to remove the game.","Unable to
Delete",JOptionPane.OK_OPTION);
```

```
                return;
            }
            //asks for confirmation before deleting games
            int confirm=JOptionPane.showConfirmDialog(this,"Are you sure
you would like to delete this
game?","Delete?",JOptionPane.YES_NO_CANCEL_OPTION);
            if(confirm!=JOptionPane.YES_OPTION){
                return;
            }
            try{
                ThreadClient.deleteOnlineGame(Integer.parseInt(String.va
lueOf(onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow(),0))),
0.5);//deletes game
                updateTable();//updates the tab;e
            }catch(Exception  e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server. Please check your network settings. Error
details: "+e.getMessage(),"Error",JOptionPane.OK_OPTION);
                updateTable();
                return;
            }
        }else{
            JOptionPane.showMessageDialog(this, "Please select a valid
game to delete","Error",JOptionPane.OK_OPTION);
            return;
        }
    }

    private void
gameAgainstPersonBtn1ActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes user to form to create game against a friend
        if(onlineGamesTbl.getValueAt(2,0).equals("")==false){
            JOptionPane.showMessageDialog(this, "You can only have three
games at any one time.","Error",JOptionPane.OK_OPTION);
            return;
        }
        CreateGameAgainstFriend game = new
CreateGameAgainstFriend(session);
        game.setVisible(true);
        this.dispose();
    }

    private void
playGameAgainstFriendEnterCodeBtnActionPerformed(java.awt.event.ActionEv
ent evt)
{
        // allows user to input game code and play against friend
        if(onlineGamesTbl.getValueAt(2,0).equals("")==false){//ensures
that user does not join more than three games
            JOptionPane.showMessageDialog(this, "You can only have three
games at any one time.","Error",JOptionPane.OK_OPTION);
            return;
```

```
        }
        String code =JOptionPane.showInputDialog(this,"Please enter the
game code provided by a friend:","Enter
code",JOptionPane.OK_CANCEL_OPTION);
        if(code==null){//if cancel is pressed, nothing happens
            return;
        }
        try{
            int codeInt=Integer.parseInt(code);//tries converting code
to an integer
            if(codeInt<0){
                //type and range check to ensure that the game code is
possibly valid
                JOptionPane.showMessageDialog(this,"The game code must
be a valid integer greater than 0","Invalid
code",JOptionPane.OK_OPTION);
                return;
            }
            try{
                boolean succ =
ThreadClient.joinGameUsingFriendCode(codeInt,
session.getUsername());//attempts to join the game
                if(succ==false){
                    JOptionPane.showMessageDialog(this,"Unable to join
game. The code entered is invalid or can't be used to join a
game","Error",JOptionPane.OK_OPTION);
                    updateTable();
                    return;
                }
                updateTable();//updates the table
            }catch(Exception e){
                 JOptionPane.showMessageDialog(this,"Error when
communicating with the server. Please check your network settings. Error
details: "+e.getMessage(),"Error",JOptionPane.OK_OPTION);
                 updateTable();
                 return;
            }
        }catch(NumberFormatException e){
            JOptionPane.showMessageDialog(this,"The game code must be a
valid integer greater than 0","Invalid code",JOptionPane.OK_OPTION);
            return;
        }
    }

    private void playBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes the user to the selected game
        if(onlineGamesTbl.getSelectedRow()<0 ||
onlineGamesTbl.getSelectedRow()>2){//ensures that a valid game is
selected
            JOptionPane.showMessageDialog(this, "Invalid game
selected.","Invalid Game",JOptionPane.OK_OPTION);
            return;
```

```
        }
        //checks to ensure game selected is playable
        if(onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow(),
0).equals("") ||
onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow(),
1).equals("")){
            JOptionPane.showMessageDialog(this, "Invalid game
selected.","Invalid Game",JOptionPane.OK_OPTION);
            return;
        }
        //gets the most recent version of the data from the server
        String []data=null;
        try{
            data=ThreadClient.getGameData(Integer.parseInt(String.valueO
f(onlineGamesTbl.getValueAt(onlineGamesTbl.getSelectedRow(),0))));//gets
updated version of game record before joining
        }catch(Exception e){
            JOptionPane.showMessageDialog(this, "Error when
communicating with the server. Error details: "+e,"Invalid
Game",JOptionPane.OK_OPTION);
            return;
        }
        //checks to see if game is already over - if it is, an
appropriate message is displayed and the game is deleted
        boolean isUserPlayerOne=data[1].equals(session.getUsername());
        if(isUserPlayerOne && data[8].equals("false") &&
data[9].equals("true")){
            if(data[11].equals("1")){
                JOptionPane.showMessageDialog(this,"You won this
game.","Win",JOptionPane.OK_OPTION);
            }else if(data[11].equals("0.5")){
                JOptionPane.showMessageDialog(this,"This game was a
draw.","Draw",JOptionPane.OK_OPTION);
            }else{
                JOptionPane.showMessageDialog(this,"You lost this
game.","Loss",JOptionPane.OK_OPTION);
            }
            data[8]="true";
            try{
                ThreadClient.updateOnlineGameRecord(data);
            }catch(Exception e){
                JOptionPane.showMessageDialog(this, "Error
when  communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
                updateTable();
                return;
            }
            updateTable();
            return;
        }else if(!isUserPlayerOne && data[9].equals("false") &&
data[8].equals("true")){
            if(data[11].equals("1")){
                JOptionPane.showMessageDialog(this,"You lost this
```

```
game.","Loss",JOptionPane.OK_OPTION);
            }else if(data[11].equals("0.5")){
                JOptionPane.showMessageDialog(this,"This game was a
draw.","Draw",JOptionPane.OK_OPTION);
            }else{
                JOptionPane.showMessageDialog(this,"You won this
game.","Win",JOptionPane.OK_OPTION);
            }
            data[9]="true";
            try{
                ThreadClient.updateOnlineGameRecord(data);
            }catch(Exception e){
                JOptionPane.showMessageDialog(this, "Error
when  communicating with the server. Error details:
"+e,"Error",JOptionPane.OK_OPTION);
                updateTable();
                return;
            }
            updateTable();
            return;
        }
        //games is playable, hence user is taken to it
        //creates the chess game object
        String fenData[]=data[3].split(OfflinePlay.SPLIT_FEN_STRING);
        ChessBoardState[]statesSoFar=new
ChessBoardState[fenData.length];
        for(int board=0;board<fenData.length;board++){
            statesSoFar[board]=new ChessBoardState(fenData[board]);
        }
        //takes the user to play the game
        ChessGame game = new ChessGame(statesSoFar, false);
         ChessBoardOnline   onlineGame = new
ChessBoardOnline(session,game,data);onlineGame.setVisible(true);this.dis
pose();

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
```

```
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(OnlineMenu.class.getName(
)).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(OnlineMenu.class.getName(
)).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(OnlineMenu.class.getName(
)).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(OnlineMenu.class.getName(
)).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new OnlineMenu().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton deleteBtn;
    private javax.swing.JButton gameAgainstPersonBtn;
    private javax.swing.JButton gameAgainstPersonBtn1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JButton menuBtn;
    private javax.swing.JTable onlineGamesTbl;
    private javax.swing.JButton playBtn;
    private javax.swing.JButton playGameAgainstFriendEnterCodeBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;
```

```
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.LinkedHashSet;
import javax.imageio.ImageIO;



/**
 * This is the main class used to create the log in form.
 * @author mortimer
 */
public class PhilipMCS5SoftwareDevelopmentUserSystem {

    /**
     * Creates and sets the log in form as visible
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        LogIn log = new LogIn();//sets log in form visible
        log.setVisible(true);
    }
    /**
     * This method is used to preprocess images of the chess pieces and
convert and save them to a format that can be used.
     */
    private static void getPiecesWrite(){
        String [] IMAGES_NAME = {"White King","White Queen","White
Bishop","White Knight","White Rook","White Pawn","Black King","Black
Queen","Black Bishop","Black Knight","Black Rook","Black Pawn"};//stores
the file names of the images of the pieces
        try{
            BufferedImage img=ImageIO.read(new
File("Assets"+File.separator+"allPieces(2).png"));//gets the image of
all the pieces
            int j=0;
            System.out.println(img.getWidth()+"
"+img.getHeight());//prints the width and height of the image
            //loops through the big image and gets all the sub images
which represent indivudal pieces
            BufferedImage subImage;int k=0;
            for(int y=0;y<640;y=y+320){
                for(int i=0;i<1920;i=i+320){
                    subImage=img.getSubimage(i, y, 320, 320);
                    ImageIO.write(subImage, "png", new
File("Assets"+File.separator+IMAGES_NAME[k++]+".png"));//writes the
image of the chess piece to the file
```

```
                }
            }
        }catch(Exception e){
            System.out.println(e);
        }
    }
    /**
     * Processes the opening book used so that it no longer contains
duplicate openings and irrelevant data.
     */
    private static void processOpeningBook(){
        try{
            //removes any move numbers from the fen strings if
appropriate
            FileReader read = new FileReader("openingMoves.txt");
            BufferedReader buffRead = new BufferedReader(read);
            String data[]=buffRead.lines().toArray(String[]::new);
            buffRead.close();read.close();
            //ensures that the full move number is not included in the
FEN as it is irrelevant
            if(data[0].split(",")[0].split(" ").length==65){
                String fen1[];String fen2[];String line[];
                for(int i=0;i<data.length;i++){
                    line= data[i].split(",");
                    fen1=line[0].split(" ");
                    fen2=line[1].split(" ");
                    data[i]=fen1[0]+" "+fen1[1]+" "+fen1[2]+"
"+fen1[3]+" "+fen1[4]+","+fen2[0]+" "+fen2[1]+" "+fen2[2]+" "+fen2[3]+"
"+fen2[4];
                }
            }
            //removes duplicates from file
            LinkedHashSet<String> linkedHashSet = new LinkedHashSet<>(
Arrays.asList(data) );
            System.out.println("Length before removing duplicates "+
data.length);
            data = linkedHashSet.toArray(new String[]{});
            System.out.println("Length after removing duplicates "+
data.length);
            String newData[][]=new String[data.length][1];
            for(int i=0;i<data.length;i++){
                newData[i][0]=data[i];
            }
            Sort.mergeSort(newData);//sorts data
            //writes array back to text file
            FileWriter write = new FileWriter("openingMoves.txt",false);
            BufferedWriter buffWrite =new BufferedWriter(write);
            buffWrite.write(newData[0][0]);
            for(int i=1;i<newData.length;i++){
                buffWrite.newLine();
                buffWrite.write(newData[i][0]);
            }buffWrite.flush();write.flush();
            buffWrite.close();write.close();//flushes and closes writers
```

```
        }catch(IOException e){
            System.out.println("Error: "+e);
        }
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.Color;
import javax.swing.JOptionPane;


/**
 * Creates form which lets the chess club leader handle requests to join
the chess club
 * @author mortimer
 */
public class RequestsToJoinClub extends javax.swing.JFrame {

    private SessionInfo session;//stores the session information
    private String[][]requestsToJoinClub;//stores the data of requests
to join the club
    private String[]emails;//stores the email addresses of the user's
requesting to join the club
    private int currentReqeustIndex=0;//stores the index which points to
the user info of the current person being viewed
    /**
     * Creates new form CreateAccount
     * @param session The session information
     * @throws java.lang.Exception An exception may occur when
attempting to load requests.
     */
    public RequestsToJoinClub(SessionInfo session) throws Exception{
        initComponents();
        this.session=session;
        noRequestsLbl.setVisible(false);//sets the label that displays
that there are no requests to handle invisible initially
        approveBtn.setBackground(Color.green);denyBtn.setBackground(Colo
r.red);
        requestsToJoinClub=ThreadClient.getAllRequestsToJoinClub();//get
s all requests to join the club and the user info of each person
requesting to join the club
```

```
        emails = new String[requestsToJoinClub.length];
        for(int req=0;req<emails.length;req++){//gets the email address
of each person requesting to join the club
            emails[req]=ThreadClient.getEmailAddressOfUser(requestsToJoi
nClub[req][0]);
        }
        checkIfThereAreMoreRequestsAndUpdateComponents();
        loadRequestToFormIfAppropraite();
    }
    /**
     * Creates new form CreateAccount
     */
    private RequestsToJoinClub(){
        initComponents();
    }
    /**
     * Loads user info onto the form if there are any more requests to
load
     */
    private void loadRequestToFormIfAppropraite(){
        if(currentReqeustIndex>=requestsToJoinClub.length ||
requestsToJoinClub.length==0){//if there are no valid requests to
handle, this method does nothing
            return;
        }
        //loads the user data onto the form
        //Username, password, first name, surname, postcode, addressline
one, address line two, address line three, address line four, discord
username, discord number, date of birth, phone number, isClubLeader,
isClubMember, ELO, hasUserRequestedToJoinClub
        usernameTxt.setText(requestsToJoinClub[currentReqeustIndex][0]);
        emailAddressTxt.setText(emails[currentReqeustIndex]);
        firstNameTxt.setText(requestsToJoinClub[currentReqeustIndex][2])
;surnameTxt.setText(requestsToJoinClub[currentReqeustIndex][3]);
        postcodeTxt.setText(requestsToJoinClub[currentReqeustIndex][4]);
addressLineOneTxt.setText(requestsToJoinClub[currentReqeustIndex][5]);
        addressLineTwoTxt.setText(requestsToJoinClub[currentReqeustIndex
][6]);addressLineThreeTxt.setText(requestsToJoinClub[currentReqeustIndex
][7]);
        addressLineFourTxt.setText(requestsToJoinClub[currentReqeustInde
x][8]);discordUsernameTxt.setText(requestsToJoinClub[currentReqeustIndex
][9]);
        discordNumberTxt.setText(requestsToJoinClub[currentReqeustIndex]
[10]);dateOfBirthTxt.setText(requestsToJoinClub[currentReqeustIndex][11]
);
        phoneNumberTxt.setText(requestsToJoinClub[currentReqeustIndex][1
2]);eloTxt.setText(requestsToJoinClub[currentReqeustIndex][15]);
    }
    /**
     * Checks to see if there are any more requests. If there are no
requests to handle, components are updated.
     */
    private void checkIfThereAreMoreRequestsAndUpdateComponents(){
```

```
        if(currentReqeustIndex>=requestsToJoinClub.length ||
requestsToJoinClub.length==0){
            denyBtn.setEnabled(false);approveBtn.setEnabled(false);
            JOptionPane.showMessageDialog(this,"There are no requests to
handle.","No requests",JOptionPane.OK_OPTION);
            noRequestsLbl.setVisible(true);
            clearForm();
        }
    }
    /**
     * Sets all fields on the form empty
     */
    private void clearForm(){
        usernameTxt.setText("");emailAddressTxt.setText("");
        firstNameTxt.setText("");surnameTxt.setText("");postcodeTxt.setT
ext("");
        eloTxt.setText("");addressLineOneTxt.setText("");addressLineTwoT
xt.setText("");
        addressLineThreeTxt.setText("");addressLineFourTxt.setText("");
        discordUsernameTxt.setText("");discordNumberTxt.setText("");
        dateOfBirthTxt.setText("");phoneNumberTxt.setText("");
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
        jLabel12 = new javax.swing.JLabel();
        jLabel13 = new javax.swing.JLabel();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        usernameTxt = new javax.swing.JTextField();
        emailAddressTxt = new javax.swing.JTextField();
        firstNameTxt = new javax.swing.JTextField();
        surnameTxt = new javax.swing.JTextField();
        postcodeTxt = new javax.swing.JTextField();
```

```
        addressLineOneTxt = new javax.swing.JTextField();
        addressLineTwoTxt = new javax.swing.JTextField();
        addressLineThreeTxt = new javax.swing.JTextField();
        addressLineFourTxt = new javax.swing.JTextField();
        discordUsernameTxt = new javax.swing.JTextField();
        discordNumberTxt = new javax.swing.JTextField();
        dateOfBirthTxt = new javax.swing.JTextField();
        phoneNumberTxt = new javax.swing.JTextField();
        backBtn = new javax.swing.JButton();
        approveBtn = new javax.swing.JButton();
        passwordLabel = new javax.swing.JLabel();
        confirmPasswordLabel = new javax.swing.JLabel();
        firstNameLabel = new javax.swing.JLabel();
        surnameLabel = new javax.swing.JLabel();
        postcodeLabel = new javax.swing.JLabel();
        addressOneLabel = new javax.swing.JLabel();
        addressTwoLabel = new javax.swing.JLabel();
        addressThreeLabel = new javax.swing.JLabel();
        addressFourLabel = new javax.swing.JLabel();
        discordUsernameLabel = new javax.swing.JLabel();
        discordNumberLabel = new javax.swing.JLabel();
        dateOfBirthLabel = new javax.swing.JLabel();
        phoneNumberLabel = new javax.swing.JLabel();
        jLabel20 = new javax.swing.JLabel();
        eloTxt = new javax.swing.JTextField();
        denyBtn = new javax.swing.JButton();
        noRequestsLbl = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Requests To Join Club");

        jLabel1.setText("Username");

        jLabel2.setText("Email Address");

        jLabel5.setText("First Name");

        jLabel6.setText("Surname");

        jLabel7.setText("Postcode");

        jLabel8.setText("Address Line 1");

        jLabel9.setText("Address Line 2");

        jLabel10.setText("Address Line 3");

        jLabel11.setText("Address Line 4");

        jLabel12.setText("Discord Username");

        jLabel13.setText("Discord Number");
```

```
        jLabel14.setText("Date of Birth");

        jLabel15.setText("Phone Number");

        usernameTxt.setEditable(false);
        usernameTxt.setFocusCycleRoot(true);
        usernameTxt.addFocusListener(new java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                usernameTxtFocusLost(evt);
            }
        });
        usernameTxt.addKeyListener(new java.awt.event.KeyAdapter() {
            public void keyTyped(java.awt.event.KeyEvent evt) {
                usernameTxtKeyTyped(evt);
            }
        });

        emailAddressTxt.setEditable(false);
        emailAddressTxt.setFocusCycleRoot(true);
        emailAddressTxt.addFocusListener(new
java.awt.event.FocusAdapter() {
            public void focusLost(java.awt.event.FocusEvent evt) {
                emailAddressTxtFocusLost(evt);
            }
        });

        firstNameTxt.setEditable(false);
        firstNameTxt.setFocusCycleRoot(true);

        surnameTxt.setEditable(false);

        postcodeTxt.setEditable(false);

        addressLineOneTxt.setEditable(false);

        addressLineTwoTxt.setEditable(false);

        addressLineThreeTxt.setEditable(false);

        addressLineFourTxt.setEditable(false);

        discordUsernameTxt.setEditable(false);

        discordNumberTxt.setEditable(false);

        dateOfBirthTxt.setEditable(false);
        dateOfBirthTxt.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                dateOfBirthTxtActionPerformed(evt);
            }
```

```
        });

        phoneNumberTxt.setEditable(false);

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        approveBtn.setText("Approve");
        approveBtn.addActionListener(new java.awt.event.ActionListener()
{
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                approveBtnActionPerformed(evt);
            }
        });

        passwordLabel.setForeground(new java.awt.Color(255, 51, 51));

        confirmPasswordLabel.setForeground(new java.awt.Color(255, 51,
51));

        firstNameLabel.setForeground(new java.awt.Color(255, 51, 51));

        surnameLabel.setForeground(new java.awt.Color(255, 51, 51));

        postcodeLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressOneLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressTwoLabel.setForeground(new java.awt.Color(255, 51, 51));

        addressThreeLabel.setForeground(new java.awt.Color(255, 51,
51));

        addressFourLabel.setForeground(new java.awt.Color(255, 51, 51));

        discordUsernameLabel.setForeground(new java.awt.Color(255, 51,
51));

        discordNumberLabel.setForeground(new java.awt.Color(255, 51,
51));

        dateOfBirthLabel.setForeground(new java.awt.Color(255, 51, 51));

        phoneNumberLabel.setForeground(new java.awt.Color(255, 51, 51));

        jLabel20.setText("Elo");
```

```
        eloTxt.setEditable(false);

        denyBtn.setText("Deny");
        denyBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                denyBtnActionPerformed(evt);
            }
        });

        noRequestsLbl.setText("No Requests To Handle");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel5)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(firstNameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel6)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(surnameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel7)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(postcodeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel20)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
```

```
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(eloTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addContainerGap()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel1)
                                        .addGap(125, 125, 125)
                                        .addComponent(usernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel2)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                        .addComponent(emailAddressTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(36, 36, 36)
                                .addComponent(backBtn)))
                        .addGap(0, 0, Short.MAX_VALUE)))
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(87, 87, 87)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(passwordLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 70, Short.MAX_VALUE)
                            .addComponent(confirmPasswordLabel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(60, 60, 60)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(noRequestsLbl)
                            .addGroup(jPanel1Layout.createParallelGroup(
```

```
javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(firstNameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(surnameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(postcodeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(107, 107, 107)
                        .addComponent(denyBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 99,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel9)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(addressLineTwoTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addComponent(jLabel8)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addComponent(addressLineOneTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel10)
                                    .addComponent(jLabel11)
                                    .addComponent(jLabel12)
                                    .addComponent(jLabel14)
                                    .addComponent(jLabel13))
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING, false)
```

```
                                    .addComponent(discordNumberTxt)
                                    .addComponent(discordUsernameTxt)
                                    .addComponent(addressLineFourTxt)
                                    .addComponent(addressLineThreeTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 255, Short.MAX_VALUE)))
                                .addGroup(javax.swing.GroupLayout.Alignment.
TRAILING, jPanel1Layout.createSequentialGroup()
                                    .addComponent(jLabel15)
                                    .addGap(29, 29, 29)
                                    .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(dateOfBirthTxt,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                            .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, 64, Short.MAX_VALUE)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(addressOneLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressTwoLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressThreeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(addressFourLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(discordUsernameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(discordNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(dateOfBirthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(phoneNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addComponent(approveBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 99,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addContainerGap())
            );
            jPanel1Layout.setVerticalGroup(
                jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
```

```
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(60, 60, 60)
                                .addComponent(addressOneLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressTwoLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressThreeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(addressFourLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addGap(18, 18, 18)
                                .addComponent(discordUsernameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(26, 26, 26)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(jLabel5)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(confirmPasswordLab
el, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(19, 19, 19)
                                        .addComponent(firstNameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.UNRELATED)
                                        .addComponent(surnameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
```

```
                                            .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(jLabel1)
                                                .addComponent(usernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                            .addGap(18, 18, 18)
                                            .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(jLabel2)
                                                .addComponent(emailAddressTx
t, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                            .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                                                .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                    .addGroup(jPanel1Layout.
createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                                        .addGroup(jPanel1Lay
out.createSequentialGroup()
                                                            .addGap(18, 18,
18)
                                                            .addComponent(pa
sswordLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                                        .addGroup(jPanel1Lay
out.createSequentialGroup()
                                                            .addGap(24, 24,
24)
                                                            .addComponent(fi
rstNameTxt, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                                                    .addGap(27, 27, 27)
                                                    .addComponent(surnameTxt
, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                                .addGroup(jPanel1Layout.crea
teSequentialGroup()
                                                    .addPreferredGap(javax.s
wing.LayoutStyle.ComponentPlacement.RELATED)
                                                    .addComponent(jLabel6)))
                                            .addGap(30, 30, 30)
                                            .addGroup(jPanel1Layout.createPa
rallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                                .addComponent(postcodeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                                                .addComponent(jLabel7))
                                            .addPreferredGap(javax.swing.Lay
outStyle.ComponentPlacement.RELATED, 14, Short.MAX_VALUE)))))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(discordNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(postcodeLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(18, 18, 18)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addGroup(jPanel1Layout.createSequen
tialGroup()
                                        .addComponent(dateOfBirthLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                        .addGap(18, 18, 18)
                                        .addComponent(phoneNumberLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                    .addComponent(noRequestsLbl)))
                            .addGroup(jPanel1Layout.createParallelGroup(
javax.swing.GroupLayout.Alignment.BASELINE)
                                .addComponent(jLabel20)
                                .addComponent(eloTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(26, 26, 26)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel8)
                            .addComponent(addressLineOneTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel9)
                            .addComponent(addressLineTwoTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel10)
                            .addComponent(addressLineThreeTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel11)
                            .addComponent(addressLineFourTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel12)
                            .addComponent(discordUsernameTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel13)
                            .addComponent(discordNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel14)
                            .addComponent(dateOfBirthTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel15)
                            .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addGap(20, 20, 20)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(backBtn)
                    .addComponent(approveBtn)
```

```
                    .addComponent(denyBtn))
                .addContainerGap(39, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void
dateOfBirthTxtActionPerformed(java.awt.event.ActionEvent evt)
{

    }

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //goes back to the log in form
        WestCrossChessClubLeader leader = new
WestCrossChessClubLeader(session);
        leader.setVisible(true);this.dispose();
    }

    private void approveBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // approves the request to join the club
        try{
            ThreadClient.withdrawRequestToJoinClub(usernameTxt.getText()
);
            ThreadClient.addUserToClubAsMember(usernameTxt.getText());
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
        }
        //updates the form following the acceptance of the request
        clearForm();
        currentReqeustIndex++;
```

```
        checkIfThereAreMoreRequestsAndUpdateComponents();
        loadRequestToFormIfAppropraite();
    }

    private void usernameTxtKeyTyped(java.awt.event.KeyEvent evt)
{

    }

    private void usernameTxtFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void emailAddressTxtFocusLost(java.awt.event.FocusEvent evt)
{

    }

    private void denyBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // denies the request to join the club
        try{
            ThreadClient.withdrawRequestToJoinClub(usernameTxt.getText()
);
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
        }
        //updates the form after the request has been denied
        clearForm();
        currentReqeustIndex++;
        checkIfThereAreMoreRequestsAndUpdateComponents();
        loadRequestToFormIfAppropraite();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
```

```
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(RequestsToJoinClub.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(RequestsToJoinClub.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(RequestsToJoinClub.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(RequestsToJoinClub.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new RequestsToJoinClub().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel addressFourLabel;
    private javax.swing.JTextField addressLineFourTxt;
    private javax.swing.JTextField addressLineOneTxt;
    private javax.swing.JTextField addressLineThreeTxt;
    private javax.swing.JTextField addressLineTwoTxt;
    private javax.swing.JLabel addressOneLabel;
    private javax.swing.JLabel addressThreeLabel;
    private javax.swing.JLabel addressTwoLabel;
    private javax.swing.JButton approveBtn;
    private javax.swing.JButton backBtn;
    private javax.swing.JLabel confirmPasswordLabel;
    private javax.swing.JLabel dateOfBirthLabel;
    private javax.swing.JTextField dateOfBirthTxt;
    private javax.swing.JButton denyBtn;
    private javax.swing.JLabel discordNumberLabel;
    private javax.swing.JTextField discordNumberTxt;
    private javax.swing.JLabel discordUsernameLabel;
    private javax.swing.JTextField discordUsernameTxt;
    private javax.swing.JTextField eloTxt;
    private javax.swing.JTextField emailAddressTxt;
```

```
    private javax.swing.JLabel firstNameLabel;
    private javax.swing.JTextField firstNameTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JLabel noRequestsLbl;
    private javax.swing.JLabel passwordLabel;
    private javax.swing.JLabel phoneNumberLabel;
    private javax.swing.JTextField phoneNumberTxt;
    private javax.swing.JLabel postcodeLabel;
    private javax.swing.JTextField postcodeTxt;
    private javax.swing.JLabel surnameLabel;
    private javax.swing.JTextField surnameTxt;
    private javax.swing.JTextField usernameTxt;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

/**
 * This class is used to search through data
 * @author mortimer
 */
public class Search {
    public static final int INDEX_NOT_FOUND=-1;
    /**
     * Performs a binary search on the one d array. Assumes it is sorted
lexographically.
     * @param DATA The data sorted lexographically
     * @param ITEM_TO_FIND the data item to be found
     * @return true is returned if the record is in the array, otherwise
```

```
false is returned
     */
    public static boolean isRecordInArray(String[]DATA,String
ITEM_TO_FIND){
        if(DATA.length==1){
            if(DATA[0].equals(ITEM_TO_FIND)){
                return true;
            }else{
                return false;
            }
        }
        if(DATA.length==0){
            return false;
        }
        String record="";
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int comparison;
        String primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=DATA[mid];
            //if pk precedes itemToFind less than 0
            comparison = primaryKeyBuffer.compareTo(ITEM_TO_FIND);
            if(comparison==0){
                return true;
            }else if(comparison>0){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return false;
    }
    /**
     * Performs a binary search on the 2d array inputted. It searches
for the itemToFind. It assumes this is a primary key at the nought
element
     * of every row. It uses binary search and the java String method
.compareTo(). Hence the data must be sorted by the primary key
     * with data of a small lexographic value at the top (i.e. sorted in
ascending order).
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @param WIDTH_OF_DATA The width of a record
     * @return the field containing the item to find. If no item is
found, this will be a null array of width equal to the width of a
record.
     */
    public static
String[]searchForRecordWithPrimaryKeyAtNoughtElement(final
String[][]DATA,final String ITEM_TO_FIND,final int WIDTH_OF_DATA){
```

```
        String record[]= new String[WIDTH_OF_DATA];
        if(DATA.length==0){
            return record;
        }
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int comparison;
        String primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=DATA[mid][0];
            //if pk precedes itemToFind less than 0
            comparison = primaryKeyBuffer.compareTo(ITEM_TO_FIND);
            if(comparison==0){
                for(int element=0;element<record.length;element++){
                    record[element]=DATA[mid][element];
                }
                break;
            }else if(comparison>0){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return record;
    }
     /**
     * Performs a binary search on the 2d array inputted. It searches
for the itemToFind. It assumes this is a primary key at the nought
element
     * of every row. It uses binary search and assumes that the nought
element of every row is an integer. It asssumes that data is sorted by
integer
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @param WIDTH_OF_DATA The width of one record
     * @return the field containing the item to find. If no item is
found, this will be a null array of width equal to the width of a
record.
     */
    public static String[]
searchForRecordWithPrimaryKeyAtNoughtElementInt(final
String[][]DATA,final int ITEM_TO_FIND, final int WIDTH_OF_DATA){
        String record[]= new String[WIDTH_OF_DATA];
        if(DATA.length==0){
            return record;
        }
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int primaryKeyBuffer;
        do{
```

```
            mid = (first+last)/2;
            primaryKeyBuffer=Integer.parseInt(DATA[mid][0]);
            if(primaryKeyBuffer==ITEM_TO_FIND){
                for(int element=0;element<record.length;element++){
                    record[element]=DATA[mid][element];
                }
                break;
            }else if(primaryKeyBuffer>ITEM_TO_FIND){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return record;
    }
    /**
     * Performs a binary search on the 2d array inputted. It searches
for the itemToFind. It assumes this is a primary key at the nought
element
     * of every row. It uses binary search and assumes that the nought
element of every row is an integer. It assumes that data is sorted by
integer
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @return The row number is returned if not, -1 is returned
     */
    public static int searchForRecordIndexInt(final String[][]DATA,final
int ITEM_TO_FIND){
        if(DATA.length==0){
            return INDEX_NOT_FOUND;
        }
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=Integer.parseInt(DATA[mid][0]);
            if(primaryKeyBuffer==ITEM_TO_FIND){
                return mid;
            }else if(primaryKeyBuffer>ITEM_TO_FIND){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return INDEX_NOT_FOUND;
    }
    /**
     * Performs a binary search on the 2d array inputted. It searches
for the itemToFind. It assumes this is a primary key at the nought
element
     * of every row. It uses binary search and assumes that the data is
```

```
lexographically sorted.
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @return The row number is returned if not, -1 is returned
     */
    public static int searchForRecordIndex(final String[][]DATA,final
String ITEM_TO_FIND){
        if(DATA.length==0){
            return INDEX_NOT_FOUND;
        }
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int comparison;
        String primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=DATA[mid][0];
            //if pk precedes itemToFind less than 0
            comparison = primaryKeyBuffer.compareTo(ITEM_TO_FIND);
            if(comparison==0){
                return mid;
            }else if(comparison>0){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return INDEX_NOT_FOUND;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import philipm.cs5.software.development.server.system.UserInfo;

/**
 * This class is used to store various user data to create an object
that gives needed information to various system components on the
current user.
 * @author mortimer
 */
//stores session Information, such as username if appropraite
```

```
public class SessionInfo {
    private boolean isGuest;//stores whether the user is a guest
    private boolean isClubMember;//stores whether user is a member of
the club
    private boolean isClubLeader;//stores whether user is the club
leader
    private String username;//stores username
    /**
     * Creates a new SessionInfo Object with the appropriate user
details
     * @param username The username
     * @param isClubMember Whether the user is currently a club member
     * @param isClubLeader Whether the user is currently a club leader
     */
    public SessionInfo( String username, boolean isClubMember,boolean
isClubLeader){
        this.setIsClubLeader(isClubLeader);this.setIsClubMember(isClubMe
mber);this.setUsername(username);this.setIsGuest(false);
    }
    /**
     * Creates a SessionInfo object
     */
    private SessionInfo(){

    }
    /**
     * Creates a new SessionInfo object for guests.
     * @return The SessionInfo Object
     */
    public static SessionInfo createGuestSession(){
        SessionInfo sess=  new SessionInfo();
        sess.setIsGuest(true);sess.setUsername(UserInfo.GUEST_ID);sess.s
etIsClubMember(false);sess.setIsClubLeader(false);
        return sess;
    }
    /**
     * Sets whether the user is a guest
     * @param isGuest whether the user is a guest
     */
    private void setIsGuest(boolean isGuest){
        this.isGuest=isGuest;
    }
    /**
     * Sets the username of the user
     * @param username The username
     */
    private void setUsername(String username){
        this.username=username;
    }
    /**
     * Sets whether the user is a member of the chess club
     * @param isClubMember Whether the user is a member of the chess
club
```

```
     */
    public void setIsClubMember(boolean isClubMember){
        this.isClubMember=isClubMember;
    }
    /**
     * Sets whether the user is the club leader.
     * @param isClubLeader Whether the user is the club leader.
     */
    public void setIsClubLeader(boolean isClubLeader){
        this.isClubLeader=isClubLeader;
    }
    /**
     * Gets whether the current user is a guest user.
     * @return Whether the current user is a guest user.
     */
    public boolean getIsGuest(){
        return this.isGuest;
    }
    /**
     * Gets whether the current user is a member of the West Cross Chess
Club
     * @return Whether the current user is a member of the West Cross
Chess Club
     */
    public boolean getIsClubMember(){
        return this.isClubMember;
    }
    /**
     * Gets whether the current is user is the leader of the West Cross
Chess Club
     * @return Whether the current is user is the leader of the West
Cross Chess Club
     */
    public boolean getIsClubLeader(){
        return this.isClubLeader;
    }
    /**
     * Gets the username of the current user
     * @return The username
     */
    public String getUsername(){
        return this.username;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

/**
 * This class is used to sort data
 * @author mortimer
 */
public class Sort {
    /**
    * Sorts data using insertion sort
    * @param unsortedData a 2d string array of the unsorted data with
each row representing one record and the nought element containing a
String primary key (which is the field that will be used to sort)
    */
    public static void insertionSort(String[][]unsortedData){
        int indexToBeChecked=0;//loop variable that needs intialisation
for method wide access
        String tempRecord[]=new String[unsortedData[0].length];//used as
buffer storage of a record
        for(int
indexOfElementToBeInserted=1;indexOfElementToBeInserted<unsortedData.len
gth;indexOfElementToBeInserted++){
            for(int
recordIndex=0;recordIndex<tempRecord.length;recordIndex++){//stores
value of record to be "inserted"
                tempRecord[recordIndex]=unsortedData[indexOfElementToBeI
nserted][recordIndex];
            }
            for(indexToBeChecked=indexOfElementToBeInserted-
1;indexToBeChecked>=0&&(tempRecord[0].compareTo(unsortedData[indexToBeCh
ecked][0])<0);indexToBeChecked--){
                for(int
recordIndex=0;recordIndex<tempRecord.length;recordIndex++){
                    unsortedData[indexToBeChecked+1][recordIndex]=unsort
edData[indexToBeChecked][recordIndex];
                }
            }
            for(int
recordIndex=0;recordIndex<tempRecord.length;recordIndex++){
                unsortedData[indexToBeChecked+1][recordIndex]=tempRecord
[recordIndex];
            }
        }
    }
        /**
     * This function recursively splits all record down into sub arrays
until each sub array only contains one record
     * these split arrays are merged back together using the merge
function.
     * Therefore the data originally input into this function will be
sorted (as arrays are passed by reference in java). The return value of
this function is also the
```

```
     * sorted data.
     * @param data The data you want sorted. Assuming each row is one
record and the first element in each row is a string that is the primary
key.
     * @return The sorted data is returned with the data being sorted in
ascending order (one row representing each record and the first element
of each row containing the integer primary key).
     */
    public static String[][] mergeSort(String[][]data){
        if(data.length==1){
            return data;
        }
        int middle = data.length/2; //calculates the middle of the
current data
        String[][]left=new String[middle][data[0].length];//stores the
left half of the current data
        String[][]right = new String[data.length-
middle][data[0].length];//stores the right half of the current data
        for(int yVal=0;yVal<data.length;yVal++){//fills up the left and
right arrays so that the left array stores the left/ top half of the
data array and the right array stores the remianing records
            if(yVal>=middle){
                for(int xVal=0;xVal<data[0].length;xVal++){
                    right[yVal-middle][xVal]=data[yVal][xVal];
                }
            }else{
                for(int xVal=0;xVal<data[0].length;xVal++){
                    left[yVal][xVal]=data[yVal][xVal];
                }
            }
        }
        return merge(mergeSort(left),mergeSort(right),data);//returns
sorted data. calls the merge function on the data that has been
recursively split into single elements
    }
    /**
     *
     * @param left the left hand sub array of data
     * @param right the right hand sub array of data
     * @param dataToBeSorted the data to be sorted
     * @return returns the merged / sorted data
     */
    private static String[][]
merge(String[][]left,String[][]right,String dataToBeSorted[][]){
        int leftIndex=0;//stores index for left array
        int rightIndex=0;//stores index for right array
        int dataToBeStoredIndex=0;//stores the row index for the data to
be sorted array
        while(leftIndex<left.length && rightIndex<right.length){//both
righ and left arrays are already sorted. sorts one entire sub array and
most of the other sub array
            if(left[leftIndex][0].compareTo(right[rightIndex][0])<0){
                for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
```

```
                    dataToBeSorted[dataToBeStoredIndex][xVal]=left[leftI
ndex][xVal];
                }
                dataToBeStoredIndex++;
                leftIndex++;
            }else{
                for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                    dataToBeSorted[dataToBeStoredIndex][xVal]=right[righ
tIndex][xVal];
                }
                dataToBeStoredIndex++;
                rightIndex++;
            }
        }
        //as one of the sub arrays has been completely sorted, the
remaining elements of the other array (which are already sorted) are
appended onto the end of the array
        while(leftIndex<left.length){
            for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                dataToBeSorted[dataToBeStoredIndex][xVal]=left[leftIndex
][xVal];
            }
            dataToBeStoredIndex++;
            leftIndex++;
        }
        while(rightIndex<right.length){
            for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                dataToBeSorted[dataToBeStoredIndex][xVal]=right[rightInd
ex][xVal];
            }
            dataToBeStoredIndex++;
            rightIndex++;
        }
        return dataToBeSorted;
    }
    /**
     * Sorts the array using quicksort. The nought element of every row
in the array must be an integer
     * @param arr The array to sort
     */
    public static void quickSortUsingIntAtFirstElement( String[][]arr){
        quickSortInt(0, arr.length-1, arr);
    }
    /**
     * Performs a quick sort assuming that each record in the array
contains an integer value at the nought element.
     * @param first A parameter used during quicksort
     * @param last A parameter used during quicksort
     * @param arr The array to sort
     */
    private static void quickSortInt(int first, int last,String
arr[][]){
        int left, right, pivot;
```

```
        String buffer[]=new String[arr[0].length];
        left =first; right=last;
        pivot = Integer.parseInt(arr[(left+right)/2][0]);
        while(left<=right){
            while(Integer.parseInt(arr[left][0])<pivot){
                left++;
            }
            while(Integer.parseInt(arr[right][0])>pivot){
                right--;
            }
            if(left<=right){
                for(int x=0;x<arr[0].length;x++){//loops through all
fields in record to copy array
                    buffer[x]=arr[right][x];
                    arr[right][x]=arr[left][x];
                    arr[left][x]=buffer[x];
                }
                left++;
                right--;
            }
        }
        if(first<right){
            quickSortInt(first, left-1,arr);
        }
        if(left<last){
            quickSortInt(right+1, last,arr);
        }
    }
    /**
     * performs a merge sort on the array provided
     * @param data The array to sort
     * @return The sorted data
     */
    public static int[][] mergeSortInt(int[][]data){
        if(data.length==1 || data.length==0){
            return data;
        }
        int middle = data.length/2; //calculates the middle of the
current data
        int[][]left=new int[middle][data[0].length];//stores the left
half of the current data
        int[][]right = new int[data.length-
middle][data[0].length];//stores the right half of the current data
        for(int yVal=0;yVal<data.length;yVal++){//fills up the left and
right arrays so that the left array stores the left/ top half of the
data array and the right array stores the remianing records
            if(yVal>=middle){
                for(int xVal=0;xVal<data[0].length;xVal++){
                    right[yVal-middle][xVal]=data[yVal][xVal];
                }
            }else{
                for(int xVal=0;xVal<data[0].length;xVal++){
                    left[yVal][xVal]=data[yVal][xVal];
```

```
                }
            }
        }
        return
mergeInt(mergeSortInt(left),mergeSortInt(right),data);//returns sorted
data. calls the merge function on the data that has been recursively
split into single elements
    }
    /**
     *
     * @param left the left hand sub array of data
     * @param right the right hand sub array of data
     * @param dataToBeSorted the data to be sorted
     * @return returns the merged / sorted data
     */
    private static int[][] mergeInt(int[][]left,int[][]right,int
dataToBeSorted[][]){
        int leftIndex=0;//stores index for left array
        int rightIndex=0;//stores index for right array
        int dataToBeStoredIndex=0;//stores the row index for the data to
be sorted array
        while(leftIndex<left.length && rightIndex<right.length){//both
righ and left arrays are already sorted. sorts one entire sub array and
most of the other sub array
            if(left[leftIndex][0]<right[rightIndex][0]){
                for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                    dataToBeSorted[dataToBeStoredIndex][xVal]=left[leftI
ndex][xVal];
                }
                dataToBeStoredIndex++;
                leftIndex++;
            }else{
                for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                    dataToBeSorted[dataToBeStoredIndex][xVal]=right[righ
tIndex][xVal];
                }
                dataToBeStoredIndex++;
                rightIndex++;
            }
        }
        //as one of the sub arrays has been completely sorted, the
remaining elements of the other array (which are already sorted) are
appended onto the end of the array
        while(leftIndex<left.length){
            for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                dataToBeSorted[dataToBeStoredIndex][xVal]=left[leftIndex
][xVal];
            }
            dataToBeStoredIndex++;
            leftIndex++;
        }
        while(rightIndex<right.length){
            for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
```

```
                dataToBeSorted[dataToBeStoredIndex][xVal]=right[rightInd
ex][xVal];
            }
            dataToBeStoredIndex++;
            rightIndex++;
        }
        return dataToBeSorted;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

/**
 * This class enables the user to make use of stockfish when playing
offline. (Stockfish is a powerful chess engine)
 * @author mortimer
 */
public class StockFish {
    private BufferedReader input;//used to input to the engine
    private BufferedWriter output;//used to read the output of the
engine
    private Process process;//stores the process (i.e. the execution of
the stockfish engine)
    private static final String
STOCK_FISH_LOCATION="stockfish_20090216_x64.exe";//stores the name of
the stockfish location
    /**
     * Creates a new stock fish object that can run Stockfish 12
     * @throws java.io.IOException An error may occur when loading
Stockfish
     */
    public StockFish()throws IOException{
        try{
            process=Runtime.getRuntime().exec(STOCK_FISH_LOCATION);//cre
ates the execution process
            //gets the input and output streams
            input = new BufferedReader(new
```

```
InputStreamReader(process.getInputStream()));
            output = new BufferedWriter(new
OutputStreamWriter(process.getOutputStream()));
        }catch(IOException e){
            throw e;//throws any erros that may occur
        }
    }
    /**
     * Gets the Stockfish 12's move from the given board state.
     * @param currentFen The FEN of the current board state
     * @return The board state after the computer has made its move
     * @throws IOException An exception may occur when attempting to
execute the Stockfish file.
     */
    public String getStockFishMove(String currentFen) throws
IOException{
        try{
            //sets board state to appropriate FEN
            executeTask("position fen "+currentFen);
            //best move is calculated
            executeTask("go movetime 1000");//this gets the best move
with a search time of 1000 milliseconds
            String bestMove= readOutput("bestmove ").split("
")[1];//gets the computer move
            executeTask("position fen "+currentFen+" moves
"+bestMove);//updates the board
            executeTask("d");//gets the computer to draw the current
board and output the corresponding FEN
            String lineWithFen=readOutput("Fen: ");//gets the line
containing the current Fen String
            String fen="";//stores the Fen to return
            for(int c=5;c<lineWithFen.length();c++){//loops through
every character in the FEN and appends it to the String storing the FEN
board
                fen = fen+String.valueOf(lineWithFen.charAt(c));
            }
            return fen;
        }catch(IOException e){
            throw e;
        }
    }
    /**
     * Executes the desired instruction
     * @param task The UCI command to execute
     * @throws java.io.IOException An exception may occur as Stockfish
is an executable file.
     */
    private void executeTask(String task) throws IOException{
        try{
            output.write(task+System.lineSeparator());
            output.flush();//writes the command and flushes the output
        }catch(IOException e){
            throw e;
```

```
        }
    }
    /**
     * Loops through all new lines and checks to see if they begin with
the expectedValue. If they do, the line is returned
     * @param expectedValue The expected start of the line
     * @return The line read
     * @throws IOException An exception may occur as Stockfish is an
executable file.
     */
    private String readOutput(String expectedValue)throws IOException{
        try{
            String lineRead;//stores the line read
            while((lineRead=input.readLine())!=null){
                if(lineRead.startsWith(expectedValue)){
                    return lineRead;
                }
            }
        }catch(IOException e){
            throw e;
        }
        return null;
    }
    /**
     * Exits the application and closes the Stockfish application. Once
this process is called, no further calls can be made to the current
StockFish instance
     * @throws IOException An exception may occur when attempting to
exit Stockfish.
     */
    public void exit()throws IOException{
        try{
            executeTask("quit");//attempts to quit the program
        }finally{
            process.destroy();//closes the task
            input.close();
            output.close();
        }
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;
```

```
import java.awt.Color;
import java.awt.Component;
import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;

/**
 * This creates a custom table renderer object to render the online
games table.
 * @author mortimer
 */
public class TableRenderer extends DefaultTableCellRenderer{
    private boolean rowOneHighlighted=false;//determines the colour of
row one
    private boolean rowTwoHighlighted=false;//determines the colour of
row two
    private boolean rowThreeHighlighted=false;//determines the colour of
row three
    private static final Color HIGHLIGHT_COLOR=Color.WHITE;//stores the
color used for highlighted rows
    private static final Color JLABEL_COLOUR=Color.LIGHT_GRAY;//stores
the default JLabel rows.
    private final OnlineMenu ONLINE;//stores the online menu in which
the table is locared
    /**
     * Creates a custom table renderer to render the online game table.
     * @param online The online menu form in which the online table is
in.
     */
    public TableRenderer(OnlineMenu online){
        ONLINE=online;//stores the online menu form
    }
    /**
     * Updates the variables that change the table row colour
     * @param rowOneIsHighlighted Whether row one should be highlighted
     * @param rowTwoIsHighlighted Whether row two should be highlighted
     * @param rowThreeIsHighlighted Whether row three should be
highlighted
     */
    public void updateRowColour(boolean rowOneIsHighlighted,boolean
rowTwoIsHighlighted,boolean rowThreeIsHighlighted){
        rowOneHighlighted=rowOneIsHighlighted;rowTwoHighlighted=rowTwoIs
Highlighted;rowThreeHighlighted=rowThreeIsHighlighted;//sets object
variables equal to the passed variables
    }
    /**
     * Returns the customised component to be rendered
     * @param table The table
     * @param value The value of the component
     * @param isSelected Whether the component is selected
     * @param hasFocus Whether the component is focused
     * @param row The row of the component
```

```
     * @param col The column of the table
     * @return The component
     */
    @Override
    public Component getTableCellRendererComponent(JTable table, Object
value, boolean isSelected, boolean hasFocus, int row, int col) {
        JLabel label = (JLabel)
super.getTableCellRendererComponent(table, value, isSelected, hasFocus,
row, col);//gets JLabel for the row + col chosen
        //Get the status for the current row and sets the colour of the
JLabel
        Color cellColour = JLABEL_COLOUR;
        switch (row) {
            case 0:
                if (rowOneHighlighted) {
                    cellColour = HIGHLIGHT_COLOR;
                }
                break;
            case 1:
                if (rowTwoHighlighted) {
                    cellColour = HIGHLIGHT_COLOR;
                }
                break;
            default:
                if (rowThreeHighlighted) {
                    cellColour = HIGHLIGHT_COLOR;
                }
                break;
        }
        if (isSelected) {
            if (cellColour == JLABEL_COLOUR) {
                ONLINE.setPlayButtonEnabled(false);
                ONLINE.setDeleteButtonEnabled(!table.getValueAt(row,
0).equals(""));
            } else {
                ONLINE.setPlayButtonEnabled(true);
                ONLINE.setDeleteButtonEnabled(false);
                cellColour = table.getSelectionBackground();
            }
        }
        label.setBackground(cellColour);//sets colour to chosen colour
        return label;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
```

```
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.IOException;
import philipm.cs5.software.development.server.system.UserInfo;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.Socket;

/**
 * This objects consists of a set of static methods which send (and
sometimes receive) data from / to the server.
 * @author mortimer
 */
public class ThreadClient {
    /**
     * Tries to log user in with provided details
     * @param userName The username of the user
     * @param password The users password
     * @return The user information
     * @throws Exception If an error occurs, such as failing to connect
to the server, an exception is thrown
     */
    public static UserInfo logInWithUserName(String userName, String
password) throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//creates a socket to connect
with the server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets the object input
stream
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());//gets the object output
stream
        InputPackages pack = new
InputPackages();pack.createLogInRequestForWithUserNameAndPassword(userNa
me, password);//creates the package to send to the server
        out.writeObject(pack);//sends package to server
        return (UserInfo)in.readObject();//gets the UserInfo object from
the server and returns it
    }
    /**
     * Tries to log user in with email address.
    * @param emailAddress The user's email address
     * @param password The user's password
     * @return The user information for the user
     * @throws Exception If a networking error occurs, an exception may
be thrown
     */
```

```
    public static UserInfo logInWithEmailAddress(String emailAddress,
String password) throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//creates socket that connects
to the server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());//gets output stream
        InputPackages pack = new
InputPackages();pack.createLogInRequestForWithEmailAndPassword(emailAddr
ess, password);//creates package to send to sever
        out.writeObject(pack);//sends package to server
        UserInfo inf=(UserInfo)in.readObject();
        return inf;//gets the UserInfo object from the server and
returns it
    }
    /**
     * Checks to see if the provided username is already stored in the
user database
     * @param username The username to be fetched
     * @return Returns true if the username if already taken and false
otherwise
     * @throws Exception An exception may be thrown due to the online
connection required to execute this method.
     */
    public static boolean isUserNameAlreadyTaken(String username)throws
Exception{
        try{
            return
logInWithUserName(username,"passw").getUserFound();//attempts to log in
with the chosen username and an arbitary password
        }catch(Exception e){
            throw e;
        }
    }
    /**
     * Checks to see if the provided email address is already stored in
the user database
     * @param emailAddress The email address to be fetched
     * @return Returns true if the email address if already taken and
false otherwise
     * @throws Exception An exception may be thrown due to the online
connection required to execute this method.
     */
    public static boolean isEmailAddressAlreadyTaken(String
emailAddress)throws Exception{
        try{
            return logInWithEmailAddress(emailAddress,
"passw").getEmailAddressFound();//attempts to log in with chosen email
address and an arbitary password
        }catch(Exception e){
```

```
            throw e;
        }
    }
    /**
     * Communicates with the server to create the new user account.
     * @param accountDetails The array containing the account record to
be written to the file
     * @param emailAddress The email address of the account
     * @throws Exception An exception may occur due to network issues.
     */
    public static void createAccount(String[]accountDetails,String
emailAddress)throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//creates socket that
communicates with server
        InputPackages pack = new
InputPackages();pack.createAccountPacket(accountDetails,
accountDetails[0], emailAddress);//creates package to send to server
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());//gets output stream
        out.writeObject(pack);//writes package to server
    }
    /**
     * Gets data for online games for a specific user
     * @param userName The username of the user
     * @return The game data
     * @throws java.lang.Exception If a networking error occurs, then an
error is thrown
     */
    public static String[][]getOnlineGameData(String userName) throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.createRequestForAllGameDataForAUser(userName);
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//creates the object input
stream
        return (String[][])in.readObject();
    }
    public static void createNewRandomOnlineGame(String userName)throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
```

```
        pack.createRandomGamePackage(userName);//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
    }
    /**
     * Communicates with server to delete the specified game
     * @param gameId The game id of the game to delete
     * @param gameResult The game result. 1 indicates a win for player
one. 0.5 indicates a draw. 0 indicates a win for player two
     * @throws java.lang.Exception An exception may occur when
communicating with the server
     */
    public static void deleteOnlineGame(int gameId,double
gameResult)throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.createRequestToDeleteGame(String.valueOf(gameId),gameResult
);//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
    }
    public static void updateOnlineGameRecord(String[]newRecord)throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.createRequestToUpdateOnlineGame(newRecord);//creates server
request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    public static String createGameAgainstFriend(String
playerOneUserName,String playerTwoName,int timeLeftForBoth)throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.createNewOnlineGameWithFriend(playerOneUserName,
playerTwoName, String.valueOf(timeLeftForBoth));//creates server request
        ObjectOutputStream out = new
```

```
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String)in.readObject();
    }
    /**
     * Tries to join a game using a friend code. If the game is found
and has been joined, true is returned. Otherwise, false is returned
     * @param code the game code
     * @param userName The username of the player attempting to join the
game
     * @return Whether the game has been joined
     * @throws java.lang.Exception An error when communicating with the
server
     */
    public static boolean joinGameUsingFriendCode(int code,String
userName)throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.joinGameUsingGameCode(code,userName);//creates server
request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (boolean)in.readObject();
    }
    /**
     * Gets the game data stored in the online game file
     * @param gameCode The game id
     * @return The record - if the record is not found, the array will
all be null
     * @throws java.lang.Exception An error when communicating with the
server
     */
    public static String[] getGameData(int gameCode) throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getGameRecord(gameCode);//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[])in.readObject();
    }
    /**
```

```
     * Gets the contact details of the West Cross chess club.
     * @return The contact details. Element 0 is the phone number and
element 1 is the email address to contact.
     * @throws java.lang.Exception An exception may occur when
communicating with the server
     */
    public static String[]getChessClubContactDetails() throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.requestClubContactInfo();//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[])in.readObject();
    }
    /**
     * Checks to see whether the user has requested to join the club
     * @param userName The username
     * @return True if the user has and false otherwise
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static boolean getHasUserRequestedToJoinChessClub(String
userName) throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getWhetherUserHasRequestedToJoinClub(userName);//creates
server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return in.readBoolean();
    }
    /**
     * Alters the status of a user's request to join the club. I.e., if
they have requested to join the club, this request is withdrawn and vice
versa.
     * @param userName The username of the user
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static void changeRequestToJoinClubStatus(String
userName)throws Exception{
```

```
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.changeRequestToJoinClubStatus(userName);//creates server
request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Removes user from the chess club.
     * @param userName The username of the user
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static void removeUserFromChessClub(String userName)throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.removeUserFromChessClub(userName);//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Gets the league table
     * @return The table
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static String[][]getLeagueTable()throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getLeagueTableRequest();//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[][])in.readObject();
    }
    /**
     * Gets the all members of the chess club
```

```
     * @return The members
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static String[]getMembersOfChessClub()throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getAllClubMembers();//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[])in.readObject();
    }
    /**
     * Gets the all fixtures and results of current season. The array
will be of length 0 if an error occurred (e.g. no fixtures can be
displayed)
     * @return The fixtures and results.
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static String[][][]getFixturesAndResults()throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getFixturesAndResults();//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[][][])in.readObject();
    }
    /**
     * Gets the all fixtures and results of current season. The array
will be of length 0 if an error occurred (e.g. no fixtures can be
displayed)
     * @return The fixtures and results.
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static
String[]getListOfPeopleAddedMidwayThroughSeason()throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
```

```
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getPeopleAddedMidwayThroughSeason();//creates server
request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[])in.readObject();
    }
    /**
     * Updates the server stored fixture's and results
     * @param newFixturesAndResults The updated fixture and results
     * @throws java.io.IOException An exception may occur when
communicating with the server
     */
    public static void
updateFixturesAndResults(String[][][]newFixturesAndResults)throws
IOException{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.updateFixturesAndResults(newFixturesAndResults);//creates
server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Starts a new season
     * @throws java.io.IOException An exception may occur when
communicating with the server
     */
    public static void createNewSeason()throws IOException{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.startNewSeason();//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Updates the chess club contact details
     * @param phoneNumber The chess club phone number
     * @param emailAddress The email address of the club
```

```
     * @throws IOException An exception may occur when communicating
with the server
     */
    public static void setChessClubContactDetails(String
phoneNumber,String emailAddress) throws IOException{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.setContactDetails(phoneNumber,emailAddress);//creates
server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Sets the inputted user as both a club member and the club leader
     * @param userName The user name
     * @throws IOException An exception may occur when communicating
with the server
     */
    public static void setUserAsClubLeader(String userName) throws
IOException{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.setClubLeader(userName);//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Returns all requests to join the west cross chess club.
     * @return The requests. Each row is one record from the user
details column
     * @throws java.io.IOException An exception may occur when
communicating with the server
     */
    public static String[][]getAllRequestsToJoinClub() throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getAllRequestsToJoinClub();//creates server request
        ObjectOutputStream out = new
```

```
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[][])in.readObject();
    }
    /**
     * Returns the email address of a chosen user
     * @param userName The username
     * @return The email address
     * @throws java.lang.Exception An exception may occur when
communicating with the server
     */
    public static String getEmailAddressOfUser(String userName)throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getEmailAddressOfUser(userName);//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return ((String[])in.readObject())[0];
    }
    /**
     * Makes it so that the user chosen has not requested to join the
chess club.
     * @param userName The user name
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static void withdrawRequestToJoinClub(String userName) throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.withdrawRequestToJoinClub(userName);//creates server
request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Adds the specified user to the west cross chess club
     * @param userName The user
     * @throws Exception An exception may occur when communicating with
the server
     */
```

```
    public static void addUserToClubAsMember(String userName)throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.addUserToClubAsMember(userName);//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Gets the user info of all members who are part of the club
     * @return The info
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static String[][] getAllClubMemberUserInfo()throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getAllClubMembersInfo();//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[][])in.readObject();
    }
    /**
     * Updates the user record of the user who's user name is stored at
element 0 of the inputted array.
     * @param userRecord The new record
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static void updateUserInfo(String[]userRecord)throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.updateUserRecord(userRecord);//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
```

```
    /**
     * Deletes the data of a non chess club member from the server
     * @param userName The username
     * @throws java.lang.Exception An exception may occur when
communicating with the server
     */
    public static void deleteUserInfoOfNonMemberUserFromServer(String
userName)throws Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.removeNonMemberUserFromRecord(userName);//creates server
request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the
server
    }
    /**
     * Gets all non chess club members
     * @return The users
     * @throws Exception An exception may occur when communicating with
the server
     */
    public static String[][]getAllUsersWhoAreNotMembers() throws
Exception{
        Socket socket = new
Socket(InetAddress.getByName(ChangeInternetSettings.getServerAddress()),
ChangeInternetSettings.getPortNumber());//connects to server
        ObjectInputStream in =new
ObjectInputStream(socket.getInputStream());//gets input stream
        InputPackages pack = new InputPackages();//creates a new input
package
        pack.getAllNonMembers();//creates server request
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeObject(pack);//writes the package to the server
        return (String[][])in.readObject();
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package philipm.cs5.software.development.user.system;

import philipm.cs5.software.development.server.system.UserInfo;

/**
 * This class performs various validation routines
 * @author mortimer
 */
public class Validation {
    private static final
String[]ILLEGAL_CHAR_SEQUENCES={UserInfo.SPLIT_ITEM,OfflinePlay.SPLIT_FE
N_STRING, UserInfo.GUEST_ID,UserInfo.GUEST_ID_EMAIL};//stores all
illegeal characters sequences
    /**
     * Checks to see whether input string is empty or not. Returns false
if length of String is 0 or a NullPointerException occurs (e.g. for
newly intialised String arrays java).
     * Otherwise true is returned. Note: this presence check views a
String with a (or any) single space(s) in it as
     * a non-empty input and returns true for it.
     * @param stringToCheck the string to be checked
     * @return returns false if the String is empty (i.e. has a length
of 0, using the .length() command or triggers a NullPointerException
caused by String arrays that have been initialised but have no value
assigned to them) and true if the string is not empty
     */
    public static boolean isPresent(String stringToCheck){//presence
check
        try{
            if(stringToCheck.length()==0){//checks to see if number of
characters are zero e.g.:""
                return false;
            }else{
                return true;
            }
        }catch(java.lang.NullPointerException e){//nullpointer exception
triggered for newly initialised elements in a string array
            return false;
        }
    }
    /**
     * Checks to see if String is of desired length. If String is less
than or equal to desired length, true is returned. Otherwise, false is
returned.
     * @param maxLengthAllowed the maximum length of the string
     * @param stringToCheck the string to perform the length check on
     * @return returns true if the length of the string is less than or
equal to the max length allowed and false if not.
     */
    public static boolean isAppropriateLength(int
maxLengthAllowed,String stringToCheck){//length check
        return !(stringToCheck.length()>maxLengthAllowed);
    }
```

```
    /**
     * A simple format check to see whether a string is in the form
dd/mm/yyyy.It checks to see that there are integers and dashes in the
correct places.
     * It also checks to see if the month field is greater than twelve
and if the day field is greater than 31 (although invalid inputs can
occur, for example if the day is 31 and the month is february. If a
valid date is not entered, false is returned.
     * @param stringToCheck the string to check
     * @return true if the inputted string is considered a valid date in
the form dd/mm/yyyy.
     */
    public static boolean formatCheckDateOfBirth(String
stringToCheck){//format check
        return((stringToCheck.matches("([0-9]{2})/([0-9]{2})/([0-
9]{4})"))&&(Integer.parseInt(String.valueOf(stringToCheck.charAt(3))+Str
ing.valueOf(stringToCheck.charAt(4)))<=12)&&(Integer.parseInt(String.val
ueOf(stringToCheck.charAt(0))+String.valueOf(stringToCheck.charAt(1)))<=
31));
    }
    /**
     * Tries converting the input string into an integer using the
Integer.parseInt(). If a NumberFormatException is thrown, it is not an
integer and false is returned, else true is returned.
     * @param stringToCheck the string to check
     * @return true is returned if the string is an integer, false is
returned if the string is not an integer
     */
    public static boolean isInteger(String stringToCheck){//type check
        try{
            int val = Integer.parseInt(stringToCheck);
            return true;
        }catch(NumberFormatException e){
            return false;
        }
    }
    /**
     * Checks to see if the integer to check is with in the range
specified. True is returned if in the range, else false
     * @param lowerLimitInclusice this is the lower integer limit for
range. Checks to see if integerToCheck is greater than OR equal to this
number
     * @param upperLimitInclusive this is the upper integer limit for
range. Checks to see if integerToCheck is less than OR equal to this
number
     * @param integerToCheck the integer number to see if it is with in
the specified range
     * @return True is returned if integerToCheck is greater than or
equal to lowerLimitInclusive AND if integerToCheck is less than or equal
to upperLimitInclusive. Otherwise, false is returned
     */
    public static boolean rangeCheck(int lowerLimitInclusice,int
upperLimitInclusive,int integerToCheck){
```

```
        return (integerToCheck>=lowerLimitInclusice &&
integerToCheck<=upperLimitInclusive);
    }
    /**
     * Performs a linear search through all acceptable inputs comparing
it to the String input. If a match is found, true is returned, else
false is returned.
     * @param acceptableInputs A String array containing all acceptable
inputs
     * @param stringToCheck The String to check against the acceptable
inputs
     * @return True is returned if stringToCheck matches any item in the
array, otherwise false is returned.
     */
    public static boolean lookUpCheck(String[]acceptableInputs,String
stringToCheck){//lookup check
        for(int
indexInAcceptableInputs=0;indexInAcceptableInputs<acceptableInputs.lengt
h;indexInAcceptableInputs++){
            if(stringToCheck.equals(acceptableInputs[indexInAcceptableIn
puts])){
                return true;
            }
        }
        return false;
    }
    /**
     * Checks to see if the length of the string is equal to the desired
length.
     * @param correctLength The desired length of the string
     * @param stringToCheck the string to have its length checked
     * @return True is returned if the length of the string equals
correctLength, else false.
     */
    public static boolean isCorrectLength(int correctLength,String
stringToCheck){
        return (stringToCheck.length()==correctLength);
    }
    /**
     * Checks whether an integer is greater than or equal to another
integer
     * @param numberThatIntegerShouldBeGreaterThanOrEqual The value that
numberToCheck should be greater than or equal to in order to return true
     * @param numberToCheck The number that is being checked to see if
it is greater than or equal to
numberThatIntegerShouldBeGreaterThanOrEqual
     * @return True is returned if numberToCheck is greater than or
equal to numberThatIntegerShouldBeGreaterThanOrEqual, otherwise false is
returned
     */
    public static boolean isgreaterThanOrEqual(int
numberThatIntegerShouldBeGreaterThanOrEqual,int numberToCheck){
        return(numberToCheck>=numberThatIntegerShouldBeGreaterThanOrEqua
```

```
l);
    }
    /**
     * Checks the string to see if it contains any of strings that are
not allowed. For example, if commas are used to separate items, commas
may be banned.
     * @param itemToCheck The string to check
     * @return true is returned if it contains any illegal characters /
strings. Otherwise, false is returned.
     */
    public static boolean containsIllegalStrings(String itemToCheck){
        for(int i=0;i<ILLEGAL_CHAR_SEQUENCES.length;i++){
            if(itemToCheck.contains(ILLEGAL_CHAR_SEQUENCES[i])){
                return true;
            }
        }
        return false;
    }
    /**
     * Returns all illegal character sequences as a string
     * @return The illegal characters sequence as a presentable string
     */
    public static String getListOfIllegalStrings(){
        String ill="";
        for(int i=0;i<ILLEGAL_CHAR_SEQUENCES.length;i++){
            ill=ill+ILLEGAL_CHAR_SEQUENCES[i];
        }
        return ill;
    }
    /**
     * Checks to see whether the string only contains characters from
the English alphabet.
     * @param stringToCheck The string to be checked
     * @return True is returned if the string consists only of letters
in the English alphabet. Otherwise, false is returned.
     */
    public static boolean consistsOnlyOfLetters(String stringToCheck){
        return stringToCheck.matches("[a-zA-Z]+");
    }
    /**
     * Checks to see if the postcode is valid.
     * @param postCode The postcode as a string
     * @return True is returned if the postcode is between 6 and 8
characters, contains exactly one space and all characters other than a
space character are capitalised.
     */
    public static boolean isValidPostcode(String postCode){
        if(postCode.length()>8 || postCode.length()<6){//length checks
            return false;
        }
        int noOfSpaces=0;//counts number of spaces
        for(int c=0;c<postCode.length();c++){
            if(postCode.charAt(c)==' '){
```

```
                if(noOfSpaces==1){
                    return false;
                }
                noOfSpaces++;
            }else{
                if(!Character.isUpperCase(postCode.charAt(c)) &&
Character.isDigit(postCode.charAt(c))==false){
                    return false;
                }
            }
        }
        if(noOfSpaces==0){
            return false;
        }
        return true;
    }
    /**
     * Checks if the string only contains +, spaces and numbers.
     * @param phoneNumber The phone number
     * @return True is returned if the string consists only of plusses,
spaces and digits. Otehrwise, false is returned.
     */
    public static boolean onlyContainsPhoneNumberCharacters(String
phoneNumber){
        for(int c=0;c<phoneNumber.length();c++){
            if(phoneNumber.charAt(c)!=' '&&
phoneNumber.charAt(c)!='+'&&!Character.isDigit(phoneNumber.charAt(c))){
                return false;
            }
        }
        return true;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import javax.swing.JOptionPane;

/**
 * Creates a new form that allows guests to access features of the chess
club available to guests.
 * @author mortimer
```

```
 */
public class WestCrossChessClubGuest extends javax.swing.JFrame {
    private boolean error=false;//stores whether an error has occurred
    private SessionInfo session;//stores the session data
    /**
     * Creates new form WestCrossChessClubGuest
     */
    private WestCrossChessClubGuest() {
        initComponents();
    }
    public WestCrossChessClubGuest(SessionInfo session){
        this.session=session;
        initComponents();
        attemptToLoadContactInfo();//attempts to load the contact
information to the form
    }
    /**
     *Loads contact details onto the appropriate forms
     */
    private void attemptToLoadContactInfo(){
        try{
            String[]contactInfo=ThreadClient.getChessClubContactDetails(
);//gets the contact information from the server
            phoneNumberTxt.setText(contactInfo[0]);emailTxt.setText(cont
actInfo[1]);//loads contact information to form
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
attempting to get the club contact information from the server. Error:
"+e,"Error",JOptionPane.OK_OPTION);
            error=true;
            Menu menu=new Menu(session);
            menu.setVisible(true);
            this.dispose();return;
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        backBtn = new javax.swing.JButton();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        phoneNumberTxt = new javax.swing.JTextField();
```

```
        jLabel16 = new javax.swing.JLabel();
        emailTxt = new javax.swing.JTextField();
        jLabel17 = new javax.swing.JLabel();
        jLabel18 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("West Cross Chess Club");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowOpened(java.awt.event.WindowEvent evt) {
                formWindowOpened(evt);
            }
        });

        jLabel1.setText("West Cross Chess Club");

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        jLabel14.setText("Contact Details");

        jLabel15.setText("Phone Number");

        phoneNumberTxt.setEditable(false);

        jLabel16.setText("Email Address");

        emailTxt.setEditable(false);

        jLabel17.setText("Guests are unable to access the West Cross
Chess Club.");

        jLabel18.setText("Please log in to access the Club.");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(43, 43, 43)
                        .addComponent(backBtn))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(274, 274, 274)
```

```
                        .addComponent(jLabel1))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel15)
                            .addComponent(jLabel16))
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.UNRELATED)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 230, Short.MAX_VALUE)
                            .addComponent(emailTxt))
                        .addGap(97, 97, 97)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel18)
                            .addComponent(jLabel17)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(167, 167, 167)
                        .addComponent(jLabel14)))
                .addContainerGap(25, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(28, 28, 28)
                .addComponent(jLabel1)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(backBtn))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(23, 23, 23)
                                .addComponent(jLabel14)
                                .addGap(18, 18, 18)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel15)
                                    .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                                .addGap(41, 41, 41)
```

```
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jLabel16)
                                    .addComponent(emailTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(51, 51, 51)
                                .addComponent(jLabel17)
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED)
                                .addComponent(jLabel18)))
                        .addGap(0, 75, Short.MAX_VALUE)))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user back to the menu
        Menu menu = new Menu(session);
        menu.setVisible(true);
        this.dispose();
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt)
{
        //opening the opening of the window, the system checks to see if
an error has occurred when making the form. If it has, the form is
disposed of.
        if(error){
            this.dispose();
        }
```

```
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubGuest.c
lass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubGuest.c
lass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubGuest.c
lass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubGuest.c
lass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new WestCrossChessClubGuest().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JButton changeContactDetailsBtn;
    private javax.swing.JButton changeContactDetailsBtn1;
    private javax.swing.JButton changeContactDetailsBtn2;
    private javax.swing.JButton changeContactDetailsBtn3;
    private javax.swing.JTextField emailField;
```

```
    private javax.swing.JTextField emailField1;
    private javax.swing.JTextField emailField2;
    private javax.swing.JTextField emailTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JPanel jpanel;
    private javax.swing.JPanel jpanel1;
    private javax.swing.JPanel jpanel2;
    private javax.swing.JLabel memberShipStatusLabel;
    private javax.swing.JLabel memberShipStatusLabel1;
    private javax.swing.JLabel memberShipStatusLabel2;
    private javax.swing.JTextArea membershipArea;
    private javax.swing.JTextArea membershipArea1;
    private javax.swing.JTextArea membershipArea2;
    private javax.swing.JTextField phoneNumberField;
    private javax.swing.JTextField phoneNumberField1;
    private javax.swing.JTextField phoneNumberField2;
    private javax.swing.JTextField phoneNumberTxt;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;
```

```
import java.awt.Color;
import javax.swing.JOptionPane;

/**
 * Creates a new form that allows the leader of the chess club to access
features of the chess club available to the leader.
 * @author mortimer
 */
public class WestCrossChessClubLeader extends javax.swing.JFrame {
    private boolean error=false;//stores whether an error occurred when
loading the form
    private boolean isFormInitialised=false;//stores whether the form
has been intialised
    private SessionInfo session;//stores the session data
    private boolean areContactDetailsBeingEdited=false;//stores whether
the contact detais are currently being altered
    private String phoneNumber;private String email;//stores the default
club contact details
    /**
     * Creates new form WestCrossChessClubLeader
     */
    private WestCrossChessClubLeader() {
        initComponents();
    }
    /**
     * Creates new form WestCrossChessClubLeader
     * @param session The session information of the user
     */
    public WestCrossChessClubLeader(SessionInfo session){
        this.session=session;
        initComponents();
        cancelChangesBtn.setVisible(false);cancelChangesBtn.setEnabled(f
alse);
        createNewLeaderBtn.setBackground(Color.red);//sets the button
the appropriate colour
        attemptToLoadContactInfo();//attempts to laod the contact
information to the form
        isFormInitialised=true;//stores that the form has been
initialised
    }
    /**
     *Loads contact details onto the appropriate forms
     */
    private void attemptToLoadContactInfo(){
        try{
            String[]contactInfo=ThreadClient.getChessClubContactDetails(
);//gets contact details
            phoneNumberTxt.setText(contactInfo[0]);emailTxt.setText(cont
actInfo[1]);//loads contact details onto form
            phoneNumber=contactInfo[0];email=contactInfo[1];//stores
contact details
        }catch(Exception e){
```

```
            JOptionPane.showMessageDialog(this,"An error occurred when
attempting to get the club contact information from the server. Error:
"+e,"Error",JOptionPane.OK_OPTION);
            error=true;//shows that an error has occurred and goes to
the menu if appropriate
            if(isFormInitialised){
                Menu menu = new Menu(session);
                menu.setVisible(true);this.dispose();
            }
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        backBtn = new javax.swing.JButton();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        phoneNumberTxt = new javax.swing.JTextField();
        jLabel16 = new javax.swing.JLabel();
        emailTxt = new javax.swing.JTextField();
        jLabel17 = new javax.swing.JLabel();
        createNewLeaderBtn = new javax.swing.JButton();
        fixturesAndResultsBtn = new javax.swing.JButton();
        viewLeagueTableBtn = new javax.swing.JButton();
        changeContactDetailsBtn = new javax.swing.JButton();
        cancelChangesBtn = new javax.swing.JButton();
        handelRequestsBtn = new javax.swing.JButton();
        viewMemberDetailsBtn = new javax.swing.JButton();
        nonMemberBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("West Cross Chess Club");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowOpened(java.awt.event.WindowEvent evt) {
                formWindowOpened(evt);
            }
        });

        jLabel1.setText("West Cross Chess Club");

        backBtn.setText("Back");
```

```
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        jLabel14.setText("Contact Details");

        jLabel15.setText("Phone Number");

        phoneNumberTxt.setEditable(false);

        jLabel16.setText("Email Address");

        emailTxt.setEditable(false);

        jLabel17.setText("You are the leader of the West Cross Chess
Club");

        createNewLeaderBtn.setText("Appoint New West Cross Chess Club
Leader");
        createNewLeaderBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                createNewLeaderBtnActionPerformed(evt);
            }
        });

        fixturesAndResultsBtn.setText("Fixtures and Results");
        fixturesAndResultsBtn.setMaximumSize(new java.awt.Dimension(165,
29));
        fixturesAndResultsBtn.setMinimumSize(new java.awt.Dimension(165,
29));
        fixturesAndResultsBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                fixturesAndResultsBtnActionPerformed(evt);
            }
        });

        viewLeagueTableBtn.setText("View League Table");
        viewLeagueTableBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewLeagueTableBtnActionPerformed(evt);
            }
        });

        changeContactDetailsBtn.setText("Change Contact Details");
```

```
        changeContactDetailsBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                changeContactDetailsBtnActionPerformed(evt);
            }
        });

        cancelChangesBtn.setText("Cancel Changes");
        cancelChangesBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                cancelChangesBtnActionPerformed(evt);
            }
        });

        handelRequestsBtn.setText("Requests to Join The Club");
        handelRequestsBtn.setMaximumSize(new java.awt.Dimension(165,
29));
        handelRequestsBtn.setMinimumSize(new java.awt.Dimension(165,
29));
        handelRequestsBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                handelRequestsBtnActionPerformed(evt);
            }
        });

        viewMemberDetailsBtn.setText("Member Details");
        viewMemberDetailsBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewMemberDetailsBtnActionPerformed(evt);
            }
        });

        nonMemberBtn.setText("Non - Members");
        nonMemberBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                nonMemberBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
```

```
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(fixturesAndResultsBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(viewLeagueTableBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(handelRequestsBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(viewMemberDetailsBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 15, Short.MAX_VALUE))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(43, 43, 43)
                .addComponent(backBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(createNewLeaderBtn)
                .addGap(54, 54, 54))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(274, 274, 274)
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel17)
                .addGap(33, 33, 33))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel15)
                            .addComponent(jLabel16))
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.UNRELATED)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 230, Short.MAX_VALUE)
                            .addComponent(emailTxt)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(cancelChangesBtn)
                        .addGap(54, 54, 54)
```

```
                        .addComponent(changeContactDetailsBtn))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(167, 167, 167)
                        .addComponent(jLabel14)))
                .addGap(106, 106, 106)
                .addComponent(nonMemberBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(28, 28, 28)
                        .addComponent(jLabel1))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(39, 39, 39)
                        .addComponent(jLabel17)))
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(12, 12, 12)
                        .addComponent(jLabel14)
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel15)
                            .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(41, 41, 41)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel16)
                            .addComponent(emailTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(changeContactDetailsBtn)
                            .addComponent(cancelChangesBtn)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(18, 18, 18)
                        .addComponent(nonMemberBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE)))
```

```
                .addGap(13, 13, 13)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(fixturesAndResultsBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(viewLeagueTableBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(handelRequestsBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(viewMemberDetailsBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 39, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(backBtn)
                    .addComponent(createNewLeaderBtn))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user back to the menu
        Menu menu = new Menu(session);
        menu.setVisible(true);
        this.dispose();
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt)
{
```

```
        //checks to see if any error occurred when initialising the
form. If it has, the current form is disposed and the user is taken back
to the menu
        if(error){
            Menu menu = new Menu(session);
            menu.setVisible(true);
            this.dispose();
        }
    }

    private void
createNewLeaderBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // appoints a new chess club leader
        //asks for comfirmation before making the appointment
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you want to make somebody else the leader? This will mean that you will
no longer be the leader.", "Proceed?",
JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            //appoints somebody else as the chess club leader
            String newLeader = JOptionPane.showInputDialog(this,"Please
input the username of the new leader:");
            if(newLeader==null){//if the user pressses cancel on the
input dialog boc, nothing should occur
                return;
            }
            try{
                //performs validation
                if(ThreadClient.isUserNameAlreadyTaken(newLeader)==false
){//checks to see that the username chosen is actually valid
                    JOptionPane.showMessageDialog(this,"This user does
not exist. Please input a real user.","Error",JOptionPane.OK_OPTION);
                    return;
                }
                if(newLeader.equals(session.getUsername())){//if the new
leader is just the same as the current leader, nothing needs to occur
                    return;
                }
                ThreadClient.setUserAsClubLeader(newLeader);//sets the
new leader
                session.setIsClubLeader(false);//updates the session
data
                JOptionPane.showMessageDialog(this,"You are no longer
the club leader.","Success",JOptionPane.OK_OPTION);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();//takes the user back
to the menu
                return;
            }catch(Exception e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server. "+e,"Error",JOptionPane.OK_OPTION);
                Menu menu = new
```

```
Menu(session);menu.setVisible(true);this.dispose();
                return;
            }
        }else{
            return;
        }
    }

    private void
viewLeagueTableBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the leader to a form that allows them to veiw the
current club league table
        try{
            String[][]leagueTable=ThreadClient.getLeagueTable();//reques
ts the league table from the server
            if(leagueTable.length==0){
                //if the fixtures have not been generated yet, a list of
all club members is requested from the server. The leader is then taken
to the form
                String[]allPlayers=ThreadClient.getMembersOfChessClub();
                LeagueTable tab = new LeagueTable(session, allPlayers);
                tab.setVisible(true);this.dispose();return;
            }
            LeagueTable tab = new LeagueTable(session, leagueTable);
            tab.setVisible(true);this.dispose();return;
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"Error when communicating
with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
            return;
        }
    }

    private void
fixturesAndResultsBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the leader to a form that allows the, to view and edit
the fixtures and results
        try{
            FixturesAndResultsLeader fixtures = new
FixturesAndResultsLeader(session);
            fixtures.setVisible(true);
            this.dispose();
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"Error when communicating
with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
        }
    }
```

```
    private void
changeContactDetailsBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //attempts to change the contact details or save changes to
contact details
        if(areContactDetailsBeingEdited){
            //attempts to save changes to server
            //updates form to reflect the fact that the contact details
are no longer activel being edited
            areContactDetailsBeingEdited=false;
            cancelChangesBtn.setVisible(false);cancelChangesBtn.setEnabl
ed(false);
            changeContactDetailsBtn.setText("Change Contact Details");
            phoneNumberTxt.setEditable(false);
            emailTxt.setEditable(false);
            try{
                ThreadClient.setChessClubContactDetails(phoneNumberTxt.g
etText(), emailTxt.getText());//saves changes
            }catch(Exception e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server: "+e,"Error",JOptionPane.OK_OPTION);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;
            }
            phoneNumber=phoneNumberTxt.getText();email=emailTxt.getText(
);
        }else{//sets it so that the leader can edit the contact details
            //updates the form to allow the contact details to be
changed
            cancelChangesBtn.setVisible(true);cancelChangesBtn.setEnable
d(true);
            changeContactDetailsBtn.setText("Save Changes");
            areContactDetailsBeingEdited=true;
            phoneNumber=phoneNumberTxt.getText();email=emailTxt.getText(
);
            phoneNumberTxt.setEditable(true);
            emailTxt.setEditable(true);
        }
    }

    private void
cancelChangesBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // cancels changes made and exits editing mode
        areContactDetailsBeingEdited=false;
        cancelChangesBtn.setVisible(false);cancelChangesBtn.setEnabled(f
alse);
        changeContactDetailsBtn.setText("Change Contact Details");
        phoneNumberTxt.setEditable(false);
        emailTxt.setEditable(false);
        emailTxt.setText(email);phoneNumberTxt.setText(phoneNumber);//re
verts contact details to their values before editing began
    }
```

```
    private void
handelRequestsBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes the leader to a form that handles requests to join the
club
        try{
            RequestsToJoinClub request = new
RequestsToJoinClub(session);
            request.setVisible(true);
            this.dispose();return;
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"Error when communicating
with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;
        }
    }

    private void
viewMemberDetailsBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // takes the leader to a form where they may view and edit
certain user details of club members. They may also add and remove
        //users from the club in this form
        try{
            MemberDetails mem = new
MemberDetails(session);mem.setVisible(true);this.dispose();
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"Error when communicating
with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;
        }
    }

    private void nonMemberBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // takes the user to a form where they may edit user info. This
form allows non-members to be added to the club.
        //it also allows for accounts to be deleted
        try{
            NonMemberDetails non = new
NonMemberDetails(session);non.setVisible(true);this.dispose();
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"Error when communicating
with the server: "+e,"Error",JOptionPane.OK_OPTION);
            Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();return;
        }
    }

    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubLeader.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubLeader.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubLeader.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubLeader.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new WestCrossChessClubLeader().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JButton cancelChangesBtn;
```

```
    private javax.swing.JButton changeContactDetailsBtn;
    private javax.swing.JButton createNewLeaderBtn;
    private javax.swing.JTextField emailTxt;
    private javax.swing.JButton fixturesAndResultsBtn;
    private javax.swing.JButton handelRequestsBtn;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton nonMemberBtn;
    private javax.swing.JTextField phoneNumberTxt;
    private javax.swing.JButton viewLeagueTableBtn;
    private javax.swing.JButton viewMemberDetailsBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.awt.Color;
import javax.swing.JOptionPane;

/**
 * Creates a new form that allows club members to access features of the
chess club available to them.
 * @author mortimer
 */
public class WestCrossChessClubMember extends javax.swing.JFrame {
    private boolean error=false;//stores whether an error has occurred
    private boolean isFormInitialised=false;//stores whether the form
has been intialised
    private SessionInfo session;//stores the session data
    /**
     * Creates new form WestCrossChessClubMember
     */
    private WestCrossChessClubMember() {
        initComponents();
    }
    /**
     * Creates new form WestCrossChessClubMember
     * @param session The session information of the current member
```

```
     */
    public WestCrossChessClubMember(SessionInfo session){
        this.session=session;//stores the session data
        initComponents();//initialises the form components
        leaveClubBtn.setBackground(Color.red);//sets the button the
appropriate colour
        attemptToLoadContactInfo();//attempts to laod the contact
details onto the form
        isFormInitialised=true;//flags that the form has now been
intialised
    }
    /**
     * Loads contact details onto the appropriate forms
     */
    private void attemptToLoadContactInfo(){
        try{
            String[]contactInfo=ThreadClient.getChessClubContactDetails(
);//loads contact information from the server
            phoneNumberTxt.setText(contactInfo[0]);emailTxt.setText(cont
actInfo[1]);//sets the contact information on the relevant fields
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
attempting to get the club contact information from the server. Error:
"+e,"Error",JOptionPane.OK_OPTION);
            error=true;//flags the fact that an error has occurred
            if(isFormInitialised){
                Menu menu = new Menu(session);
                menu.setVisible(true);this.dispose();
            }
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        backBtn = new javax.swing.JButton();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        phoneNumberTxt = new javax.swing.JTextField();
        jLabel16 = new javax.swing.JLabel();
        emailTxt = new javax.swing.JTextField();
        jLabel17 = new javax.swing.JLabel();
        leaveClubBtn = new javax.swing.JButton();
```

```
        viewFixturesAndResultsBtn = new javax.swing.JButton();
        viewLeagueTableBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("West Cross Chess Club");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowOpened(java.awt.event.WindowEvent evt) {
                formWindowOpened(evt);
            }
        });

        jLabel1.setText("West Cross Chess Club");

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        jLabel14.setText("Contact Details");

        jLabel15.setText("Phone Number");

        phoneNumberTxt.setEditable(false);

        jLabel16.setText("Email Address");

        emailTxt.setEditable(false);

        jLabel17.setText("You are a member of the West Cross Chess
Club.");

        leaveClubBtn.setText("Leave the West Cross Chess Club");
        leaveClubBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                leaveClubBtnActionPerformed(evt);
            }
        });

        viewFixturesAndResultsBtn.setText("View Fixtures and Results");
        viewFixturesAndResultsBtn.setMaximumSize(new
java.awt.Dimension(165, 29));
        viewFixturesAndResultsBtn.setMinimumSize(new
java.awt.Dimension(165, 29));
        viewFixturesAndResultsBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
```

```
                viewFixturesAndResultsBtnActionPerformed(evt);
            }
        });

        viewLeagueTableBtn.setText("View League Table");
        viewLeagueTableBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                viewLeagueTableBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(167, 167, 167)
                        .addComponent(jLabel14))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(274, 274, 274)
                        .addComponent(jLabel1))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel15)
                            .addComponent(jLabel16))
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.UNRELATED)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 230, Short.MAX_VALUE)
                            .addComponent(emailTxt))))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 130, Short.MAX_VALUE)
                .addComponent(jLabel17)
                .addGap(23, 23, 23))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(43, 43, 43)
                        .addComponent(backBtn))
```

```
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(233, 233, 233)
                        .addComponent(viewFixturesAndResultsBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addComponent(viewLeagueTableBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 231,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(leaveClubBtn))
                .addGap(129, 129, 129))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(28, 28, 28)
                .addComponent(jLabel1)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(backBtn)
                        .addContainerGap())
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(23, 23, 23)
                                .addComponent(jLabel14))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED)
                                .addComponent(jLabel17)))
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel15)
                            .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(41, 41, 41)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel16)
```

```
                                .addComponent(emailTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGap(37, 37, 37)
                            .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addComponent(viewFixturesAndResultsBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 185,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.RELATED, 55, Short.MAX_VALUE)
                                    .addComponent(leaveClubBtn)
                                    .addGap(29, 29, 29))
                                .addGroup(jPanel1Layout.createSequentialGrou
p()
                                    .addComponent(viewLeagueTableBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 188,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addContainerGap(javax.swing.GroupLayout
.DEFAULT_SIZE, Short.MAX_VALUE))))))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user back to the menu
        Menu menu = new Menu(session);
        menu.setVisible(true);
        this.dispose();
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt)
```

```
{
        //checks to see if an error has occured during the form
intialisation. If it has, the user is taken back to the menu.
        if(error){
            Menu menu = new Menu(session);
            menu.setVisible(true);
            this.dispose();
        }
    }

    private void leaveClubBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        // lets the user leave the chess club after confirming they wish
to do so
        int confirm = JOptionPane.showConfirmDialog(this, "Are you sure
you would like to leave the chess club?", "Quit?",
JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            //removes user from the chess club
            try{
                ThreadClient.removeUserFromChessClub(session.getUsername
());
                session.setIsClubMember(false);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
                return;
            }catch(Exception e){
                JOptionPane.showMessageDialog(this,"Error when
communicating with the server. "+e,"Error",JOptionPane.OK_OPTION);
                Menu menu = new
Menu(session);menu.setVisible(true);this.dispose();
                return;
            }
        }else{
            return;
        }
    }

    private void
viewLeagueTableBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user to a form to view the current league table
        try{
            String[][]leagueTable=ThreadClient.getLeagueTable();//reques
ts the league table from the server
            if(leagueTable.length==0){
                //if the no fixtures have been generated, then the
league table requires all the chess club members as an input instead
                String[]allPlayers=ThreadClient.getMembersOfChessClub();
                LeagueTable tab = new LeagueTable(session, allPlayers);
                tab.setVisible(true);this.dispose();return;
            }
            LeagueTable tab = new LeagueTable(session, leagueTable);
```

```
            tab.setVisible(true);this.dispose();return;
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"Error when communicating
with the server: "+e,"Error",JOptionPane.OK_OPTION);
            return;
        }
    }

    private void
viewFixturesAndResultsBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes member to form where they can view the fixtures and
results
        try{
            FixturesAndResults fixtures = new
FixturesAndResults(session);
            fixtures.setVisible(true);
            this.dispose();
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"Error when communicating
with the server: "+e,"Error",JOptionPane.OK_OPTION);
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubMember.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubMember.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubMember.
```

```
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubMember.
class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new WestCrossChessClubMember().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JTextField emailTxt;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JButton leaveClubBtn;
    private javax.swing.JTextField phoneNumberTxt;
    private javax.swing.JButton viewFixturesAndResultsBtn;
    private javax.swing.JButton viewLeagueTableBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import javax.swing.JOptionPane;

/**
 * Creates a new form that allows non-member users to access features of
the chess club available to them.
 * @author mortimer
```

```
 */
public class WestCrossChessClubUser extends javax.swing.JFrame {
    private boolean error=false;//stores whether an error has occurred
    private boolean isFormInitialised=false;//stores whether the form
has been initialised
    private SessionInfo session;//stores the session data
    /**
     * Creates new form WestCrossChessClubUser
     */
    private WestCrossChessClubUser() {
        initComponents();
    }
    /**
     * Creates new form WestCrossChessClubUser
     * @param session The session information of the user
     */
    public WestCrossChessClubUser(SessionInfo session){
        this.session=session;
        initComponents();
        attemptToLoadContactInfo();//attempts to load
        if(error==false){//if no error has occured, the form attempts to
load whether the user has requested to join the club
            setFormBasedOnWhetherUserHasRequestedToJoinTheClub();
        }
        isFormInitialised=true;//indicates that the form has now been
intialised
    }
    /**
     * Updates the form based on whether the user has requested to join
the club
     */
    private void setFormBasedOnWhetherUserHasRequestedToJoinTheClub(){
        try{
            boolean hasRequested =
ThreadClient.getHasUserRequestedToJoinChessClub(session.getUsername());/
/gets whether the user has requested to join the club
            //updates the button and label text based on whether the
user has requested to join the club
            if(hasRequested){
                requestToJoinClubBtn.setText("Withdraw request to join
the West Cross Chess Club");
                infoLbl.setText("Please feel free to contact the club or
withdraw your request to join.");
            }else{
                requestToJoinClubBtn.setText("Request to Join the West
Cross Chess Club");
                infoLbl.setText("Please feel free to contact the club or
request to join.");
            }
        }catch(Exception e){
            error=true;//flags that an error has occurred
            JOptionPane.showMessageDialog(this,"An error occurred when
communicating with the server. Error:
```

```
"+e,"Error",JOptionPane.OK_OPTION);
            if(isFormInitialised){
                Menu menu = new Menu(session);menu.setVisible(true);
                this.dispose();
            }
        }
    }
    /**
     * Loads contact details onto the appropriate forms
     */
    private void attemptToLoadContactInfo(){
        try{
            String[]contactInfo=ThreadClient.getChessClubContactDetails(
);//gets the contact details from the server
            phoneNumberTxt.setText(contactInfo[0]);emailTxt.setText(cont
actInfo[1]);//laods the details to the form
        }catch(Exception e){
            JOptionPane.showMessageDialog(this,"An error occurred when
attempting to get the club contact information from the server. Error:
"+e,"Error",JOptionPane.OK_OPTION);
            error=true;
            if(isFormInitialised){
                Menu menu = new Menu(session);
                menu.setVisible(true);this.dispose();

            }
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        backBtn = new javax.swing.JButton();
        jLabel14 = new javax.swing.JLabel();
        jLabel15 = new javax.swing.JLabel();
        phoneNumberTxt = new javax.swing.JTextField();
        jLabel16 = new javax.swing.JLabel();
        emailTxt = new javax.swing.JTextField();
        jLabel17 = new javax.swing.JLabel();
        infoLbl = new javax.swing.JLabel();
        requestToJoinClubBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
```

```
SE);
        setTitle("West Cross Chess Club");
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowOpened(java.awt.event.WindowEvent evt) {
                formWindowOpened(evt);
            }
        });

        jLabel1.setText("West Cross Chess Club");

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        jLabel14.setText("Contact Details");

        jLabel15.setText("Phone Number");

        phoneNumberTxt.setEditable(false);

        jLabel16.setText("Email Address");

        emailTxt.setEditable(false);

        jLabel17.setText("You are not a member of the West Cross Chess
Club.");

        infoLbl.setText("Please feel free to contact the club or request
to join.");

        requestToJoinClubBtn.setText("Request to Join the West Cross
Chess Club");
        requestToJoinClubBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                requestToJoinClubBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
```

```
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(274, 274, 274)
                        .addComponent(jLabel1))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(167, 167, 167)
                        .addComponent(jLabel14))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addContainerGap()
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jLabel15)
                                    .addComponent(jLabel16))
                                .addPreferredGap(javax.swing.LayoutStyle
.ComponentPlacement.UNRELATED)
                                .addGroup(jPanel1Layout.createParallelGr
oup(javax.swing.GroupLayout.Alignment.LEADING, false)
                                    .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.DEFAULT_SIZE, 230, Short.MAX_VALUE)
                                    .addComponent(emailTxt)))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(43, 43, 43)
                                .addComponent(backBtn)))
                        .addGap(97, 97, 97)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(requestToJoinClubBtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 390,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(infoLbl)
                            .addComponent(jLabel17))))
                .addContainerGap(27, Short.MAX_VALUE))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(28, 28, 28)
                .addComponent(jLabel1)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(backBtn))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(23, 23, 23)
                        .addComponent(jLabel14)
```

```
                        .addGap(18, 18, 18)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel15)
                            .addComponent(phoneNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(41, 41, 41)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel16)
                            .addComponent(emailTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(0, 75, Short.MAX_VALUE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(51, 51, 51)
                        .addComponent(jLabel17)
                        .addPreferredGap(javax.swing.LayoutStyle.Compone
ntPlacement.RELATED)
                        .addComponent(infoLbl)
                        .addGap(18, 18, 18)
                        .addComponent(requestToJoinClubBtn,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        //takes the user back to the menu
        Menu menu = new Menu(session);
```

```
        menu.setVisible(true);
        this.dispose();
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt)
{
        //checks to see if an error has occurred during form
intialisation. If it has, then form is disposed and the user is taken
back to the menu
        if(error){
            Menu menu = new Menu(session);
            menu.setVisible(true);
            this.dispose();
        }
    }

    private void
requestToJoinClubBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // adds a request to join the West Cross Chess Club or withdraws
it depending on the requests statis
        try{
            ThreadClient.changeRequestToJoinClubStatus(session.getUserna
me());//chanegs request status
        }catch(Exception e){
           JOptionPane.showMessageDialog(this,"An error occurred when
attempting to communicate with the server. Error:
"+e,"Error",JOptionPane.OK_OPTION);
            error=true;
            if(isFormInitialised){
                Menu menu = new Menu(session);
                menu.setVisible(true);this.dispose();
            }
            return;
        }
        setFormBasedOnWhetherUserHasRequestedToJoinTheClub();//updates
form based on request status
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
```

```
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubUser.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubUser.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubUser.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(WestCrossChessClubUser.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new WestCrossChessClubUser().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JTextField emailTxt;
    private javax.swing.JLabel infoLbl;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField phoneNumberTxt;
    private javax.swing.JButton requestToJoinClubBtn;
    // End of variables declaration
}
```

Source Code – Server System

The server system has two packages: “philipm.cs5.software.development.user.system” and “philipm.cs5.software.development.server.system”.

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;

/**
 * This class loads a form used to change the port number of the server
if appropriate
 * @author mortimer
 */

public class ChangeServerDetails extends javax.swing.JFrame {
    public static final String
SERVER_SETTINGS_TEXT_FILE_NAME="serverSettings.txt";// stores the name
of the text file that stores the server settings
    private boolean isPortNoBeingEdited=false;//stores state of form
(i.e. whether the port number is being edited).
    private static final int DEFAULT_PORT_NO=80;//stores the default
port number setting
    /**
     * Creates new form ChangeServerDetails
     */
    public ChangeServerDetails() {
        initComponents();
        setComponentsBasedOnIsBeingEdited();
        loadDetailsFromFileOntoForm();
    }
    /**
     * Determines whether various components are visible and enabled
depending on whether editing is currently happening.
     */
    private void setComponentsBasedOnIsBeingEdited(){
        setToDefaultBtn.setEnabled(isPortNoBeingEdited);setToDefaultBtn.
setVisible(isPortNoBeingEdited);
        cancelChangesBtn.setEnabled(isPortNoBeingEdited);cancelChangesBt
n.setVisible(isPortNoBeingEdited);
        saveDetailsBtn.setEnabled(isPortNoBeingEdited);saveDetailsBtn.se
tVisible(isPortNoBeingEdited);
        editBtn.setEnabled(!isPortNoBeingEdited);editBtn.setVisible(!isP
ortNoBeingEdited);
        portNumberTxt.setEditable(isPortNoBeingEdited);
    }
```

```
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        portNumberTxt = new javax.swing.JTextField();
        backBtn = new javax.swing.JButton();
        cancelChangesBtn = new javax.swing.JButton();
        saveDetailsBtn = new javax.swing.JButton();
        setToDefaultBtn = new javax.swing.JButton();
        editBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Server Settings");

        jLabel1.setText("Port Number");

        portNumberTxt.setEditable(false);

        backBtn.setText("Back");
        backBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                backBtnActionPerformed(evt);
            }
        });

        cancelChangesBtn.setText("Cancel Changes");
        cancelChangesBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                cancelChangesBtnActionPerformed(evt);
            }
        });

        saveDetailsBtn.setText("Save Details");
        saveDetailsBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                saveDetailsBtnActionPerformed(evt);
            }
```

```
        });

        setToDefaultBtn.setText("Set to Default Value");
        setToDefaultBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                setToDefaultBtnActionPerformed(evt);
            }
        });

        editBtn.setText("Edit Port Number");
        editBtn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                editBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(92, 92, 92)
                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 99,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(60, 60, 60)
                .addComponent(portNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE, 144,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(40, 40, 40)
                .addComponent(backBtn)
                .addGap(106, 106, 106)
                .addComponent(cancelChangesBtn)
                .addGap(27, 27, 27)
                .addComponent(saveDetailsBtn)
                .addGap(42, 42, 42)
                .addComponent(editBtn)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(setToDefaultBtn)
                .addGap(58, 58, 58))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
```

```
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(56, 56, 56)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(portNumberTxt,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 316, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(backBtn)
                    .addComponent(cancelChangesBtn)
                    .addComponent(saveDetailsBtn)
                    .addComponent(setToDefaultBtn)
                    .addComponent(editBtn))
                .addGap(24, 24, 24))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>
/**
 * Updates buttons based on whether the user is currently editing the
port number
 * @param evt the edit button being clicked
 */
    private void editBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        isPortNoBeingEdited=!isPortNoBeingEdited;
        setComponentsBasedOnIsBeingEdited();
    }
    /**
     * Gets the port number inputted and if it is an integer greater
than or equal to 1, it is stored in the correct file and the form is
updated.
```

```
     * @param evt The save details button being pressed by the user
     */
    private void
saveDetailsBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        try{
            int portNum = Integer.parseInt(portNumberTxt.getText());
            if(portNum<=0){//range check
                JOptionPane.showMessageDialog(this, "The port number
must be an integer that is greater than 0.","Invalid
Input",JOptionPane.OK_OPTION);
                return;
            }
            setPortNumberTextFile(portNum);
            isPortNoBeingEdited=!isPortNoBeingEdited;
            setComponentsBasedOnIsBeingEdited();loadDetailsFromFileOntoF
orm();
        }catch(NumberFormatException e){
            JOptionPane.showMessageDialog(this, "The port number must be
an integer that is greater than 0.","Invalid
Input",JOptionPane.OK_OPTION);
            return;
        }
    }
    /**
     * Sets the port number back to its default value. This default
value is stored under the constant titled DEFUALT_PORT_NO
     * @param evt the button being pressed by the user
     */
    private void
setToDefaultBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        setPortNumberTextFile(DEFAULT_PORT_NO);
        isPortNoBeingEdited=!isPortNoBeingEdited;
        setComponentsBasedOnIsBeingEdited();loadDetailsFromFileOntoForm(
);
    }
    /**
     * Cancels changes and reverts the port number text field to the
value stored in the text file.
     * @param evt the button being pressed
     */
    private void
cancelChangesBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        isPortNoBeingEdited=!isPortNoBeingEdited;
        setComponentsBasedOnIsBeingEdited();loadDetailsFromFileOntoForm(
);
    }
    /**
     * Takes the user to the ServerUI form after appropriate
confirmation
     * @param evt The back button is pressed by the user
```

```
     */

    private void backBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        if(isPortNoBeingEdited){
            int confirm = JOptionPane.showConfirmDialog(this, "If you
continue, any unsaved changes to the port number will not be saved.
Would you like to
continue?","Continue?",JOptionPane.YES_NO_CANCEL_OPTION);
            if(confirm==JOptionPane.YES_OPTION){
                ServerUI serv = new ServerUI();
                serv.setVisible(true);
                this.dispose();
            }
            return;
        }else{
            ServerUI serv = new ServerUI();
            serv.setVisible(true);
            this.dispose();
        }

    }
    /*
    Loads details stored in the server settings text file and displays
them on the form
    */
    private void loadDetailsFromFileOntoForm(){
        try{
            FileReader read = new
FileReader(SERVER_SETTINGS_TEXT_FILE_NAME);
            BufferedReader buffRead = new BufferedReader(read);
            portNumberTxt.setText(buffRead.readLine());read.close();buff
Read.close();
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An error occured when
reading from file: "+SERVER_SETTINGS_TEXT_FILE_NAME+". Error details
"+e, "Error", JOptionPane.OK_OPTION);
        }
    }
    /**
     * Stores the new port number in the correct text file
     * @param portNumber the new port number
     */
    private void setPortNumberTextFile(int portNumber){
        try{
            FileWriter write = new
FileWriter(SERVER_SETTINGS_TEXT_FILE_NAME,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write(String.valueOf(portNumber));
            buffWrite.flush();write.flush();
            buffWrite.close();write.close();
        }catch(IOException e){
            JOptionPane.showMessageDialog(this, "An error occured when
```

```
writing to file: "+SERVER_SETTINGS_TEXT_FILE_NAME+". Error details "+e,
"Error", JOptionPane.OK_OPTION);
        }
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ChangeServerDetails.class
.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ChangeServerDetails.class
.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ChangeServerDetails.class
.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ChangeServerDetails.class
.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ChangeServerDetails().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton backBtn;
    private javax.swing.JButton cancelChangesBtn;
    private javax.swing.JButton editBtn;
    private javax.swing.JLabel jLabel1;
```

```
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextField portNumberTxt;
    private javax.swing.JButton saveDetailsBtn;
    private javax.swing.JButton setToDefaultBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

/**
 * This static class is used to interact with the fixture and results
stored.
 * @author mortimer
 */
public class Fixture {
    //stores various constants used thorughout the fixture code- such as
values that represent a win or draw
    public static final String FILE_NAME="fixtures.txt";
    public static final String SPLIT=Server.SPLIT_ITEM;
    public static final String GAME_NOT_YET_PLAYED="-1";
    private static final String UNABLE_TO_CREATE_GAME="cantCreateGame";
    public static final String BYE_GAME_STATE="-2";//indicates that the
game is a bye. Does not necesarily mean that game is scheduled against a
bye opponent
    public static final String BY_NAME="@";
    private static final String NO_PLAYERS_TO_ADD="aasifoakofako325";
    public static final String TOO_FEW_PLAYERS_ARRAY="Unable to load
array too few players";
    public static final String GAME_WHITE_WIN="whiteW";
    public static final String GAME_BLACK_WIN="blackW";
    public static final String GAME_DRAW="draww";
    /**
     * Creates a new set of fixtures from scratch with the status of not
having been played
     * @param players The name of all players. They must all be unique.
     * @throws Exception An exception if the fixtures can't be generated
```

```
due to player size
     */
    public static void createdNewFixturesFromStart(String[]players)
throws Exception{
        if(players.length==0 || players.length==1){
            //stores the fact that there are too few players to play a
game
            try{
                FileWriter w = new FileWriter(FILE_NAME,false);
                BufferedWriter buffW = new BufferedWriter(w);
                buffW.write(UNABLE_TO_CREATE_GAME);
                buffW.flush();w.flush();
                buffW.close();w.close();
                throw new Exception("Unable to create game as too few
players are present");
            }catch(IOException e){
                System.out.println(e);
            }
        }
        if(players.length % 2 == 1){
            String newNames[]=new String[players.length+1];//creates a
new array and adds a bye to the list (as only an even number of players
can exist in a head-to-head league.
            //copies old array to new one
            for(int i=0;i<players.length;i++){
                newNames[i]=players[i];
            }
            newNames[players.length]=BY_NAME;
            players=newNames;
        }
        //randomly swaps names in list
        String buffer;int index;
        Random rnd = new Random();
        for(int i=0;i<players.length;i++){
            index=rnd.nextInt(players.length);
            buffer = players[index];
            players[index]=players[i];
            players[i]=buffer;
        }
        String[][][]fixtures=new String[2*(players.length  -
1)][3][players.length/2];//stores all fixtures.
        //the 3d array is indexed fixtures[z][y][x]. z represents the
round number, y represents some attribute in the round.
        //For example, y = 0 represents all white players. y=1
represents black players. y=2 represents the game state.
        //x represents the match element in the round. E.g.
fixtures[0][0][0] return the player playing as white in the nought round
and in the nought fixture
        //sets all games to have not yet been played
        for(int round=0;round<fixtures.length;round++){
            for(int game=0;game<players.length/2;game++){
                fixtures[round][2][game]=GAME_NOT_YET_PLAYED;
            }
```

```
        }
        //generates fixtures for first round
        for(int y=0;y<2;y++){
            for(int x=0;x<players.length/2;x++){
                fixtures[0][y][x]=players[y*(players.length/2) +x];
            }
        }
        //loads first half of fixtures
        for(int round=1;round<fixtures.length/2;round++){//fills in the
rest of the round
            //pushes all of top to the right apart from first element,
which is fixed
            fixtures[round][0][0]=fixtures[0][0][0];
            for(int x=1;x<players.length/2 -1;x++){
                fixtures[round][0][x+1]=fixtures[round-1][0][x];
            }
            //adds in buffer at appropriate spot
            fixtures[round][0][1]=fixtures[round-1][1][0];
            //pushes all of bottom to the left
            for(int x=0;x<players.length/2 -1;x++){
                fixtures[round][1][x]=fixtures[round-1][1][x+1];
            }
            fixtures[round][1][players.length/2 -1]=fixtures[round-
1][0][players.length/2 -1];
        }
        //second half of fixtures is the same as the first half just
swapping them around
        for(int round=fixtures.length/2;round<fixtures.length;round++){
            for(int x=0;x<players.length/2;x++){
                fixtures[round][0][x]=fixtures[round-
(fixtures.length/2)][1][x];
                fixtures[round][1][x]=fixtures[round-
(fixtures.length/2)][0][x];
            }
        }
        //shuffles the games in each round randomly and then shuffles
the rounds randomly
        String[][]bufferArr;
        for(int round=0;round<fixtures.length;round++){
            //shuffles games within round randomly
            for(int game=0;game<players.length/2;game++){
                index=rnd.nextInt(players.length/2);
                buffer=fixtures[round][0][game];
                fixtures[round][0][game]=fixtures[round][0][index];
                fixtures[round][0][index]=buffer;
                buffer=fixtures[round][1][game];
                fixtures[round][1][game]=fixtures[round][1][index];
                fixtures[round][1][index]=buffer;
            }
            //shuffles the rounds randomly
            index=rnd.nextInt(fixtures.length);
            bufferArr=fixtures[index];
            fixtures[index]=fixtures[round];
```

```
            fixtures[round]=bufferArr;
        }
        for(int round=0;round<fixtures.length;round++){
            for(int game=0;game<players.length/2;game++){
                if(fixtures[round][0][game].equals(BY_NAME) ||
fixtures[round][1][game].equals(BY_NAME)){//stores games which are byes
                    fixtures[round][2][game]=BYE_GAME_STATE;
                }
            }

        }
        //writes generated fixtures to a file, overwriting anything that
was previously there, in a form that the computer can intepret
        writeToFile(fixtures, new String[0]);
    }
    /**
     * Writes the fixtures list to the file.
     * @param fixtures The fixtures array to write
     * @param nameOfPlayersAddedLate The players added halfway through
the season
     * @throws IOException An IOException may occur when writing to file
     */
    public static void writeToFile(String fixtures[][][], String[]
nameOfPlayersAddedLate) throws IOException{
            int playerLength=fixtures[0][0].length*2;
            FileWriter write = new FileWriter(FILE_NAME,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write(String.valueOf(playerLength));buffWrite.newL
ine();
            //writes fixtures for white
            for(int round=0;round<fixtures.length;round++){
                for(int game=0;game<playerLength/2;game++){
                    if(round==0 && game==0){
                        buffWrite.write(fixtures[round][0][game]);
                    }else{
                        buffWrite.write(SPLIT+fixtures[round][0][game]);
                    }
                }
            }
            buffWrite.newLine();
            //writes fixtures for black
            for(int round=0;round<fixtures.length;round++){
                for(int game=0;game<playerLength/2;game++){
                    if(round==0 && game==0){
                        buffWrite.write(fixtures[round][1][game]);
                    }else{
                        buffWrite.write(SPLIT+fixtures[round][1][game]);
                    }
                }
            }
            buffWrite.newLine();
            //writes game states to file
            for(int round=0;round<fixtures.length;round++){
```

```
                for(int game=0;game<playerLength/2;game++){
                     if(round==0 && game==0){
                        buffWrite.write(fixtures[round][2][game]);
                    }else{
                        buffWrite.write(SPLIT+fixtures[round][2][game]);
                    }
                }
            }
            //writes queue of players to add to file
            buffWrite.newLine();
            if(nameOfPlayersAddedLate.length==0){
                buffWrite.write(NO_PLAYERS_TO_ADD);
            }else{
                String lineWrite=nameOfPlayersAddedLate[0];
                for(String name: nameOfPlayersAddedLate){
                    lineWrite = lineWrite+SPLIT+name;
                }
                buffWrite.write(lineWrite);
            }
            buffWrite.flush();write.flush();
            buffWrite.close();write.close();
    }
    /**
     * Gets an array containing the people who wish to play in the
league
     * @return The array
     * @throws IOException An exception when reading from the text file
     */
    public static String[]getListOfPeopleToAddedDuringSeason()throws
IOException{
        FileReader read = new FileReader(FILE_NAME);
        BufferedReader buffRead = new BufferedReader(read);
        buffRead.readLine();buffRead.readLine();buffRead.readLine();buff
Read.readLine();
        String line = buffRead.readLine();
        buffRead.close();read.close();
        if(NO_PLAYERS_TO_ADD.equals(line)){
            return new String[0];
        }
        return line.split(SPLIT);
    }
    /**
     * Loads the fixtures array from the text file.
     * @return The array of fixtures
     * @throws java.lang.Exception An exception if the array can't be
loaded due to the number of players being too small.
     */
    public static String[][][]loadFixturesArray() throws Exception{
        try{
            FileReader read = new FileReader(FILE_NAME);
            BufferedReader buffRead = new BufferedReader(read);
            String lin=buffRead.readLine();
            if(lin.equals(UNABLE_TO_CREATE_GAME)){//if there are too few
```

```
players and hence no fixture has been created, an exception is thrown
                buffRead.close();read.close();
                throw new Exception(TOO_FEW_PLAYERS_ARRAY);
            }
            int lengthOfFile = Integer.parseInt(lin);
            String[][][]fixtures = new String[2*(lengthOfFile -
1)][3][lengthOfFile/2];//stores the fixtures / results
            String line[]=buffRead.readLine().split(SPLIT);int
lineIndex=0;//stores the white players
            for(int round=0;round<fixtures.length;round++){//loops
through file updating all white players
                for(int game=0;game<lengthOfFile/2;game++){
                    fixtures[round][0][game]=line[lineIndex++];
                }
            }
            line=buffRead.readLine().split(SPLIT);lineIndex=0;//stores
all black players
            for(int round=0;round<fixtures.length;round++){//loops
through array to store all black players
                for(int game=0;game<lengthOfFile/2;game++){
                    fixtures[round][1][game]=line[lineIndex++];
                }
            }
            line=buffRead.readLine().split(SPLIT);lineIndex=0;//stores
the game states of each game
            for(int round=0;round<fixtures.length;round++){
                for(int game=0;game<lengthOfFile/2;game++){
                    fixtures[round][2][game]=line[lineIndex++];
                }
            }
            buffRead.close();read.close();
            return fixtures;
        }catch(IOException e){
            System.out.println("Error "+e);
            throw new Exception("Error "+e);
        }
    }
    /**
     * Adds a player who is not part of the fixtures already to the
league. If no games have been played, the fixtures will be generated
from scratch. Otherwise, they cannot actually play until a new season
begins.
     * @param playerName The player name
     * @param allClubMembers An array of all club member names
(INCLUDING PLAYER TO BE ADDED)
     * @throws Exception An exception may occur when reading and writing
to text files
     */
    public static void addPlayerToFixtures(String playerName, String[]
allClubMembers) throws Exception{
        /*
        If no games have been played yet, then just regernate fixtures
from scratch. If the data is stored as too few players
```

```
        then generate fixtures from scratch also.

        Else, player is added to list of players who cannot play until
next season.
        */
        try{
            String[][][]fixtures = loadFixturesArray();
            //checks to see if any games have been played
            boolean hasMatchBeenPlayed=false;
            for(int round=0;round<fixtures.length;round++){
                for(int match=0;match<fixtures[0][0].length;match++){
                    if((fixtures[round][2][match].equals(BYE_GAME_STATE)
|| fixtures[round][2][match].equals(GAME_NOT_YET_PLAYED))==false){
                        hasMatchBeenPlayed=true;
                        break;
                    }
                }
                if(hasMatchBeenPlayed){
                    break;
                }
            }
            if(hasMatchBeenPlayed){
                //adds player to a list of people added halfway through
season
                String[]playersAdded=getListOfPeopleToAddedDuringSeason(
);
                String playersNew[]=new String[playersAdded.length+1];
                for(int i=0;i<playersAdded.length;i++){
                    playersNew[i]=playersAdded[i];
                }
                playersNew[playersAdded.length]=playerName;
                writeToFile(fixtures, playersNew);
                return;
            }else{
                //generates the fixtures from scratch as no games have
been played
                createdNewFixturesFromStart(allClubMembers);
                return;
            }
        }catch(Exception e){
            //catches exceptions. An exception may be thrown due to file
handling. If this is the case, this is a genuine error and needs to be
thrown.
            //The error could simply be that fixtures cannot be loaded
as there are too few players to generate any fixtures. In this case,
fixtures are generated again.
            if(TOO_FEW_PLAYERS_ARRAY.equals(e.getMessage())){
                createdNewFixturesFromStart(allClubMembers);
                return;
            }else{
                throw e;
            }
        }
```

```
    }
    /**
     * Checks to see if the player is part of the league (or waiting to
be) and removes them if they are. If they appear in the fixtures, those
games will be marked as bye games.
     * @param playerName The name of the player
     * @throws Exception An exception may occur when reading to and from
a text file
     */
    public static void removePlayerFromFixtures(String playerName)throws
Exception{
        try{
            String fixtures[][][]=loadFixturesArray();
            //checks to see if player is in fixture. If they are, all
their matches are set to bye games (even the ones already played)
            boolean playerFound=false;
            for(int round=0;round<fixtures.length;round++){
                for(int match=0;match<fixtures[0][0].length;match++){
                    if(fixtures[round][0][match].equals(playerName)){
                        fixtures[round][0][match]=BY_NAME;
                        fixtures[round][2][match]=BYE_GAME_STATE;
                        playerFound=true;
                    }else
if(fixtures[round][1][match].equals(playerName)){
                        fixtures[round][1][match]=BY_NAME;
                        fixtures[round][2][match]=BYE_GAME_STATE;
                        playerFound=true;
                    }
                }
            }
            if(playerFound){
                //writes the new fixtures to file
                writeToFile(fixtures,
getListOfPeopleToAddedDuringSeason());
                return;
            }else{
                //checks to see if player was added mid way through
season.
                int indexOfPlayer=-1;
                String[]people=getListOfPeopleToAddedDuringSeason();
                for(int i=0;i<people.length;i++){
                    if(people[i].equals(playerName)){
                        indexOfPlayer=i;
                        break;
                    }
                }
                if(indexOfPlayer==-1){
                    return ;//no action is needed
                }else{
                    //removes player from list of people waiting
                    String peopleNew[]=new String[people.length-1];
                    for(int i=0;i<indexOfPlayer;i++){
```

```
                        peopleNew[i]=people[i];
                    }
                    for(int i=indexOfPlayer+1;i<people.length;i++){
                        peopleNew[i-1]=people[i];
                    }
                    //writes updated file
                    writeToFile(fixtures, peopleNew);
                    return;
                }
            }
        }catch(Exception e){
            //an exception may occur if there are too few players to
create fixtures. In this case, no action is needed.
            if(TOO_FEW_PLAYERS_ARRAY.equals(e.getMessage())){
                return;
            }else{
                throw e;
            }
        }

    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;



/**
 * The main class of the server system, used to create the server form.
 * @author mortimer
 */
public class PhilipMCS5SoftwareDevelopmentServerSystem {

    /**
     * The main method that loads the server form
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        ServerUI server = new ServerUI();//creates a server form
        server.setVisible(true);
    }

}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Random;
import philipm.cs5.software.development.user.system.InputPackages;

/**
 * Creates a new Queue which handle the server's requests to write to
files and change arrays. Essentially, this object
 * allows threads from an unknown number of client's to interact with a
server sequentially. All processing is performed on one thread
 * to avoid issues that may arise from two threads writing
simultaneously to the same file. This is justified as the user base is
very small.
 * @author mortimer
 */
public class Queue implements Runnable{
    private InputPackages packages[];//stores the queue of input
packages
    private final Server SERVER;//stores the server
    private boolean keepThreadRunning=true;//stores whether thread
should keep running
    private boolean isRunning=true;
    /**
     * Creates a new queue object
     * @param server The server
     */
    public Queue (Server server){
        this.SERVER=server;
        packages = new InputPackages[0];
    }
    @Override
    public void run(){
        //exectutes all tasks until thread is ended - these tasks are
write operations
            while(getKeepThreadRunning()){
                if(getPackages().length>0){//performs any tasks needed
to execute
                    handlePackage();
```

```
                }
            }
        setIsRunningFalse();
    }
    /**
     * Sets the variable is running to false
     */
    private synchronized  void setIsRunningFalse(){
        isRunning=false;
    }
    /**
     * Searches for the user in the user files. Returns a UserInfo
object with appropriate details
     * @param username The desired username
     * @param password The user inputted password
     * @return The UserInfo object
     */
    public synchronized UserInfo
getUserDetailsAndTryLogInWithUsername(String username, String password){
        String record[]=
Search.searchForRecordWithPrimaryKeyAtNoughtElement(SERVER.userFile,
username,UserInfo.WIDTH_OF_USER_DETAILS_FILE);//searches for user record
        return new UserInfo(record,password);
    }
    /**
     * Looks for a user with email address and attempts to log them in.
     * @param email The email address
     * @param password The password
     * @return A UserInfo object that stores the user data
     */
    public synchronized  UserInfo getEmailAddressAndTryLogIn(String
email,String password){
        String emailLinkRecord[] =
Search.searchForRecordWithPrimaryKeyAtNoughtElement(SERVER.emailLinkFile
, email,UserInfo.WIDTH_OF_EMAIL_LINK_FILE);
        if(emailLinkRecord[0]==null){//if email address is not found, an
appropriate UserInfo instance is returned
            return UserInfo.CreateUserInfoEmailNotFound();
        }else{
            return
getUserDetailsAndTryLogInWithUsername(emailLinkRecord[1],
password);//attempts to log in with found username
        }
    }
    /**
     * Gets all games that contain the specified user. This includes
games that are pending approval
     * @param userName The user name
     * @return All games with the user. The array will always be of
length 3, even if no games exist
     */
    public synchronized String[][] getAllGamesWithUser(String userName){
        String[][]game = new String[3][Server.WIDTH_OF_ONLINE_FILE];
```

```
        for(int y=0;y<3;y++){
            for(int x=0;x<Server.WIDTH_OF_ONLINE_FILE;x++){
                game[y][x]="";
            }
        }
        int i=0;//stores the element of the game array being accessed
        //performs a linear search through file and adds game data where
appropriate
        for(int
gameIndex=0;gameIndex<SERVER.onlineGames.length;gameIndex++){
            if((userName.equals(SERVER.onlineGames[gameIndex][1])&&SERVE
R.onlineGames[gameIndex][8].equals("false"))||(userName.equals(SERVER.on
lineGames[gameIndex][2]) &&
SERVER.onlineGames[gameIndex][9].equals("false"))){
                for(int x=0;x<Server.WIDTH_OF_ONLINE_FILE;x++){
                    game[i][x]=SERVER.onlineGames[gameIndex][x];
                }
                if(i==2){
                    break;
                }
                i++;
            }
        }
        return game;
    }
    /**
     * Gets whether the user has requested to join the club
     * @param userName The username
     * @return True if the user has, otherwise false.
     */
    public synchronized boolean hasUserRequestedToJoinClub(String
userName){
        //attempts to find user
        int indexOfRecord=Search.searchForRecordIndex(SERVER.userFile,
userName);
        if(indexOfRecord==Search.INDEX_NOT_FOUND){
            return false;
        }

        //checks to see if user has requested to join the club
        return SERVER.userFile[indexOfRecord][16].equals("true");
    }
    /**
     * Gets the club contact information
     * @return The contact details
     */
    public synchronized  String[] getClubContactInfo(){
        return SERVER.clubContactDetails;

    }
    /**
     * Makes a user who is not a club member or leader a member of the
chess club
```

```
     * @param playerName The name of the player
     */
    public synchronized void makeNonMemberNonLeaderUserClubMember(String
playerName){
        //searches for the user record in the user file
        int index= Search.searchForRecordIndex(SERVER.userFile,
playerName);
        if(index==Search.INDEX_NOT_FOUND){
            return;
        }
        //updates the variable indicating whether the user is a club
member
        SERVER.userFile[index][14]="true";SERVER.userFile[index][13]="fa
lse";
        //writes change to text file
        SERVER.writeArrayToFileWithSplit(SERVER.userFile,
Server.USER_FILE_NAME);
        //adds user to fixtures if relevant
        try{
            Fixture.addPlayerToFixtures(playerName,
getAllClubMembers());
        }catch(Exception e){
            System.out.println("Possible non fatal adding user to chess
club: "+e);
            return;
        }
    }
    /**
     * Gets the variable that indicates whether the thread should
continue to run
     * @return The variable
     */
    private synchronized boolean getKeepThreadRunning(){
        return keepThreadRunning;
    }
    /**
     * Gets the queue of tasks
     * @return The queue
     */
    private synchronized InputPackages[] getPackages(){
        return packages;
    }
    /**
     * Sets the variable to false - indicating that the thread should
not continue to run
     */
    private synchronized void setKeepRunningFalse(){
        keepThreadRunning=false;
    }
    /**
     * Gets whether the thread is currently alive
     * @return Whether the thread is running
     */
```

```
    private synchronized boolean getIsRunning(){
        return isRunning;
    }
    /**
     * Removes a single package from the queue and handles it
     */
    private synchronized void handlePackage(){
        String command;
        InputPackages message;
        message=removeFromQueue();
        command=message.getActionName();
        //gets the action command from the packages and performs the
appropriate corresponding task
        switch (command) {
            case InputPackages.CREATE_ACCOUNT:
                createAccount(message.getUserRecod(),
message.getUserName(), message.getEmail());
                break;
            case InputPackages.CREATE_RANDOM_GAME:
                handleRequestForNewOnlineRandomGame(message.getPackageTo
Send()[1]);
                break;
            case InputPackages.DELETE_GAME:
                deleteGame(Integer.parseInt(message.getPackageToSend()[1
]), Double.parseDouble(message.getPackageToSend()[2]));
                break;
            case InputPackages.UPDATED_ONLINE_GAME_RECORD:
                String record[]=new String[Server.WIDTH_OF_ONLINE_FILE];
                for(int i=1;i<message.getPackageToSend().length;i++){
                    record[i-1]=message.getPackageToSend()[i];
                }   updateOnlineGameRecord(record);
                break;
            case InputPackages.CHANGE_STATUS_REQUEST_TO_JOIN_CLUB:
                updateRequestToJoinClubStatus(message.getPackageToSend()
[1]);
                break;
            case InputPackages.REMOVE_USER_FROM_CHESS_CLUB:
                removeUserFromChessClub(message.getPackageToSend()[1]);
                break;
            case InputPackages.UPDATED_FIXTURES_AND_RESULTS:
                updateFixturesAndResults(message.getFixturesAndResultsAr
ray());
                break;
            case InputPackages.CREATE_NEW_SEASON:
                createNewSeason();
                break;
            case InputPackages.SET_CONTACT_DETAILS:
                setContactDetails(message.getPackageToSend()[1],message.
getPackageToSend()[2]);
                break;
            case InputPackages.REMOVE_REQUEST_JOIN_CLUB:
                withdrawRequestToJoin(message.getPackageToSend()[1]);
                break;
```

```
            case InputPackages.ADD_USER_TO_CLUB:
                makeUserAClubMember(message.getPackageToSend()[1]);
                break;
            case InputPackages.UPDATE_USER_INFO_FILE_RECORD:
                String newRecord[]=new
String[UserInfo.WIDTH_OF_USER_DETAILS_FILE];
                for(int i=0;i<newRecord.length;i++){
                    newRecord[i]=message.getPackageToSend()[i+1];
                }   updateUserFileRecord(newRecord);
                break;
            case InputPackages.REMOVE_NON_MEMBER_USER:
                removeNonMemberUser(message.getPackageToSend()[1]);
                break;
            default:
                break;
        }
    }
    /**
     * Deletes a user from the database. This user must not be a member
of the West Cross Chess Club.
     * @param userName The name of the user to remove
     */
    private synchronized void removeNonMemberUser(String userName){
        int index=Search.searchForRecordIndex(SERVER.userFile,
userName);
        if(index!=Search.INDEX_NOT_FOUND){
            //removes user from user file - by creating a new array that
does not include their user record.
            String newUserFile[][]=new String[SERVER.userFile.length-
1][UserInfo.WIDTH_OF_USER_DETAILS_FILE];
            for(int i=0;i<index;i++){
                for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                    newUserFile[i][x]=SERVER.userFile[i][x];
                }
            }
            for(int i=index+1;i<SERVER.userFile.length;i++){
                for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                    newUserFile[i-1][x]=SERVER.userFile[i][x];
                }
            }
            SERVER.userFile=newUserFile;
            SERVER.writeArrayToFileWithSplit(SERVER.userFile,
Server.USER_FILE_NAME);
        }
        //removes user from email link file if appropriate
        int indexOfEmail=-1;
        for(int record=0;record<SERVER.emailLinkFile.length;record++){
            if(SERVER.emailLinkFile[record][1].equals(userName)){
                indexOfEmail=record;
                break;
            }
        }
        if(indexOfEmail!=-1){
```

```
            //updates email file
            String[][] newEmailFile=new
String[SERVER.emailLinkFile.length-
1][UserInfo.WIDTH_OF_EMAIL_LINK_FILE];
            for(int i=0;i<index;i++){
                for(int x=0;x<UserInfo.WIDTH_OF_EMAIL_LINK_FILE;x++){
                    newEmailFile[i][x]=SERVER.emailLinkFile[i][x];
                }
            }
            for(int i=index+1;i<SERVER.emailLinkFile.length;i++){
                for(int x=0;x<UserInfo.WIDTH_OF_EMAIL_LINK_FILE;x++){
                    newEmailFile[i-1][x]=SERVER.emailLinkFile[i][x];
                }
            }
            SERVER.emailLinkFile=newEmailFile;
            SERVER.writeArrayToFileWithSplit(SERVER.emailLinkFile,
Server.EMAIL_FILE_NAME);
        }
        //removes user from online games. This is done by appending all
games that are not from the user to an ArrayList.
        ArrayList<String[]> onlineGamesNew=new
ArrayList<String[]>(SERVER.onlineGames.length);
        String[]recordBuffer;
        for(int game=0;game<SERVER.onlineGames.length;game++){
            //checks to see if user is in the game
            if((SERVER.onlineGames[game][1].equals(userName) ||
SERVER.onlineGames[game][2].equals(userName)) ==false){
                recordBuffer=new String[Server.WIDTH_OF_ONLINE_FILE];
                for(int x=0;x<recordBuffer.length;x++){
                    recordBuffer[x]=SERVER.onlineGames[game][x];
                }
                onlineGamesNew.add(recordBuffer);
            }
        }
        //creates a new online game array from the array list
        String newOnlineGameArray[][]=new
String[onlineGamesNew.size()][Server.WIDTH_OF_ONLINE_FILE];
        for(int i=0;i<newOnlineGameArray.length;i++){
            recordBuffer=onlineGamesNew.get(i);
            for(int x=0;x<Server.WIDTH_OF_ONLINE_FILE;x++){
                newOnlineGameArray[i][x]=recordBuffer[x];
            }
        }
        //changes are saved
        SERVER.onlineGames=newOnlineGameArray;
        SERVER.writeArrayToFileWithSplit(SERVER.onlineGames,
Server.ONLINE_GAMES_FILE);
    }
    /**
     * Updates the user record to a new array.
     * @param newRecord The new record
     */
    private synchronized void updateUserFileRecord(String []newRecord){
```

```
        int index=Search.searchForRecordIndex(SERVER.userFile,
newRecord[0]);//locates user
        if(index==Search.INDEX_NOT_FOUND){
            return;
        }
        //updates data
        for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
            SERVER.userFile[index][x]=newRecord[x];
        }
        //writes the new array to the text file
        SERVER.writeArrayToFileWithSplit(SERVER.userFile,
Server.USER_FILE_NAME);
    }
    /**
     * Makes the user a club member
     * @param userName The name of the user
     */
    private synchronized void makeUserAClubMember(String userName){
        int userIndex=Search.searchForRecordIndex(SERVER.userFile,
userName);//find the user in the user file
        if(userIndex==Search.INDEX_NOT_FOUND){
            return;
        }
        //updates variables to indicate that they are a club member who
is not requesting to join the club and who is not the leader
        SERVER.userFile[userIndex][16]="false";SERVER.userFile[userIndex
][14]="true";
        SERVER.userFile[userIndex][13]="false";
        try{
            Fixture.addPlayerToFixtures(userName,
getAllClubMembers());//adds the players to the fixtures
        }catch(Exception e){
            System.out.println("Error when adding user to club: "+e);
        }

    }
    /**
     * Withdraws a user's request to join the West Cross Chess Club
     * @param userName The name of the user who's request is to be
withdrawn
     */
    private synchronized void withdrawRequestToJoin(String userName){
        int userIndex=Search.searchForRecordIndex(SERVER.userFile,
userName);//locates user in user file
        if(userIndex==Search.INDEX_NOT_FOUND){
            return;
        }
        SERVER.userFile[userIndex][16]="false";//updates variable
indicating that they are not requesting to join the club
        SERVER.writeArrayToFileWithSplit(SERVER.userFile,
Server.USER_FILE_NAME);//writes change to text file
    }
    /**
```

```
     * Returns the user data of every non club member. (I.e. the user
file record of every non chess club member).
     * @return The data
     */
    public synchronized String[][] getAllNonMemberDetails(){
        //searches through every user in the user file to see if they
have requested to join the club.
        String currentUser[];
        ArrayList<String[]> allNonMembers=new ArrayList<String[]>();
        for(int user=0;user<SERVER.userFile.length;user++){
            if(SERVER.userFile[user][14].equals("false")){
                currentUser=new
String[UserInfo.WIDTH_OF_USER_DETAILS_FILE];
                //copies array
                for(int x=0;x<currentUser.length;x++){
                    currentUser[x]=SERVER.userFile[user][x];
                }
                allNonMembers.add(currentUser);//adds user to array list
            }
        }
        String[][]nonMem = new
String[allNonMembers.size()][UserInfo.WIDTH_OF_USER_DETAILS_FILE];//conv
erts array list to an array
        for(int user=0;user<nonMem.length;user++){
            for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                nonMem[user][x]=allNonMembers.get(user)[x];
            }
        }
        return nonMem;
    }
    /**
     * Returns the user data of every club member. (I.e. the user file
record of every chess club member).
     * @return The data
     */
    public synchronized String[][] getAllClubMemberDetails(){
        //searches through every user in the user file to see if they
have requested to join the club.
        String currentUser[];
        ArrayList<String[]> allMembers=new ArrayList<String[]>();
        for(int user=0;user<SERVER.userFile.length;user++){
            if(SERVER.userFile[user][14].equals("true")){
                currentUser=new
String[UserInfo.WIDTH_OF_USER_DETAILS_FILE];
                //copies array
                for(int x=0;x<currentUser.length;x++){
                    currentUser[x]=SERVER.userFile[user][x];
                }
                allMembers.add(currentUser);//adds user to ArrayList
            }
        }
        String[][]mem = new
String[allMembers.size()][UserInfo.WIDTH_OF_USER_DETAILS_FILE];//convert
```

```
s array list to a 2d array
        for(int user=0;user<mem.length;user++){
            for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                mem[user][x]=allMembers.get(user)[x];
            }
        }
        return mem;
    }
    /**
     * Updates the club contact information.
     * @param phoneNumber The new phone number
     * @param emailAddress The new email address
     */
    private synchronized void setContactDetails(String phoneNumber,
String emailAddress){
        try{
            //writes the changes to the files
            FileWriter writePhone = new
FileWriter(Server.CLUB_CONTACT_PHONE_NUMBER_FILE,false);
            BufferedWriter buffWritePhone = new
BufferedWriter(writePhone);
            buffWritePhone.write(phoneNumber);
            buffWritePhone.flush();writePhone.flush();buffWritePhone.clo
se();writePhone.close();//flushes and closes writers
            FileWriter writeEmail = new
FileWriter(Server.CLUB_CONTACT_EMAIL_FILE,false);
            BufferedWriter buffWriteEmail = new
BufferedWriter(writeEmail);
            buffWriteEmail.write(emailAddress);
            buffWriteEmail.flush();writeEmail.flush();buffWriteEmail.clo
se();writeEmail.close();//flushes and closes writers
            //updates the array
            SERVER.clubContactDetails[0]=phoneNumber;SERVER.clubContactD
etails[1]=emailAddress;
        }catch(IOException e){
            System.out.println("Possible error when set club contact
details: "+e);
        }
    }
    /**
     * Generates a new season of games for the chess club.
     */
    private synchronized void createNewSeason(){
        try{
            Fixture.createdNewFixturesFromStart(getAllClubMembers());//g
enerates the new season
        }catch(Exception e){
            System.out.println("Possible non fatal error when creating
new season: "+e);
        }
    }
    /**
     * Updates the fixtures and results.
```

```
     * @param fixturesAndResults The updated value of the fixtures and
results
     */
    private synchronized void
updateFixturesAndResults(String[][][]fixturesAndResults){
        try{
         Fixture.writeToFile(fixturesAndResults,
Fixture.getListOfPeopleToAddedDuringSeason());
        }catch(IOException e){
            System.out.println("Possibly non fatal error when updating
fixtures: "+e);
        }
    }
    /**
     * Returns all requests to join the chess club. This method returns
the user file record  of every user who has requested to join the club.
     * @return The requests to join the club
     */
    public synchronized String [][]getAllRequestsToJoinClub(){
        //searches through every user in the user file to see if they
have requested to join the club.
        String currentUser[];
        ArrayList<String[]> allRequests=new ArrayList<String[]>();
        for(int user=0;user<SERVER.userFile.length;user++){
            if(SERVER.userFile[user][16].equals("true")){
                currentUser=new
String[UserInfo.WIDTH_OF_USER_DETAILS_FILE];
                //copies array
                for(int x=0;x<currentUser.length;x++){
                    currentUser[x]=SERVER.userFile[user][x];
                }
                allRequests.add(currentUser);
            }
        }
        String[][]req = new
String[allRequests.size()][UserInfo.WIDTH_OF_USER_DETAILS_FILE];//conver
ts array list into a 2d array of requests
        for(int user=0;user<req.length;user++){
            for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                req[user][x]=allRequests.get(user)[x];
            }
        }
        return req;
    }
    /**
     * Gets the email address of the chosen user
     * @param userName The username
     * @return The email address as a string array of length 1. Element
nought (the only element of the array) stores the email address.
     */
    public synchronized String[]getEmailOfUser(String userName){
        String user []=new String[1];user[0]="";
        for(int
```

```
person=0;person<SERVER.emailLinkFile.length;person++){//searches
sequentially through file to find email address
            if(SERVER.emailLinkFile[person][1].equals(userName)){
                user[0]=SERVER.emailLinkFile[person][0];
                return user;
            }
        }
        return user;
    }
    /**
     * Sets the chosen user as the club leader and member.
     * @param userName The username
     * @throws java.lang.Exception An error may occur if the user
inputted is not found in the user records
     */
    public synchronized void setUserAsLeaderAndMember(String userName)
throws Exception{
        int index=Search.searchForRecordIndex(SERVER.userFile,
userName);//find user in user file
        if(index==Search.INDEX_NOT_FOUND){//if user is not found, an
appropriate error message is returned
            throw new Exception("User to be leader was not found.");
        }
        if(SERVER.userFile[index][14].equals("false")){//adds club
leader to fixtures if they are not already a member
            Fixture.addPlayerToFixtures(userName, getAllClubMembers());
        }
        //updates data
        //sets all users in file to not being the leader
        for(int user=0;user<SERVER.userFile.length;user++){
            SERVER.userFile[user][13]="false";
        }
        SERVER.userFile[index][16]="false";//sets user as a leader and
club member who is not requesting to join the club
        SERVER.userFile[index][13]="true";SERVER.userFile[index][14]="tr
ue";
        SERVER.writeArrayToFileWithSplit(SERVER.userFile,
Server.USER_FILE_NAME);//updates text file
    }
    /**
     * Gets the people added midway through the season to the club.
     * @return The list of people
     */
    public synchronized String[]getPeopleAddedMidwayThroughSeason(){
        try{
            return Fixture.getListOfPeopleToAddedDuringSeason();
        }catch(IOException e){
            return new String[0];
        }
    }
    /**
     * Gets all fixtures and results. Returns an empty string array if
an error occurs. An error may occur if the fixtures have not yet been
```

```
generated.
     * @return The results and fixtures
     */
    public synchronized String[][][]getAllFixturesAndResults(){
        try{
            return Fixture.loadFixturesArray();
        }catch(Exception e){
           return new String[0][0][0];
        }
    }
    /**
     * Gets all members of the chess club. This will be returned in
lexographical order (i.e., the array is sorted)
     * @return All members of the chess club
     */
    public synchronized String[]getAllClubMembers(){
        //loops through user file to find club members
        ArrayList<String> allMembers=new ArrayList<String>();
        for(int i=0;i<SERVER.userFile.length;i++){
            if(SERVER.userFile[i][14].equals("true")){
                allMembers.add(SERVER.userFile[i][0]);//adds member to
list
            }
        }
        return allMembers.toArray(new String[0]);//converts array list
to String array
    }
    /**
     * Gets the current league table
     * @return The league table
     */
    public synchronized String[][]getLeagueTable(){
        //table is in the form: username, wins, draws, losses, points,
isPartOfLeague
        String[]clubMembers=getAllClubMembers();//gets club members in a
sorted array
        final int WIDTH_OF_TABLE=6;
        String[][]leagueTable=new
String[clubMembers.length][WIDTH_OF_TABLE];
        int whiteIndex;
        int blackIndex;
        //intialises values of table
        for(int i=0;i<leagueTable.length;i++){
            leagueTable[i][0]=clubMembers[i];
            leagueTable[i][1]="0";leagueTable[i][2]="0";leagueTable[i][3
]="0";leagueTable[i][4]="0";
            leagueTable[i][5]="false";
        }
        try{
            String[][][]fixtures=Fixture.loadFixturesArray();//loads
fixture array
            //loops through all played games and updates array
accordingly. Calculating points and number of wins, draws and losses for
```

```
each player
            for(int round=0;round<fixtures.length;round++){
                for(int match=0;match<fixtures[0][0].length;match++){
                    if(!(fixtures[round][2][match].equals(Fixture.BYE_GA
ME_STATE) ||
fixtures[round][2][match].equals(Fixture.GAME_NOT_YET_PLAYED))){//if the
game has been played and is not a bye game, the league table is updated
                        whiteIndex=Search.searchForRecordIndex(leagueTab
le, fixtures[round][0][match]);//searches for white player in list of
players in club
                        blackIndex=Search.searchForRecordIndex(leagueTab
le, fixtures[round][1][match]);//searches for black player in list of
players in club
                        if(fixtures[round][2][match].equals(Fixture.GAME
_DRAW)){//indicates that both players have drawn the game
                            leagueTable[whiteIndex][2]=String.valueOf(In
teger.parseInt(leagueTable[whiteIndex][2])+1);
                            leagueTable[whiteIndex][4]=String.valueOf(In
teger.parseInt(leagueTable[whiteIndex][4])+1);
                            leagueTable[whiteIndex][5]="true";
                            leagueTable[blackIndex][2]=String.valueOf(In
teger.parseInt(leagueTable[blackIndex][2])+1);
                            leagueTable[blackIndex][4]=String.valueOf(In
teger.parseInt(leagueTable[blackIndex][4])+1);
                            leagueTable[blackIndex][5]="true";
                        }else
if(fixtures[round][2][match].equals(Fixture.GAME_WHITE_WIN)){//updates
league table for white win
                            leagueTable[whiteIndex][1]=String.valueOf(In
teger.parseInt(leagueTable[whiteIndex][1])+1);
                            leagueTable[whiteIndex][4]=String.valueOf(In
teger.parseInt(leagueTable[whiteIndex][4])+3);
                            leagueTable[whiteIndex][5]="true";
                            leagueTable[blackIndex][3]=String.valueOf(In
teger.parseInt(leagueTable[blackIndex][3])+1);
                            leagueTable[blackIndex][5]="true";
                        }else
if(fixtures[round][2][match].equals(Fixture.GAME_BLACK_WIN)){//updates
league table for black win
                            leagueTable[whiteIndex][3]=String.valueOf(In
teger.parseInt(leagueTable[whiteIndex][3])+1);
                            leagueTable[whiteIndex][5]="true";
                            leagueTable[blackIndex][1]=String.valueOf(In
teger.parseInt(leagueTable[blackIndex][1])+1);
                            leagueTable[blackIndex][4]=String.valueOf(In
teger.parseInt(leagueTable[blackIndex][4])+3);
                            leagueTable[blackIndex][5]="true";

                        }
                    }
                }
            }
            //sorts the league table in order of points
```

```
            String[][]pointsAndPointer=new
String[leagueTable.length][2];//this array stores the points of a user
and an index pointing to said user
            for(int i=0;i<pointsAndPointer.length;i++){
                pointsAndPointer[i][0]=leagueTable[i][4];
                pointsAndPointer[i][1]=String.valueOf(i);
            }
            //sorts array
            Sort.quickSortUsingIntAtFirstElement(pointsAndPointer);
            //league table is generated in points order. Those with the
least points are stored at index 0.
            String [][]newLeagueTable=new
String[leagueTable.length][WIDTH_OF_TABLE];
            int indexPointer;
            for(int i=0;i<leagueTable.length;i++){
                indexPointer=Integer.parseInt(pointsAndPointer[i][1]);
                for(int x=0;x<WIDTH_OF_TABLE;x++){
                    newLeagueTable[i][x]=leagueTable[indexPointer][x];
                }
            }
            //reverses array so player with most points is at top
            String temp;
            for(int i = 0; i < newLeagueTable.length / 2; i++){
                for(int x=0;x<WIDTH_OF_TABLE;x++){
                    temp=newLeagueTable[i][x];
                    newLeagueTable[i][x]=newLeagueTable[newLeagueTable.l
ength - i - 1][x];
                    newLeagueTable[newLeagueTable.length - i -
1][x]=temp;
                }
            }
            return newLeagueTable;//returns league table
        }catch(Exception e){
            System.out.println("Error when getting league table: "+e);
            return new String[0][0];
        }
    }
    /**
     * Removes the chosen user from the West Cross Chess Club
     * @param userName The username of the user
     */
    private synchronized  void removeUserFromChessClub(String userName){
        //searches for the user record in the user file
        int index= Search.searchForRecordIndex(SERVER.userFile,
userName);
        if(index==Search.INDEX_NOT_FOUND){
            return;
        }
        //updates the variable indicating whether the user is a club
member
        SERVER.userFile[index][14]="false";SERVER.userFile[index][13]="f
alse";
        //writes change to text file
```

```
        SERVER.writeArrayToFileWithSplit(SERVER.userFile,
Server.USER_FILE_NAME);
        //removes user from fixtures if approrpaite
        try{
            Fixture.removePlayerFromFixtures(userName);
        }catch(Exception e){
            System.out.println("Possible non fatal error when removing
user from chess club: "+e);
            return;
        }

    }
    /**
     * Changes status of request to join club
     * @param userName The name of the user
     */
    private synchronized void updateRequestToJoinClubStatus(String
userName){
        //attempts to find user in user file
        int index = Search.searchForRecordIndex(SERVER.userFile,
userName);
        if(index==Search.INDEX_NOT_FOUND){
            return;
        }
        //updates status
        if(SERVER.userFile[index][16].equals("true")){
            SERVER.userFile[index][16]="false";
        }else{
            SERVER.userFile[index][16]="true";
        }
        //writes array to text file
        SERVER.writeArrayToFileWithSplit(SERVER.userFile,
Server.USER_FILE_NAME);
    }
    /**
     * Searches for the game id and deletes it if is found. Note, this
method deletes the record so it must be ensured that both users have
actually seen the game results before the game is deleted
     * @param GAME_ID the game id to delete
     * @param RESULT The game result. 1 indicates that player one won,
0.5 indicates a draw and 0 indicates that player two won
     */
    private synchronized void deleteGame(final int GAME_ID,final double
RESULT){
        //checks to ensure that game has not already been deleted
        int playerOneElo;//stores the first player's ELO score
        int playerTwoElo;//stores player two's ELO score
        double resultPlayerOne=0.5;//stores the result from player one's
perspective
        double resultPlayerTwo=0.5;//stores the result from player two's
perspective
        double probabilityPlayerOneWins;//stores the probability that
player one wins
```

```
        double probabilityPlayerTwoWins;//stores the probability that
player two wins
        int
recordIndex=Search.searchForRecordIndexInt(SERVER.onlineGames,
GAME_ID);//gets index where record is located
        if(recordIndex==Search.INDEX_NOT_FOUND){
            return;//games has already been deleted and hence this
method is not needed
        }
        //if a game that has not yet been accepted is being deleted, one
of the username may be blank. In this case, no ELO update is needed
        if((SERVER.onlineGames[recordIndex][1]==null ||
SERVER.onlineGames[recordIndex][2]==null
||SERVER.onlineGames[recordIndex][1].equals("")||SERVER.onlineGames[reco
rdIndex][2].equals(""))==false){
            int playerOneRecord=Search.searchForRecordIndex(
SERVER.userFile, SERVER.onlineGames[recordIndex][1]);
            int playerTwoRecord=Search.searchForRecordIndex(
SERVER.userFile, SERVER.onlineGames[recordIndex][2]);
            if(playerOneRecord==Search.INDEX_NOT_FOUND||playerTwoRecord=
=Search.INDEX_NOT_FOUND){
                return;
            }
            //gets the current ELO ratings of the players
            playerOneElo=Integer.parseInt(SERVER.userFile[playerOneRecor
d][15]);
            playerTwoElo=Integer.parseInt(SERVER.userFile[playerTwoRecor
d][15]);
            //updates ELO values of users involved
            //gets score of game
            if(RESULT==1){
                resultPlayerTwo=0;
                resultPlayerOne=1;
            }else if(RESULT==0){
                resultPlayerTwo=1;
                resultPlayerOne=0;
            }
            //implements ELO algorithm
            probabilityPlayerOneWins=1.0/(1.0+Math.pow(10.0,
(playerTwoElo-playerOneElo)/400.0));
            probabilityPlayerTwoWins=1.0/(1.0+Math.pow(10.0,
(playerOneElo-playerTwoElo)/400.0));
            playerOneElo=(int)Math.round(playerOneElo+24.0*(resultPlayer
One-probabilityPlayerOneWins));
            playerTwoElo=(int)Math.round(playerTwoElo+24.0*(resultPlayer
Two-probabilityPlayerTwoWins));
            //updates elo results and writes it to array and file
            SERVER.userFile[playerOneRecord][15]=String.valueOf(playerOn
eElo);
            SERVER.userFile[playerTwoRecord][15]=String.valueOf(playerTw
oElo);
            SERVER.writeArrayToFileWithSplit(SERVER.userFile,
Server.USER_FILE_NAME);
```

```
        }
        //now the game details are deleted from the file
        String newOnlineGameFile[][]=new
String[SERVER.onlineGames.length-1][Server.WIDTH_OF_ONLINE_FILE];
        //copies the array to a new one, simply deleteing the old record
        for(int i=0;i<recordIndex;i++){
            for(int x=0;x<Server.WIDTH_OF_ONLINE_FILE;x++){
                newOnlineGameFile[i][x]=SERVER.onlineGames[i][x];
            }
        }
        for(int i=recordIndex+1;i<SERVER.onlineGames.length;i++){
            for(int x=0;x<Server.WIDTH_OF_ONLINE_FILE;x++){
                newOnlineGameFile[i-1][x]=SERVER.onlineGames[i][x];
            }
        }
        //sets the array to the updated file
        SERVER.onlineGames=newOnlineGameFile;
        //writes the new file to a text file
        SERVER.writeArrayToFileWithSplit(SERVER.onlineGames,
Server.ONLINE_GAMES_FILE);
    }
    /**
     * Updates a record in the online game file. If the record can't be
found, nothing happens.
     * @param newRecordValue The new record that should replace the
current one.
     */
    private synchronized void
updateOnlineGameRecord(String[]newRecordValue){
        int
indexOfRecord=Search.searchForRecordIndexInt(SERVER.onlineGames,
Integer.parseInt(newRecordValue[0]));//searches for record
        if(indexOfRecord==Search.INDEX_NOT_FOUND){//if the item is not
found, nothing happens
            return;
        }

        //updates record
        for(int
itemInFile=0;itemInFile<Server.WIDTH_OF_ONLINE_FILE;itemInFile++){
            SERVER.onlineGames[indexOfRecord][itemInFile]=newRecordValue
[itemInFile];
        }
        SERVER.writeArrayToFileWithSplit(SERVER.onlineGames,
Server.ONLINE_GAMES_FILE);//writes new array to file
        if(newRecordValue[9].equals("true")&&
newRecordValue[8].equals("true")){//deletes the game if appropraite
            deleteGame(Integer.parseInt(newRecordValue[0]),
Double.parseDouble(newRecordValue[11]));
            return;
        }
    }
    /**
```

```
     * Stops the queue thread and finishes all remaining tasks
     */
    public  void stopThread(){
        setKeepRunningFalse();
        while(getIsRunning()){//waits for the thread to finish
        }
        while(getPackages().length>0){//finishes remaining tasks queued
            handlePackage();
        }
    }
    /**
     * Adds a new account to the user file.
     * @param record The entire entry to be added to the user file
     * @param userName The username
     * @param emailAddress The email address
     */
    private synchronized void createAccount(String record[],String
userName,String emailAddress){
        if(Search.searchForRecordWithPrimaryKeyAtNoughtElement(SERVER.us
erFile,
userName,UserInfo.WIDTH_OF_USER_DETAILS_FILE)[0]!=null||Search.searchFor
RecordWithPrimaryKeyAtNoughtElement(SERVER.emailLinkFile,
emailAddress,UserInfo.WIDTH_OF_EMAIL_LINK_FILE)[0]!=null){//checks to
ensure account hasn't already been created
            return;
        }
        String [][]userNew = new
String[SERVER.userFile.length+1][UserInfo.WIDTH_OF_USER_DETAILS_FILE];
        for(int y=0;y<SERVER.userFile.length;y++){//copies old user file
to new array
            for(int x =0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){
                userNew[y][x]=SERVER.userFile[y][x];
            }
        }
        for(int x=0;x<UserInfo.WIDTH_OF_USER_DETAILS_FILE;x++){//adds
new record to new array
            userNew[userNew.length-1][x]=record[x];
        }
        Sort.mergeSort(userNew);//sorts the user array using merge sort
        String [][]emailLinkNew=new
String[SERVER.emailLinkFile.length+1][UserInfo.WIDTH_OF_EMAIL_LINK_FILE]
;
        for(int y =0;y<SERVER.emailLinkFile.length;y++){//copies current
email file
            for(int x =0;x<UserInfo.WIDTH_OF_EMAIL_LINK_FILE;x++){
                emailLinkNew[y][x]=SERVER.emailLinkFile[y][x];
            }
        }
        emailLinkNew[emailLinkNew.length-
1][0]=emailAddress;emailLinkNew[emailLinkNew.length-
1][1]=userName;//adds new record to file
        Sort.mergeSort(emailLinkNew);
        SERVER.writeArrayToFileWithSplit(userNew,
```

```
Server.USER_FILE_NAME);SERVER.writeArrayToFileWithSplit(emailLinkNew,
Server.EMAIL_FILE_NAME);
        SERVER.userFile=userNew;SERVER.emailLinkFile=emailLinkNew;
    }
    /**
     * Adds the input package to the queue
     * @param pack The input package to add to the queue
     */
    public synchronized  void addInputPackageTask(InputPackages pack){
        if(!getKeepThreadRunning()){//doesnt allow any more packages to
be added to the queue once the thread is closed
            return;
        }
        InputPackages newPack[]=new InputPackages[packages.length+1];
        //copies old packages to new array
        for(int i=0;i<packages.length;i++){
            newPack[i]=packages[i];
        }
        //adds most recent package to end of array
        newPack[packages.length]=pack;
        packages=newPack;
    }
    /**
     * Removes an InputPackage from the queue and returns it
     * @return The input package removed
     */
    private synchronized InputPackages removeFromQueue(){
        InputPackages[]newPack=new InputPackages[packages.length-1];
        for(int i=1;i<packages.length;i++){//creates new array
            newPack[i-1]=packages[i];
        }
        InputPackages pack=packages[0];
        packages=newPack;
        return pack;
    }
    /**
     * Creates a new online game and adds it to the system
     * @param opponentOne The name of the opponent
     * @param opponentTwo The name of the other opponent - leave this
blank if the opponent name is not yet known
     * @param isUnlimitedTime Whether the game uses unlimited time
control
     * @param isTenMins Whether the game uses 10 minute time control
     * @param isGameForFriendCode If the game to be created required a
friend code to be enterer
     * @param isPlayerOneWhite Stores whether player one is white or not
     * @return the game id
     */
    private synchronized String createNewOnlineGame(String opponentOne,
String opponentTwo, boolean isUnlimitedTime, boolean isTenMins,boolean
isGameForFriendCode,boolean isPlayerOneWhite){
        //file is stored: gameId, opponentOne, opponentTwo, gameDetails
(FENS(s)), timePlayed, timeLeftWhite, timeLeftBlack, is a friend code
```

```
needed, has playerOne seen, hasPlayerTwoseen win, isPlayerOneWhite,
gameState, drawP1?,drawp2?
        String[] record=new String[Server.WIDTH_OF_ONLINE_FILE];//stores
the record to write to the game file
        String gameId="";//stores the gameId
        record[1]=opponentOne;record[2]=opponentTwo;record[3]=Server.STA
RTING_BOARD_FEN;record[4]="0";record[7]=String.valueOf(isGameForFriendCo
de);
        record[8]="false";record[9]="false";record[10]=String.valueOf(is
PlayerOneWhite);record[11]="-1";record[12]="false";record[13]="false";
        //sets time left for each player
        if(isUnlimitedTime){
            record[5]="-100";record[6]="-100";
        }else if(isTenMins){
            record[5]="600";record[6]="600";
        }else{
            record[5]="300";record[6]="300";
        }
        if(SERVER.onlineGames.length==0){//the first game to be added is
done like this
            SERVER.onlineGames=new
String[1][Server.WIDTH_OF_ONLINE_FILE];
            gameId="000000";
            record[0]=gameId;
            for(int x=0;x<Server.WIDTH_OF_ONLINE_FILE;x++){
                SERVER.onlineGames[0][x]=record[x];
            }
        }else{
            //gets the game id by looping sequentially through the file
and finding the next available integer game id
            //as the file is sorted, if the end of the file is reached,
the game is simply the current id plus one
            boolean hasFileEndBeenReachedWithoutId=true;
            for(int i=0;i<SERVER.onlineGames.length;i++){
                if(Integer.parseInt(SERVER.onlineGames[i][0])!=i){
                    //ensures that the integer is not less than six
characters
                    gameId=String.valueOf(i);
                    while(gameId.length()<6){
                        gameId="0"+gameId;
                    }
                    record[0]=gameId;
                    hasFileEndBeenReachedWithoutId=false;
                    break;
                }
            }
            if(hasFileEndBeenReachedWithoutId){//if the end of the file
is reached without an id being assigned this code runs
                gameId=String.valueOf(SERVER.onlineGames.length);
                while(gameId.length()<6){
                    gameId="0"+gameId;
                }
                record[0]=gameId;
```

```
            }
            //updates onlineGames Array
            String newOnlineGames[][]=new
String[SERVER.onlineGames.length+1][Server.WIDTH_OF_ONLINE_FILE];
            //copies old file to new one
            for(int i=0;i<SERVER.onlineGames.length;i++){
                for(int x=0;x<Server.WIDTH_OF_ONLINE_FILE;x++){
                    newOnlineGames[i][x]=SERVER.onlineGames[i][x];
                }
            }
            //adds new record
            for(int x=0;x<Server.WIDTH_OF_ONLINE_FILE;x++){
                newOnlineGames[SERVER.onlineGames.length][x]=record[x];
            }
            //sort file
            Sort.quickSortUsingIntAtFirstElement(newOnlineGames);
            SERVER.onlineGames=newOnlineGames;
        }
        SERVER.writeArrayToFileWithSplit(SERVER.onlineGames,
Server.ONLINE_GAMES_FILE);//updates the text file
        return gameId;//returns the game id
    }
    /**
     * Handles the request to play a new random online game.
     * @param userName The username of the player making the request
     */
    private synchronized void handleRequestForNewOnlineRandomGame(String
userName){
        //checks to see if any requests for online random games exsist -
if they do then the user is assigned to one of these requests
        for(int y=0;y<SERVER.onlineGames.length;y++){
            if(SERVER.onlineGames[y][7].equals("false")&&SERVER.onlineGa
mes[y][2].equals("")&&!SERVER.onlineGames[y][1].equals(userName)){//chec
ks to see if a random game has already been created but not accpeted by
another player
                SERVER.onlineGames[y][2]=userName;
                SERVER.writeArrayToFileWithSplit(SERVER.onlineGames,
Server.ONLINE_GAMES_FILE);//updates the text file
                return;
            }
        }
        //if no existing play request is found, a new one must be
created
        Random rnd = new Random();
        int randNum=rnd.nextInt(3);
        boolean isPlayerOneWhite=Math.random()>=0.5;
        //a random time control is chosen for the new game
        switch (randNum) {
            case 0:
                createNewOnlineGame(userName, "", true, false,
false,isPlayerOneWhite);
                break;
            case 1:
```

```
                createNewOnlineGame(userName, "", false, true,
false,isPlayerOneWhite);
                break;
            default:
                createNewOnlineGame(userName, "", false, false,
false,isPlayerOneWhite);
                break;
        }

    }
    /**
     * Creates a new online game against a friend, checking that the
game can legally be created
     * @param playerOne The username of player one
     * @param playerTwo The username of player two -"" should be input
if the game code is to be shared
     * @param timeControl The time control used
     * @return The game code. "-1" is returned if the game can't be
created
     */
    public synchronized String createOnlineGameAgainstFriend(String
playerOne,String playerTwo,String timeControl){
        //checks to ensure that game can legally be created
        if(!getAllGamesWithUser(playerOne)[2][0].equals("")
||(!playerTwo.equals("")&&!getAllGamesWithUser(playerTwo)[2][0].equals("
")){
            return "-1";
        }
        //adds game to file
        switch (timeControl) {
            case "-100":
                return(createNewOnlineGame(playerOne, playerTwo, true,
false, playerTwo.equals(""), Math.random()>=0.5));
            case "500":
                return(createNewOnlineGame(playerOne, playerTwo, false,
true, playerTwo.equals(""), Math.random()>=0.5));
            default:
                return(createNewOnlineGame(playerOne, playerTwo, false,
false, playerTwo.equals(""), Math.random()>=0.5));
        }
    }
    /**
     * Attempts to join an existing game using the code
     * @param userName The username of the player attempting to join
     * @param gameCode The game code
     * @return Whether the game was joined
     */
    public synchronized boolean joinGameUsingFriendCode(String
userName,int gameCode){
        int
recordIndex=Search.searchForRecordIndexInt(SERVER.onlineGames,
gameCode);//attempts to locate game
        if(recordIndex==Search.INDEX_NOT_FOUND){
```

```
            return false;
        }
        if(SERVER.onlineGames[recordIndex][1].equals(userName)){//ensure
s that user can't join a game against themselves
            return false;
        }
        if(SERVER.onlineGames[recordIndex][7].equals("false")){//prevent
s the user from joining a random game
            return false;
        }
        //allows the user to join the game
        SERVER.onlineGames[recordIndex][2]=userName;
        SERVER.writeArrayToFileWithSplit(SERVER.onlineGames,
Server.ONLINE_GAMES_FILE);
        return true;
    }
    /**
     * Gets the game data requested
     * @param gameCode the game code
     * @return The data
     */
    public synchronized String []getGameRecord(int gameCode){
        String []record=new String[Server.WIDTH_OF_ONLINE_FILE];
        int index=Search.searchForRecordIndexInt(SERVER.onlineGames,
gameCode);
        if(index==-1){
            return record;
        }
        //copies record to new array
        for(int i=0;i<record.length;i++){
            record[i]=SERVER.onlineGames[index][i];
        }
        return record;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

/**
 * This class is used to search through data
 * @author mortimer
 */
public class Search {
```

```
    public static final int INDEX_NOT_FOUND=-1;
    /**
     * Performs a binary search on the one d array. Assumes it is sorted
lexographically.
     * @param DATA The data sorted lexographically
     * @param ITEM_TO_FIND the data item to be found
     * @return true is returned if the record is in the array, otherwise
false is returned
     */
    public static boolean isRecordInArray(String[]DATA,String
ITEM_TO_FIND){
        if(DATA.length==1){
            if(DATA[0].equals(ITEM_TO_FIND)){
                return true;
            }else{
                return false;
            }
        }
        if(DATA.length==0){
            return false;
        }
        String record="";
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int comparison;
        String primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=DATA[mid];
            //if pk precedes itemToFind less than 0
            comparison = primaryKeyBuffer.compareTo(ITEM_TO_FIND);
            if(comparison==0){
                return true;
            }else if(comparison>0){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return false;
    }
    /**
     * Performs a binary search on the 2d array inputted. It searches
for the itemToFind. It assumes this is a primary key at the nought
element
     * of every row. It uses binary search and the java String method
.compareTo(). Hence the data must be sorted by the primary key
     * with data of a small lexographic value at the top (i.e. sorted in
ascending order).
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @param WIDTH_OF_DATA The width of a record
```

```
     * @return the field containing the item to find. If no item is
found, this will be a null array of width equal to the width of a
record.
     */
    public static
String[]searchForRecordWithPrimaryKeyAtNoughtElement(final
String[][]DATA,final String ITEM_TO_FIND,final int WIDTH_OF_DATA){
        String record[]= new String[WIDTH_OF_DATA];
        if(DATA.length==0){
            return record;
        }
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int comparison;
        String primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=DATA[mid][0];
            //if pk precedes itemToFind less than 0
            comparison = primaryKeyBuffer.compareTo(ITEM_TO_FIND);
            if(comparison==0){
                for(int element=0;element<record.length;element++){
                    record[element]=DATA[mid][element];
                }
                break;
            }else if(comparison>0){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return record;
    }
     /**
     * Performs a binary search on the 2d array inputted. It searches
for the itemToFind. It assumes this is a primary key at the nought
element
     * of every row. It uses binary search and assumes that the nought
element of every row is an integer. It asssumes that data is sorted by
integer
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @param WIDTH_OF_DATA The width of one record
     * @return the field containing the item to find. If no item is
found, this will be a null array of width equal to the width of a
record.
     */
    public static String[]
searchForRecordWithPrimaryKeyAtNoughtElementInt(final
String[][]DATA,final int ITEM_TO_FIND, final int WIDTH_OF_DATA){
        String record[]= new String[WIDTH_OF_DATA];
        if(DATA.length==0){
```

```
            return record;
        }
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=Integer.parseInt(DATA[mid][0]);
            if(primaryKeyBuffer==ITEM_TO_FIND){
                for(int element=0;element<record.length;element++){
                    record[element]=DATA[mid][element];
                }
                break;
            }else if(primaryKeyBuffer>ITEM_TO_FIND){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return record;
    }
    /**
     * Performs a binary search on the 2d array inputted. It searches
for the itemToFind. It assumes this is a primary key at the nought
element
     * of every row. It uses binary search and assumes that the nought
element of every row is an integer. It assumes that data is sorted by
integer
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @return The row number is returned if not, -1 is returned
     */
    public static int searchForRecordIndexInt(final String[][]DATA,final
int ITEM_TO_FIND){
        if(DATA.length==0){
            return INDEX_NOT_FOUND;
        }
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=Integer.parseInt(DATA[mid][0]);
            if(primaryKeyBuffer==ITEM_TO_FIND){
                return mid;
            }else if(primaryKeyBuffer>ITEM_TO_FIND){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
```

```
        return INDEX_NOT_FOUND;
    }
    /**
     * Performs a binary search on the array inputted. It searches for
the itemToFind.
     * It uses binary search and assumes that the data is
lexographically sorted.
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @return The row number is returned if not, -1 is returned
     */
    public static int searchForRecordIndex(final String[]DATA,final
String ITEM_TO_FIND){
        if(DATA.length==0){
            return INDEX_NOT_FOUND;
        }
        int first = 0;
        int last = DATA.length-1;
        int mid;
        int comparison;
        String primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=DATA[mid];
            //if pk precedes itemToFind less than 0
            comparison = primaryKeyBuffer.compareTo(ITEM_TO_FIND);
            if(comparison==0){
                return mid;
            }else if(comparison>0){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return INDEX_NOT_FOUND;
    }
    /**
     * Performs a binary search on the 2d array inputted. It searches
for the itemToFind. It assumes this is a primary key at the nought
element
     * of every row. It uses binary search and assumes that the data is
lexographically sorted.
     * @param DATA The data to be searched through.
     * @param ITEM_TO_FIND The item to find
     * @return The row number is returned if not, -1 is returned
     */
    public static int searchForRecordIndex(final String[][]DATA,final
String ITEM_TO_FIND){
        if(DATA.length==0){
            return INDEX_NOT_FOUND;
        }
        int first = 0;
        int last = DATA.length-1;
```

```
        int mid;
        int comparison;
        String primaryKeyBuffer;
        do{
            mid = (first+last)/2;
            primaryKeyBuffer=DATA[mid][0];
            //if pk precedes itemToFind less than 0
            comparison = primaryKeyBuffer.compareTo(ITEM_TO_FIND);
            if(comparison==0){
                return mid;
            }else if(comparison>0){
                last = mid-1;
            }else{
                first = mid + 1;
            }
        }while(last>=first);
        return INDEX_NOT_FOUND;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import javax.swing.Timer;
import philipm.cs5.software.development.user.system.InputPackages;

/**
 * This class stores the server that enables communication with any
number of clients.
 * @author mortimer
 */
// class that performs server side commands. This class is a thread to
allow for the serverUI form to be used whilst the server is active
(hence it implements runnable)
```

```
public class Server implements Runnable{
    public static final String
STARTING_BOARD_FEN="rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq -
0 1";//stores the string that represents a new chess board
    private final int PORT_NO;//stores the port number for the server to
map too
    public String errorMessages[];//stores all errors that have occured
in an array. If the length of this array is not equal to zero, some form
of error has occured.
    public String[][]userFile;//stores the user text file contents as an
arry to speed up file access speed
    public String[][]emailLinkFile;///stores the email link file as an
array to speed up file access speed
    public String[][]onlineGames;//stores online games
    public static final String USER_FILE_NAME="userDetails.txt";//stores
the file name of the user details text file
    public static final String SPLIT_ITEM=UserInfo.SPLIT_ITEM;
    public static final String EMAIL_FILE_NAME="emailLinkFile.txt";
    public static final int WIDTH_OF_ONLINE_FILE=14;//stores the number
of fields in a record in the online file: game code, opp1,opp2, FEN(s),
time elapsed, time White, time Black, boolean hasPlayerOneSeenGameOver,
hasPlayerTwoSeenGameOver, isPlayerOneWhite, gameState,
drawPlayerOne?,drawPlayerTwo?
    public static final String GUEST_ID=UserInfo.GUEST_ID;
    public static final String GUEST_EMAIL=UserInfo.GUEST_ID_EMAIL;
    public static final String
GUEST_RECORD_USER_FILE=GUEST_ID+SPLIT_ITEM+SPLIT_ITEM+SPLIT_ITEM+SPLIT_I
TEM+SPLIT_ITEM+SPLIT_ITEM+SPLIT_ITEM+SPLIT_ITEM+SPLIT_ITEM+SPLIT_ITEM+SP
LIT_ITEM+SPLIT_ITEM+SPLIT_ITEM+SPLIT_ITEM+SPLIT_ITEM+SPLIT_ITEM;
    private ServerSocket serverSocket;
    private Timer timer;
    public static final String
ONLINE_GAMES_FILE="onlineGames.txt";//stores the file name of the online
games
    public Queue queue;//stores the queue used when writing to text
files
    private Thread queueThread;//stores the thread which the queue uses
    public static final String
CLUB_CONTACT_PHONE_NUMBER_FILE="phoneNumber.txt";
    public static final String
CLUB_CONTACT_EMAIL_FILE="emailClubContact.txt";
    public String[]clubContactDetails;//stores the club's contact
details
    /**
     * Constructor method initialises appropriate constants and values.
     * @param errorMessArray The array which stores the error messages.
This error should be of length zero but will be set to length zero
regardless.
     * @throws Exception If an error occurred, an exception is throw.
This exception contains a String message that can be displayed.
     */
    public Server(String[] errorMessArray ) throws Exception{
        errorMessArray=new String[0];
```

```
        errorMessages=errorMessArray;
        try{
            //gets the port number for the server to run on
            FileReader read = new
FileReader(ChangeServerDetails.SERVER_SETTINGS_TEXT_FILE_NAME);
            BufferedReader buffRead = new BufferedReader(read);
            String portLine=buffRead.readLine();
            buffRead.close();read.close();
            try{
                PORT_NO = Integer.parseInt(portLine);
            }catch(NumberFormatException ex){
                throw new Exception("Port Number stored is not a valid
integer. Please change the port number.");
            }
        }catch(IOException e){
            throw new Exception("An error occured when attempting to
retrieve the port number from the file
"+ChangeServerDetails.SERVER_SETTINGS_TEXT_FILE_NAME+" Error
details"+e);
        }
        //loads text files into arrays for faster access
        emailLinkFile=loadFileToArray(UserInfo.WIDTH_OF_EMAIL_LINK_FILE,
EMAIL_FILE_NAME);
        userFile=loadFileToArray(UserInfo.WIDTH_OF_USER_DETAILS_FILE,
USER_FILE_NAME);
        onlineGames=loadFileToArray(WIDTH_OF_ONLINE_FILE,
ONLINE_GAMES_FILE);
        queue = new Queue(this);//creates new queue
        queueThread = new Thread(queue);queueThread.start();//starts the
thread for the queue
        clubContactDetails=getClubContactDataFromFile();//loads club
contact details
    }
    /**
     * Gets the club contact details from file
     * @return The contact details
     */
    private String[] getClubContactDataFromFile(){
        String details[]=new String[2];details[0]="";details[1]="";
        try{
            FileReader readPhone = new
FileReader(CLUB_CONTACT_PHONE_NUMBER_FILE);//read the phone number from
the file
            BufferedReader buffReadPhone = new
BufferedReader(readPhone);
            String fileCont=buffReadPhone.readLine();
            if(fileCont!=null){
                details[0]=fileCont;
            }
            buffReadPhone.close();readPhone.close();
            FileReader readEmail = new
FileReader(CLUB_CONTACT_EMAIL_FILE);//read the email address from the
file
```

```
            BufferedReader buffReadEmail = new
BufferedReader(readEmail);
            fileCont=buffReadEmail.readLine();
            if(fileCont!=null){
                details[1]=fileCont;
            }
            buffReadEmail.close();readEmail.close();
        }catch(IOException e){
            System.out.println("Error with club contact files: "+e);
        }
        return details;

    }
    /**
     * This method is used to add the guest id to the user file.
     * This method is no longer used as Guest records are not stored.
This has been kept to make the code maintainable and easy to expand.
     */
    private void addGuestIdToFilesIfAppropriate(){
        try{
            //checks to see if GuestId is already in user file
            FileReader r = new FileReader(USER_FILE_NAME);
            BufferedReader buffUser = new BufferedReader(r);
            String line=buffUser.readLine();
            buffUser.close();r.close();
            if(line==null){//this will be true if the file is empty
(i.e. no guest)
                FileWriter write = new FileWriter(USER_FILE_NAME,false);
                BufferedWriter buffWrite = new BufferedWriter(write);
                buffWrite.write(GUEST_RECORD_USER_FILE);//adds guest
user file record to file
                buffWrite.flush();write.flush();buffWrite.close();write.
close();
            }
            //checks to see if GuestId is alraedy in email file
            FileReader rEmail = new FileReader(EMAIL_FILE_NAME);
            BufferedReader buffEmail = new BufferedReader(rEmail);
            line = buffEmail.readLine();
            buffEmail.close();rEmail.close();
            if(line==null){//this will be true if the file is empty
(i.e. no guest)
                FileWriter write = new
FileWriter(EMAIL_FILE_NAME,false);
                BufferedWriter buffWrite = new BufferedWriter(write);
                buffWrite.write(GUEST_EMAIL+SPLIT_ITEM+GUEST_ID);//adds
guest user file record to file
                buffWrite.flush();write.flush();buffWrite.close();write.
close();
            }
        }catch(IOException e){
            addErrorToErrorMessages("An error occurred when adding guest
details: "+e.getMessage());
        }
```

```
    }
    /**
     * Reads a text file and converts into a 2d array.
     * @param widthOfFile The number of fields in the file
     * @param fileName The fileName
     * @return The array that represents the data stored in the file
     */
    private String[][]loadFileToArray(int widthOfFile,String fileName){
        try{
            FileReader read = new FileReader(fileName);
            BufferedReader buffRead = new BufferedReader(read);
            int length =0;
            while(buffRead.readLine()!=null){//caclaulates number of
records in file
                length++;
            }
            buffRead.close();read.close();
            String newFile[][]=new String[length][widthOfFile];
            for(int i=0;i<newFile.length;i++){
                for(int j=0;j<widthOfFile;j++){
                    newFile[i][j]="";//intialises all values to "" to
avoid problems that can arise with null values
                }
            }
            FileReader readTwo = new FileReader(fileName);
            BufferedReader buffTwo = new BufferedReader(readTwo);
            String line;int i=0;
            String[]lineArr;
            //loads text file to array
            while((line=buffTwo.readLine())!=null){
                lineArr=line.split(SPLIT_ITEM);
                for(int j=0;j<lineArr.length;j++){
                    newFile[i][j]=lineArr[j];
                    if(lineArr[j]==null){
                        newFile[i][j]="";
                    }
                }
                i++;
            }buffTwo.close();readTwo.close();
            return newFile;
        }catch(IOException e){
            addErrorToErrorMessages("An error occured when accessing
"+fileName+" Error: "+e.getMessage());//adds error to display
        }
        return new String[0][0];
    }
    /**
     * This thread runs and creates a new thread for every client
communication. This allows for any number of clients to communicate with
the server
     * simultaneously.
     */
    @Override
```

```
    public void run() {
        try{
            serverSocket = new ServerSocket(PORT_NO);//creates the
server
            while(true){//continuosly checks for new connections
                Socket clientSocket = serverSocket.accept();//accepts
nay client communications
                //creates a new thread that reads the client input and
performs an approriate action accoridngly. This involves read and write
objects.
                Thread thread = new Thread(){
                    @Override
                    public void run(){
                        InputPackages message=null;
                        try{
                            ObjectOutputStream objOut = new
ObjectOutputStream(clientSocket.getOutputStream());//gets the output
stream incase a value needs to be returned to the client
                            ObjectInputStream objIn = new
ObjectInputStream(clientSocket.getInputStream()); //gets the input of
client
                            try {
                                message=(InputPackages)objIn.readObject(
);//gets the InputPackage
                                String command =
message.getActionName();
                                //checks to see the client message and
perform various actions depending on the contents
                                switch (command) {
                                    case
InputPackages.GET_USER_INFO_WITH_USERNAME_AND_PASSWORD:
                                        {
                                            UserInfo ret
=queue.getUserDetailsAndTryLogInWithUsername(message.getPackageToSend()[
1],message.getPackageToSend()[2]);
                                            objOut.writeObject(ret);
                                            break;
                                        }
                                    case
InputPackages.GET_USER_INFO_WITH_EMAIL_AND_PASSWORD:
                                        {
                                            UserInfo ret =
queue.getEmailAddressAndTryLogIn(message.getPackageToSend()[1],message.g
etPackageToSend()[2]);
                                            objOut.writeObject(ret);
                                            break;
                                        }
                                    case InputPackages.CREATE_ACCOUNT:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.GET_ONLINE_GAMES_FOR_USER:
```

```
                                        objOut.writeObject(queue.getAllG
amesWithUser(message.getPackageToSend()[1]));//returns all online games
with user
                                        break;
                                    case
InputPackages.CREATE_RANDOM_GAME:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case InputPackages.DELETE_GAME:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.CREATE_NEW_GAME_AGAINST_FRIEND:
                                        objOut.writeObject(queue.createO
nlineGameAgainstFriend(message.getPackageToSend()[1],
message.getPackageToSend()[2], message.getPackageToSend()[3]));
                                        break;
                                    case
InputPackages.JOIN_GAME_USING_GAME_CODE:
                                        objOut.writeObject(queue.joinGam
eUsingFriendCode(message.getPackageToSend()[2],Integer.parseInt(message.
getPackageToSend()[1])));
                                        break;
                                    case InputPackages.GET_GAME_RECORD:
                                        objOut.writeObject(queue.getGame
Record(Integer.parseInt(message.getPackageToSend()[1])));
                                        break;
                                    case
InputPackages.UPDATED_ONLINE_GAME_RECORD:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.GET_CLUB_CONTACT_DETAILS:
                                        objOut.writeObject(queue.getClub
ContactInfo());
                                        break;
                                    case
InputPackages.GET_WHETHER_USER_HAS_REQUESTED_TO_JOIN_CLUB:
                                        objOut.writeBoolean(queue.hasUse
rRequestedToJoinClub(message.getPackageToSend()[1]));
                                        break;
                                    case
InputPackages.CHANGE_STATUS_REQUEST_TO_JOIN_CLUB:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.REMOVE_USER_FROM_CHESS_CLUB:
                                        queue.addInputPackageTask(messag
e);
```

```
                                        break;
                                    case InputPackages.GET_LEAGUE_TABLE:
                                        objOut.writeObject(queue.getLeag
ueTable());
                                        break;
                                    case
InputPackages.GET_ALL_CLUB_MEMBERS:
                                        objOut.writeObject(queue.getAllC
lubMembers());
                                        break;
                                    case
InputPackages.GET_FIXTURES_AND_RESULTS:
                                        objOut.writeObject(queue.getAllF
ixturesAndResults());
                                        break;
                                    case
InputPackages.GET_PEOPLE_ADDED_MIDWAY_THROUGH_SEASON:
                                        objOut.writeObject(queue.getPeop
leAddedMidwayThroughSeason());
                                        break;
                                    case
InputPackages.UPDATED_FIXTURES_AND_RESULTS:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.CREATE_NEW_SEASON:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.SET_CONTACT_DETAILS:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.SET_USER_AS_LEADER:
                                        try{
                                            queue.setUserAsLeaderAndMemb
er(message.getPackageToSend()[1]);
                                        }catch(Exception e){
                                            System.out.println("Error
with setting new leader: "+e);
                                        }   break;
                                    case InputPackages.GET_ALL_REQUESTS:
                                        objOut.writeObject(queue.getAllR
equestsToJoinClub());
                                        break;
                                    case
InputPackages.GET_EMAIL_OF_USER:
                                        objOut.writeObject(queue.getEmai
lOfUser(message.getPackageToSend()[1]));
                                        break;
```

```
                                    case
InputPackages.REMOVE_REQUEST_JOIN_CLUB:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case InputPackages.ADD_USER_TO_CLUB:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.GET_CLUB_MEMBER_DETAILS_ALL_MEMBERS:
                                        objOut.writeObject(queue.getAllC
lubMemberDetails());
                                        break;
                                    case
InputPackages.UPDATE_USER_INFO_FILE_RECORD:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.REMOVE_NON_MEMBER_USER:
                                        queue.addInputPackageTask(messag
e);
                                        break;
                                    case
InputPackages.GET_ALL_NON_MEMBERS:
                                        objOut.writeObject(queue.getAllN
onMemberDetails());
                                        break;
                                    default:
                                        break;
                                }
                            } catch (ClassNotFoundException ex) {
                                addErrorToErrorMessages("There has been
an ClassNotFoundException with the server, specifically the object
stream. Error details: "+ex);
                                objOut.flush();objOut.close();objIn.clos
e();
                                return;
                            }
                            objOut.flush();objOut.close();//closes and
flushes approriate streams before terminating the thread
                            objIn.close();
                        }catch(IOException e){
                            System.out.println("Error that may not be
fatal: "+e);
                            try{
                                System.out.println(message.getActionName
());
                            }catch(Exception ez){
                                System.out.println(ez);
                            }
                            //addErrorToErrorMessages("There has been an
```

```
IOException with the server, specifically the object stream. Error
details: "+e);
                            return;
                        }
                    }
                };
                thread.start(); //starts the thread
            }
        }catch(IOException e){
            addErrorToErrorMessages("There has been an IOException with
the server. Error details: "+e);
            return;
        }
    }
    /**
     * Writes the array to the specified text file, with each field
being split with an appropriate character.
     * @param array The 2d string array
     * @param fileName The file name
     */
    public void writeArrayToFileWithSplit(String[][]array,String
fileName){
        try{
            FileWriter write = new FileWriter(fileName,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            String line;
            for(int y=0;y<array.length;y++){//loops through every line
in the array and seperates each filed with a seperation character
                line="";
                for(int x=0;x<array[0].length;x++){
                    if(x==0){
                        if(array[y][x]!=null){//null values are not
written to the file
                            line =array[y][x];
                        }
                    }else{
                        if(array[y][x]!=null){
                            line = line +SPLIT_ITEM+array[y][x];
                        }else{
                            line = line +SPLIT_ITEM;
                        }
                    }
                }
                //the appropriate line of text is written to the file
                if(y!=0){
                    buffWrite.newLine();
                }
                buffWrite.write(line);
            }
            //closes and flushes writers
            buffWrite.flush();write.flush();
            buffWrite.close();write.close();
        }catch(IOException e){
```

```
            addErrorToErrorMessages("Exception when writing to file
"+fileName+" Error "+e.getMessage());
        }
    }
    /**
     * Closes the server socket and stops the thread
     * @throws IOException An IOException may occur
     */
    public void closeServerSocket() throws IOException{
        queue.stopThread();
        serverSocket.close();
    }
    /**
     * Checks to see if any error have occurred in the program
     * @return True is returned if the error messages array length is
not equal to one. Otherwise false is returned
     */
    public boolean haveAnyErrorsOccured(){
        return errorMessages.length!=0;
    }
    /**
     * Gets all error messages.
     * @return The error message. Each element in the array represents
an error message.
     */
    public String[]getErrorMessages(){
        return errorMessages;
    }
    /**
     * Adds a new error message to the array that stores all error
messages. Also sets the variable hasErrorOccuredToTrue
     * @param newError the new error message to be appended
     */
    private void addErrorToErrorMessages(String newError){
        String updatedArray[] = new String[errorMessages.length+1];
        for(int i=0;i<errorMessages.length;i++){
            updatedArray[i]=errorMessages[i];
        }
        updatedArray[errorMessages.length]=newError;
        errorMessages=updatedArray;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
```

```
 */
package philipm.cs5.software.development.server.system;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import javax.swing.JOptionPane;
import javax.swing.Timer;

/**
 * Creates a form that is used to run the server
 * @author mortimer
 */
public class ServerUI extends javax.swing.JFrame{
    /**
     * Creates new form ServerUI
     */
    private Server serv;//stores the server object
    private Thread serverThread=null;//stores the thread used to execute
the server on
    private String[]errorMessagesArray;
    private boolean isServerRunning=false;
    Timer timer;//stores a timer object that periodically checks to see
if any errors have occured for the server
    public ServerUI() {
        initComponents();
        updateCompontentsBasedOnServerStatus();
    }
    /**
     * Alters the look of several components depending on whether the
server is currently online
     */
    private void updateCompontentsBasedOnServerStatus(){
        if(isServerRunning){
            serverStatusColourLbl.setBackground(Color.green);
            serverStatusLbl.setText("Server IS Running");
            changeServerStateBtn.setText("Turn Server Off");
            appointLeaderBtn.setEnabled(true);appointLeaderBtn.setVisibl
e(true);
        }else{
            serverStatusColourLbl.setBackground(Color.red);
            serverStatusLbl.setText("Server IS NOT Running");
            changeServerStateBtn.setText("Turn Server On");
            appointLeaderBtn.setEnabled(false);appointLeaderBtn.setVisib
le(false);
        }
    }
    /**
     * Adds the text message to the chat along with the date and time
when the message was added.
```

```
     * @param message The message to be added
     */
    private void addMessageToTextArea(String message){
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd
HH:mm:ss");
        LocalDateTime now = LocalDateTime.now();
        String currentText=serverInfoTxt.getText();
        //adds the message and time of the message
        if(currentText.equals("")){
            serverInfoTxt.setText(dtf.format(now)+" - "+message);
        }else{
            serverInfoTxt.setText(currentText+System.lineSeparator()+dtf
.format(now)+" - "+message);
        }
    }
    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        changeServerStateBtn = new javax.swing.JButton();
        settingsBtn = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        serverInfoTxt = new javax.swing.JTextArea();
        instructionsBtn = new javax.swing.JButton();
        serverStatusLbl = new javax.swing.JLabel();
        serverStatusColourLbl = new javax.swing.JLabel();
        appointLeaderBtn = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLO
SE);
        setTitle("Server");

        changeServerStateBtn.setText("Turn Server On");
        changeServerStateBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                changeServerStateBtnActionPerformed(evt);
            }
        });

        settingsBtn.setText("Change Port Number");
        settingsBtn.addActionListener(new
java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                settingsBtnActionPerformed(evt);
            }
        });

        serverInfoTxt.setColumns(20);
        serverInfoTxt.setRows(5);
        jScrollPane1.setViewportView(serverInfoTxt);

        instructionsBtn.setText("Instructions");
        instructionsBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                instructionsBtnActionPerformed(evt);
            }
        });

        serverStatusLbl.setText("Server IS NOT Running");

        serverStatusColourLbl.setForeground(new java.awt.Color(255, 0,
0));
        serverStatusColourLbl.setOpaque(true);

        appointLeaderBtn.setText("Set Club Leader");
        appointLeaderBtn.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                appointLeaderBtnActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addContainerGap()
                                .addComponent(settingsBtn)
                                .addGap(259, 259, 259)
```

```
                                .addComponent(instructionsBtn)
                                .addGap(235, 235, 235)
                                .addComponent(appointLeaderBtn))
                            .addGroup(jPanel1Layout.createSequentialGrou
p()
                                .addGap(145, 145, 145)
                                .addComponent(changeServerStateBtn)
                                .addGap(370, 370, 370)
                                .addComponent(serverStatusLbl)
                                .addGap(76, 76, 76)
                                .addComponent(serverStatusColourLbl,
javax.swing.GroupLayout.PREFERRED_SIZE, 129,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(0, 111, Short.MAX_VALUE)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Al
ignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(37, 37, 37)
                        .addGroup(jPanel1Layout.createParallelGroup(java
x.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(changeServerStateBtn)
                            .addComponent(serverStatusLbl)))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(serverStatusColourLbl,
javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(16, 16, 16)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 350,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacem
ent.RELATED, 26, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
                    .addComponent(settingsBtn)
                    .addComponent(instructionsBtn)
                    .addComponent(appointLeaderBtn))
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
```

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment
.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>
    /**
     * This method asks the user to confirm whether they would like to
change server settings. If they do, they will be taken to a form that
allows them to edit the server settings
     * @param evt The appropriate button is clicked
     */
    private void settingsBtnActionPerformed(java.awt.event.ActionEvent
evt) {
        int confirm = JOptionPane.showConfirmDialog(this, "Changing the
server settings will mean that the server will go offline until it is
restarted. Are you sure you would you like to
proceed?","Continue?",JOptionPane.YES_NO_CANCEL_OPTION);
        //asks the user to confirm that they wish to proceed
        if(confirm==JOptionPane.YES_OPTION){
            if(isServerRunning){
                changeServerStateBtn.doClick();
            }
            ChangeServerDetails change = new ChangeServerDetails();
            change.setVisible(true);
            this.dispose();
        }
    }

    private void
changeServerStateBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        isServerRunning=!isServerRunning;//changes the sever state
        if(isServerRunning){
            try{
```

```
                //creates a new server and starts it
                errorMessagesArray=new String[0];
                serv = new Server(errorMessagesArray);
                serverThread = new Thread(serv);
                serverThread.start();//starts the server
                timer = new Timer(1000, new ActionListener() {//checks
periodically to see if any erros have occured, and kills the server if
this is the case
                    @Override
                    public void actionPerformed(ActionEvent e) {
                        if(serv.haveAnyErrorsOccured()){
                            //shuts the server down in any errros have
occurred and displays these messages on the text field
                            serverThread.interrupt();
                            errorMessagesArray=serv.getErrorMessages();
                            for(int
i=0;i<errorMessagesArray.length;i++){
                                addMessageToTextArea(errorMessagesArray[
i]);
                            }
                            try {
                                serv.closeServerSocket();
                            } catch (IOException ex) {
                                addMessageToTextArea(ex.getMessage());
                            }
                            isServerRunning=false;
                            serv =null;
                            serverThread=null;
                            timer.stop();
                            updateCompontentsBasedOnServerStatus();
                        }
                    }
                });
                timer.start();//starts the timer
            }catch(Exception e){
                isServerRunning=false;
                addMessageToTextArea(e.getMessage());
                updateCompontentsBasedOnServerStatus();
            }
        }else{
            if(timer!=null){//stops the timer
                timer.stop();
            }
            try {
                serv.closeServerSocket();//closes the server safely
            } catch (IOException ex) {
                addMessageToTextArea(ex.getMessage());
            }
            serv=null;serverThread=null;timer=null;
        }
        updateCompontentsBasedOnServerStatus();//updates the form
    }
```

```
    private void
appointLeaderBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // sets a new club leader if appropriate. This can also be used
to choose a leader when no leaders exist
        int confirm = JOptionPane.showConfirmDialog(this, "Are you would
like to appoint a leader? If a leader exists, they will be replaced.",
"Proceed?", JOptionPane.YES_NO_CANCEL_OPTION);
        if(confirm==JOptionPane.YES_OPTION){
            //appoints somebody else as the chess club leader
            String newLeader = JOptionPane.showInputDialog(this,"Please
input the username of the new leader:");
            if(newLeader==null){
                return;
            }
            try{
                serv.queue.setUserAsLeaderAndMember(newLeader);
                JOptionPane.showMessageDialog(this,"The leader was
appointed.","Leader Appointed.",JOptionPane.OK_OPTION);
            }catch(Exception e){
                JOptionPane.showMessageDialog(this,"The username
inputted was not valid and hence leadership has not changed.","Username
invalid.",JOptionPane.OK_OPTION);
            }
        }
    }

    private void
instructionsBtnActionPerformed(java.awt.event.ActionEvent evt)
{
        // displays the instructions
        JOptionPane.showMessageDialog(this,
                "This application is the server. For the West Cross
Chess software to function, this server needs to be on at all times."
                +System.lineSeparator()+"Simply press the button titled
turn server on to turn the server on. If the server is running"

+System.lineSeparator()+", this form will say 'Server IS Running'. The
server runs on what is known as a port. It is set at 80, which should be
fine"
                +System.lineSeparator()+" however you may wish to change
this using the 'Change Port Number'. This is not advised unless port 80
is used for some other process on your computer."
                +System.lineSeparator()+"The 'Set Club Leader' button is
used to choose a user to be the club leader. This is sometimes used if
an old leader leaves, or when setting the system up."
                +System.lineSeparator()+"In order for users to make use
of this, you must find out the public IPV4 address of your router (or
host network). This may change over time depending on what"
                +System.lineSeparator()+"internet provider you use."
                +System.lineSeparator()+"If any tech support is needed,
please don't hesitate to contact the maker of this system (Philip
Mortimer). Contact Philip at philipmortimer99@aol.com"
```

```
                ,"Instructions",JOptionPane.OK_OPTION);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassNa
me());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ServerUI.class.getName())
.log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ServerUI.class.getName())
.log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ServerUI.class.getName())
.log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ServerUI.class.getName())
.log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ServerUI().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton appointLeaderBtn;
    private javax.swing.JButton changeServerStateBtn;
    private javax.swing.JButton instructionsBtn;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
```

```
    private javax.swing.JTextArea serverInfoTxt;
    private javax.swing.JLabel serverStatusColourLbl;
    private javax.swing.JLabel serverStatusLbl;
    private javax.swing.JButton settingsBtn;
    // End of variables declaration
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

/**
 * This class is used to sort data
 * @author mortimer
 */
public class Sort {
    /**
    * Sorts data using insertion sort
    * @param unsortedData a 2d string array of the unsorted data with
each row representing one record and the nought element containing a
String primary key (which is the field that will be used to sort)
    */
    public static void insertionSort(String[][]unsortedData){
        int indexToBeChecked=0;//loop variable that needs intialisation
for method wide access
        String tempRecord[]=new String[unsortedData[0].length];//used as
buffer storage of a record
        for(int
indexOfElementToBeInserted=1;indexOfElementToBeInserted<unsortedData.len
gth;indexOfElementToBeInserted++){
            for(int
recordIndex=0;recordIndex<tempRecord.length;recordIndex++){//stores
value of record to be "inserted"
                tempRecord[recordIndex]=unsortedData[indexOfElementToBeI
nserted][recordIndex];
            }
            for(indexToBeChecked=indexOfElementToBeInserted-
1;indexToBeChecked>=0&&(tempRecord[0].compareTo(unsortedData[indexToBeCh
ecked][0])<0);indexToBeChecked--){
                for(int
recordIndex=0;recordIndex<tempRecord.length;recordIndex++){
                    unsortedData[indexToBeChecked+1][recordIndex]=unsort
edData[indexToBeChecked][recordIndex];
                }
            }
```

```
            for(int
recordIndex=0;recordIndex<tempRecord.length;recordIndex++){
                unsortedData[indexToBeChecked+1][recordIndex]=tempRecord
[recordIndex];
            }
        }
    }
        /**
     * This function recursively splits all record down into sub arrays
until each sub array only contains one record
     * these split arrays are merged back together using the merge
function.
     * Therefore the data originally input into this function will be
sorted (as arrays are passed by reference in java). The return value of
this function is also the
     * sorted data.
     * @param data The data you want sorted. Assuming each row is one
record and the first element in each row is a string that is the primary
key.
     * @return The sorted data is returned with the data being sorted in
ascending order (one row representing each record and the first element
of each row containing the integer primary key).
     */
    public static String[][] mergeSort(String[][]data){
        if(data.length==1){
            return data;
        }
        int middle = data.length/2; //calculates the middle of the
current data
        String[][]left=new String[middle][data[0].length];//stores the
left half of the current data
        String[][]right = new String[data.length-
middle][data[0].length];//stores the right half of the current data
        for(int yVal=0;yVal<data.length;yVal++){//fills up the left and
right arrays so that the left array stores the left/ top half of the
data array and the right array stores the remianing records
            if(yVal>=middle){
                for(int xVal=0;xVal<data[0].length;xVal++){
                    right[yVal-middle][xVal]=data[yVal][xVal];
                }
            }else{
                for(int xVal=0;xVal<data[0].length;xVal++){
                    left[yVal][xVal]=data[yVal][xVal];
                }
            }
        }
        return merge(mergeSort(left),mergeSort(right),data);//returns
sorted data. calls the merge function on the data that has been
recursively split into single elements
    }
    /**
     *
     * @param left the left hand sub array of data
```

```
     * @param right the right hand sub array of data
     * @param dataToBeSorted the data to be sorted
     * @return returns the merged / sorted data
     */
    private static String[][]
merge(String[][]left,String[][]right,String dataToBeSorted[][]){
        int leftIndex=0;//stores index for left array
        int rightIndex=0;//stores index for right array
        int dataToBeStoredIndex=0;//stores the row index for the data to
be sorted array
        while(leftIndex<left.length && rightIndex<right.length){//both
righ and left arrays are already sorted. sorts one entire sub array and
most of the other sub array
            if(left[leftIndex][0].compareTo(right[rightIndex][0])<0){
                for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                    dataToBeSorted[dataToBeStoredIndex][xVal]=left[leftI
ndex][xVal];
                }
                dataToBeStoredIndex++;
                leftIndex++;
            }else{
                for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                    dataToBeSorted[dataToBeStoredIndex][xVal]=right[righ
tIndex][xVal];
                }
                dataToBeStoredIndex++;
                rightIndex++;
            }
        }
        //as one of the sub arrays has been completely sorted, the
remaining elements of the other array (which are already sorted) are
appended onto the end of the array
        while(leftIndex<left.length){
            for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                dataToBeSorted[dataToBeStoredIndex][xVal]=left[leftIndex
][xVal];
            }
            dataToBeStoredIndex++;
            leftIndex++;
        }
        while(rightIndex<right.length){
            for(int xVal=0;xVal<dataToBeSorted[0].length;xVal++){
                dataToBeSorted[dataToBeStoredIndex][xVal]=right[rightInd
ex][xVal];
            }
            dataToBeStoredIndex++;
            rightIndex++;
        }
        return dataToBeSorted;
    }
    /**
     * Sorts the array using quicksort. The nought element of every row
in the array must be an integer
```

```
     * @param arr The array to sort
     */
    public static void quickSortUsingIntAtFirstElement( String[][]arr){
        quickSortInt(0, arr.length-1, arr);
    }
    /**
     * Performs a quick sort assuming that each record in the array
contains an integer value at the nought element.
     * @param first A parameter used during quicksort
     * @param last A parameter used during quicksort
     * @param arr The array to sort
     */
    private static void quickSortInt(int first, int last,String
arr[][]){
        int left, right, pivot;
        String buffer[]=new String[arr[0].length];
        left =first; right=last;
        pivot = Integer.parseInt(arr[(left+right)/2][0]);
        while(left<=right){
            while(Integer.parseInt(arr[left][0])<pivot){
                left++;
            }
            while(Integer.parseInt(arr[right][0])>pivot){
                right--;
            }
            if(left<=right){
                for(int x=0;x<arr[0].length;x++){//loops through all
fields in record to copy array
                    buffer[x]=arr[right][x];
                    arr[right][x]=arr[left][x];
                    arr[left][x]=buffer[x];
                }
                left++;
                right--;
            }
        }
        if(first<right){
            quickSortInt(first, left-1,arr);
        }
        if(left<last){
            quickSortInt(right+1, last,arr);
        }
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package philipm.cs5.software.development.server.system;

import java.io.Serializable;

/**
 * This is a class that creates a UserInfo Object. This object is
Serializable, allowing to be sent across a stream.
 * @author mortimer
 */
public class UserInfo implements Serializable{
    public static final int WIDTH_OF_USER_DETAILS_FILE=17;//stores the
width of the user file
    private String[]userDetails;//stores the user record
    private boolean isPasswordCorrect=false;//stores whether the
password is correct
    private boolean userFound;//stores whether the user with a given
name was found in the user file
    public static final int WIDTH_OF_EMAIL_LINK_FILE=2;//stores the
width of the email link file
    public static  transient final String
GUEST_ID="GUEST_ID9239923";//stores the guest id
    public static transient final String
GUEST_ID_EMAIL="GuestEmail9932d3@l.com";//stores the guest email address
    private boolean emailAddressFound=true;//stores whether the email
address was found
    public static final String SPLIT_ITEM=";lO9";//used to split data
items
    /*structure of user text FILE:
    Username, password, first name, surname, postcode, addressline one,
address line two, address line three, address line four, discord
username, discord number, date of birth, phone number, isClubLeader,
isClubMember, ELO, hasUserRequestedToJoinClub
    */
    /**
     * Creates a new User with the record found and the password entered
     * @param record The user file record
     * @param userEnteredPasssword The password the user entered to
access the account
     */
    public UserInfo(String[]record,String userEnteredPasssword){
        userFound = record[0]!=null;//updates whether the user was found
        userDetails = record;//stores the record
        if(userFound){
            //checks to see if the password is correct
            isPasswordCorrect=userEnteredPasssword.equals(record[1]);
        }
    }
    /**
     * Creates a new UserInfo object that indicates that the chosen
email address was not found.
     * @return The UserInfoObject
```

```
     */
    public static UserInfo CreateUserInfoEmailNotFound(){
        return new UserInfo();
    }
    /**
     * Called privately to create a user info when email address is not
found
     */
    private UserInfo(){
        emailAddressFound=false;
    }
    /**
     * Gets whether the email address was found
     * @return whether the email address was found
     */
    public boolean getEmailAddressFound(){
        return emailAddressFound;
    }
    /**
     * Gets the password associated with the user
     * @return Returns the password
     */
    public String getPassword(){
        return userDetails[1];
    }
    /**
     * Gets whether the user was found
     * @return whether the user was found
     */
    public boolean getUserFound(){
        return userFound;
    }
    /**
     * Gets whether the password entered was correct
     * @return whether the password entered was correct
     */
    public boolean getIsPasswordCorrect(){
        return isPasswordCorrect;
    }
    /**
     * Returns the user name stored.
     * @return The username
     */
    public String getUserName(){
        return userDetails[0];
    }
    /**
     * Returns whether the user is a club member.
     * @return Whether the user is  a club member
     */
    public boolean getIsClubMember(){
        return userDetails[14].equals("true");
    }
```

```
    /**
     * Returns whether the user is the club leader or not
     * @return Whether the user is club leader or not.
     */
    public boolean getIsClubLeader(){
        return userDetails[13].equals("true");
    }
    /**
     * Returns the ELO rating of the user
     * @return The ELO rating
     */
    public int getElo(){
        return Integer.parseInt(userDetails[15]);
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.user.system;

import java.io.Serializable;
import philipm.cs5.software.development.server.system.UserInfo;

/**
 * This class is used to create a package to send to the server. It
creates an object which essentially stores an array. Element 0 of this
 * array is a command which indicates what the server has to do. The
command is a constant and stored in this class.
 * @author mortimer
 */
public class InputPackages implements Serializable {
    private String[]packageToSend;//this stores the package to send
    //these are the constant which are inserted at the nought element of
the array. These commands indicate which task needs to be performed
    public static final String
GET_USER_INFO_WITH_USERNAME_AND_PASSWORD="UserInfWithPassAndUserName9123
";
    public static final String
GET_USER_INFO_WITH_EMAIL_AND_PASSWORD="UserInfWithPassAndEmail3468";
    public static final String CREATE_ACCOUNT="cReateAccount234s";
    public static final String
GET_ONLINE_GAMES_FOR_USER="getOnlinegamedetailsforuser";
    public static final String
CREATE_RANDOM_GAME="newRandomGameRequest189249";//constant that stores
request to create a new random online game
```

```
    public static final String
DELETE_GAME="deletegameRequest1224";//stores the constant that signals a
request to delete a game
    public static final String
UPDATED_ONLINE_GAME_RECORD="updateOnlinerecordRequestConst23";//requests
to update online record
    public static final String
CREATE_NEW_GAME_AGAINST_FRIEND="createNewGameAgainstFriendReq3924";
    public static final String
JOIN_GAME_USING_GAME_CODE="joindWithCode9123";
    public static final String GET_GAME_RECORD="getGameRecordWithCode";
    public static final String
GET_CLUB_CONTACT_DETAILS="getClubConatctddata35";
    public static final String
GET_WHETHER_USER_HAS_REQUESTED_TO_JOIN_CLUB="hasUserrequestedtojoinclun1
4";
    public static final String
CHANGE_STATUS_REQUEST_TO_JOIN_CLUB="changeREquestStatsu12984";
    public static final String
REMOVE_USER_FROM_CHESS_CLUB="removeUserFromChessClub124";
    public static final String GET_LEAGUE_TABLE="getELaegy";
    public static final String GET_ALL_CLUB_MEMBERS="getAllMembers";
    public static final String
GET_FIXTURES_AND_RESULTS="fixturesAndRes";
    public static final String
GET_PEOPLE_ADDED_MIDWAY_THROUGH_SEASON="getMidwaypead";
    public static final String
UPDATED_FIXTURES_AND_RESULTS="updateFixtsAndres";
    private String[][][]fixturesAndResults; //stores the fixtures and
results if that is what is being sent to the server
    public static final String CREATE_NEW_SEASON="newSeasonas";
    public static final String SET_CONTACT_DETAILS="setContactdakl";
    public static final String SET_USER_AS_LEADER="setLEaderaf";
    public static final String GET_ALL_REQUESTS="alLRequessf";
    public static final String GET_EMAIL_OF_USER="getEmailUseraf";
    public static final String
REMOVE_REQUEST_JOIN_CLUB="setUserRequstClubFalse";
    public static final String ADD_USER_TO_CLUB="addUserToClub";
    public static final String
GET_CLUB_MEMBER_DETAILS_ALL_MEMBERS="getAllClubmeiaefd";
    public static final String
UPDATE_USER_INFO_FILE_RECORD="updahusasf";
    public static final String REMOVE_NON_MEMBER_USER="sausdf";
    public static final String GET_ALL_NON_MEMBERS="getAllNonMebrsfd";
    /**
     * Creates an input package to get user info by providing log in
details
     * @param userName The username
     * @param password The password
     */
    public void createLogInRequestForWithUserNameAndPassword(String
userName,String password){
        packageToSend= new String[3];
```

```
        packageToSend[0]=GET_USER_INFO_WITH_USERNAME_AND_PASSWORD;packag
eToSend[1]=userName;packageToSend[2]=password;
    }
    /**
     * Creates an InputPackage object
     */
    public InputPackages(){

    }
    /**
     * Creates package that requests the fixtures and results
     */
    public void getFixturesAndResults(){
        packageToSend=new
String[1];packageToSend[0]=GET_FIXTURES_AND_RESULTS;
    }
    /**
     * Creates package that requests the the list of people added midway
through the season.
     */
    public void getPeopleAddedMidwayThroughSeason(){
        packageToSend=new
String[1];packageToSend[0]=GET_PEOPLE_ADDED_MIDWAY_THROUGH_SEASON;
    }
    /**
     * Creates package that requests the fixtures and results be updated
to the inputted fixture and results.
     * @param fixturesAndResults The new fixtures and results.
     */
    public void
updateFixturesAndResults(String[][][]fixturesAndResults){
        packageToSend=new String[]{UPDATED_FIXTURES_AND_RESULTS};
        this.fixturesAndResults=fixturesAndResults;
    }
    /**
     * Returns the fixtures and results array stored in the package
     * @return The fixtures and results array
     */
    public String[][][]getFixturesAndResultsArray(){
        return this.fixturesAndResults;
    }
    /**
     * Creates a package that requests to add a specified user to the
chess club.
     * @param userName The username of the user to add.
     */
    public void addUserToClubAsMember(String userName){
        packageToSend=new String[]{ADD_USER_TO_CLUB,userName};
    }
    /**
     * Creates package that requests that a new season is started on the
server.
     */
```

```
    public void startNewSeason(){
        packageToSend=new String[]{CREATE_NEW_SEASON};
    }
    /**
     * Creates a package that requests the server alter the club contact
details.
     * @param phoneNumber The new phone number
     * @param emailAddress The new email address
     */
    public void setContactDetails(String phoneNumber, String
emailAddress){
        packageToSend=new
String[]{SET_CONTACT_DETAILS,phoneNumber,emailAddress};
    }
    /**
     * Creates package that requests for the server to make the chosen
user the club leader
     * @param userName The name of the user
     */
    public void setClubLeader(String userName){
        packageToSend=new String[]{SET_USER_AS_LEADER,userName};
    }
    /**
     * Creates a package that requests the user details of all users who
have requested to join the club.
     */
    public void getAllRequestsToJoinClub(){
        packageToSend=new String[]{GET_ALL_REQUESTS};
    }
    /**
     * Creates a package that requests the email address of a chosen
user
     * @param userName The username of the chosen user
     */
    public void getEmailAddressOfUser(String userName){
        packageToSend=new String[]{GET_EMAIL_OF_USER,userName};
    }
    /**
     * Creates a package that requests that the server give the client
the user details of all non members.
     */
    public void getAllNonMembers(){
        packageToSend=new String[]{GET_ALL_NON_MEMBERS};
    }
    /**
     * Creates a new input package which requests all online game data
on a user
     * @param userName The username
     */
    public void createRequestForAllGameDataForAUser(String userName){
        packageToSend=new String[2];
        packageToSend[1]=userName;
        packageToSend[0]=GET_ONLINE_GAMES_FOR_USER;
```

```
    }
    /**
     * Creates a package that tells the server to withdraw the request
to join the club from a given user.
     * @param userName The username of the user to withdraw the request
from.
     */
    public void withdrawRequestToJoinClub(String userName){
        packageToSend=new String[]{REMOVE_REQUEST_JOIN_CLUB,userName};
    }
    /**
     * Creates package that requests the user information on all club
members.
     */
    public void getAllClubMembers(){
        packageToSend=new
String[1];packageToSend[0]=GET_ALL_CLUB_MEMBERS;
    }
    /**
     * Creates an input package to get user info by providing log in
details
     * @param emailAddress The email Address
     * @param password The password
     */
    public void createLogInRequestForWithEmailAndPassword(String
emailAddress,String password){
        packageToSend= new String[3];
        packageToSend[0]=GET_USER_INFO_WITH_EMAIL_AND_PASSWORD;packageTo
Send[1]=emailAddress;packageToSend[2]=password;
    }
    /**
     * Creates a new package that requests for an online game to be
created
     * @param userName The username of the user who wishes to play a new
random online game
     */
    public void createRandomGamePackage(String userName){
        packageToSend=new String[2];
        packageToSend[0]=CREATE_RANDOM_GAME;packageToSend[1]=userName;
    }
    /**
     * Creates a package that requests all data on club members.
     */
    public void getAllClubMembersInfo(){
        packageToSend=new String[]{GET_CLUB_MEMBER_DETAILS_ALL_MEMBERS};
    }
    /**
     * Creates a package to tell server to update a user record.
     * @param newRecord The new record
     */
    public void updateUserRecord(String[]newRecord){
        packageToSend=new String[UserInfo.WIDTH_OF_USER_DETAILS_FILE+1];
        packageToSend[0]=UPDATE_USER_INFO_FILE_RECORD;
```

```
        for(int i=0;i<UserInfo.WIDTH_OF_USER_DETAILS_FILE;i++){
            packageToSend[i+1]=newRecord[i];
        }
    }
    /**
     * Gets the package array
     * @return The string array storing the action
     */
    public String[]getPackageToSend(){
        return packageToSend;
    }
    /**
     * Tells server to remove non club member from system.
     * @param userName The name of the user account to delete
     */
    public void removeNonMemberUserFromRecord(String userName){
        packageToSend=new String[]{REMOVE_NON_MEMBER_USER,userName};
    }
    /**
     * Returns the action name
     * @return The nought element of the package to send array is
returned
     */
    public String getActionName(){
        return packageToSend[0];
    }
    /**
     * Creates a package that requests for the league table
     */
    public void getLeagueTableRequest(){
        packageToSend=new String[1];
        packageToSend[0]=GET_LEAGUE_TABLE;
    }
    /**
     * Creates a package that requests the server to add a new user
record.
     * @param userRecord The user record array
     * @param userName The username of the user
     * @param emailAddress The email address
     */
    public void createAccountPacket(String []userRecord,String userName,
String emailAddress){
        packageToSend=new String[userRecord.length+3];
        packageToSend[0]=CREATE_ACCOUNT;
        for(int i=0;i<userRecord.length;i++){
            packageToSend[i+1]=userRecord[i];
        }
        packageToSend[userRecord.length+1]=userName;
        packageToSend[userRecord.length+2]=emailAddress;
    }
    /**
     * Returns the user record. Note: this only works if the command is
actually for creating an account.
```

```
     * @return The user record.
     */
    public String[]getUserRecod(){
        String[] rec = new String [UserInfo.WIDTH_OF_USER_DETAILS_FILE];
        for(int i=1;i<UserInfo.WIDTH_OF_USER_DETAILS_FILE+1;i++){
            rec[i-1]=packageToSend[i];
        }
        return rec;
    }
    /**
     * Returns the user name. Must be called only when the input package
is to create an account.
     * @return The username
     */
    public String getUserName(){
        return packageToSend[UserInfo.WIDTH_OF_USER_DETAILS_FILE+1];
    }
    /**
     * Returns the email address. Must be called only when the input
package is to create an account.
     * @return The email address
     */
    public String getEmail(){
        return packageToSend[UserInfo.WIDTH_OF_USER_DETAILS_FILE+2];
    }
    /**
     * Creates a copy of the current input package
     * @return A copy of the package
     */
    public InputPackages getCopyOfInputPackage(){
        String oldArray[]=this.getPackageToSend();
        String newArray[]=new String[oldArray.length];//copies previous
array
        for(int i=0;i<newArray.length;i++){
            newArray[i]=oldArray[i];
        }
        return new InputPackages(newArray);
    }
    private InputPackages(String[]messageArray){
        packageToSend=messageArray;
    }
    /**
     * Creates a packet used to request that a game is deleted
     * @param gameId The game id of the game to be deleted
     * @param gameResult The game result: 1 signifies a win for player
one. 0.5 indicates a draw and 0 indicates a player two win
     */
    public void createRequestToDeleteGame(String gameId, double
gameResult){
        packageToSend=new
String[3];packageToSend[0]=DELETE_GAME;packageToSend[1]=gameId;packageTo
Send[2]=String.valueOf(gameResult);
    }
```

```
    /**
     * Updates the online record
     * @param newOnlineGameRecord The updated record
     */
    public void createRequestToUpdateOnlineGame(String
[]newOnlineGameRecord){
        packageToSend=new String[1+newOnlineGameRecord.length];
        packageToSend[0]=UPDATED_ONLINE_GAME_RECORD;
        for(int i=0;i<newOnlineGameRecord.length;i++){
            packageToSend[i+1]=newOnlineGameRecord[i];
        }
    }
    /**
     * Creates package that tells the server to tell the client whether
the chosen user has requested to join the club.
     * @param userName The name of the user
     */
    public void getWhetherUserHasRequestedToJoinClub(String userName){
        packageToSend=new
String[2];packageToSend[0]=GET_WHETHER_USER_HAS_REQUESTED_TO_JOIN_CLUB;p
ackageToSend[1]=userName;
    }
    /**
     * Creates a package that requests that the server changes the
request status of a user.
     * @param userName The name of the user
     */
    public void changeRequestToJoinClubStatus(String userName){
        packageToSend=new
String[2];packageToSend[0]=CHANGE_STATUS_REQUEST_TO_JOIN_CLUB;packageToS
end[1]=userName;
    }
    /**
     * Creates a package that requests that the chosen user is removed
from the chess club.
     * @param userName The name of the user
     */
    public void removeUserFromChessClub(String userName){
        packageToSend=new
String[2];packageToSend[0]=REMOVE_USER_FROM_CHESS_CLUB;packageToSend[1]=
userName;
    }
    /**
     * Creates a request to get the contact info of the chess club
     */
    public void requestClubContactInfo(){
        packageToSend=new
String[1];packageToSend[0]=GET_CLUB_CONTACT_DETAILS;
    }
    /**
     * Creates a new request to play an online game against a friend
     * @param userNamePlayerOne the username of the first player
     * @param userNamePlayerTwo The second username or "" if the game is
```

```
to be played by sharing a code
     * @param timeLeftForBoth The time left for both players
     */
    public void createNewOnlineGameWithFriend(String userNamePlayerOne,
String userNamePlayerTwo,String timeLeftForBoth){
        packageToSend=new
String[4];packageToSend[0]=CREATE_NEW_GAME_AGAINST_FRIEND;packageToSend[
1]=userNamePlayerOne;
        packageToSend[2]=userNamePlayerTwo;packageToSend[3]=timeLeftForB
oth;
    }
    /**
     * Attempts to join a game using the provided game code
     * @param gameCode The game code
     * @param userName The username of the player
     */
    public void joinGameUsingGameCode(int gameCode,String userName){
        packageToSend=new
String[3];packageToSend[0]=JOIN_GAME_USING_GAME_CODE;packageToSend[1]=St
ring.valueOf(gameCode);packageToSend[2]=userName;
    }
    /**
     * Creates a request to get game data
     * @param gameCode The game id
     */
    public void getGameRecord(int gameCode){
        packageToSend=new
String[2];packageToSend[0]=GET_GAME_RECORD;packageToSend[1]=String.value
Of(gameCode);
    }
}
```

Source Code – Neural Network

The neural network project only has one package: “philipm.cs5.software.development.neural.network.train”.

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.neural.network.train;

/**
 * This class is used to store all data needed for a single board state
 * @author mortimer
 */
public class ChessBoardState {
```

```
    public final int BOARD[][]= new int[8][8];//stores chess
board  white always starts at bottom
    public final boolean IS_BLACK_TURN_TO_PLAY_NEXT;//stores which
colour is to move next
    public final boolean[] CASTLING_RIGHTS=new boolean[4];//stores the
castling rights. Stored in format: White kingside, white queenside,
black kingside, black queenside
    public final int FULL_MOVE_NUMBER;//the numbe of moves black has
made. This value starts at one
    public final int HALF_MOVE_CLOCK;//the number of moves made since a
piece was capture or a pawn advanced a square. This is used to enforce
the fifty move rule. This value starts at 0
    public final  int[]SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT;//stores
sqaure that can be captured via en passant. Has a length of 0 if no such
squares are available
    private  String fenString;//stores fen string of board state
    private boolean isDraw;//stores whether the state is a draw. Note
this must be set externally
    private boolean isWin;//stores whether state is a win for player
currently moving. Note this must be set externally
    private boolean isLoss;//stores whether state is a loss for player
currently moving. Note this must be set externally
    private boolean hasFenBeenCreated=false;
    public static final int WHITE_MULTIPLIER=1;public static final int
BLACK_MULTIPLIER=-1;
    public static final int ROOK_VALUE=5;public static final int
BISHOP_VALUE =4;public static final int KNIGHT_VALUE=3;
    public static final int PAWN_VALUE=1;
    public static final int QUEEN_VALUE=8;public static final int
KING_VALUE=100; public static final int EMPTY_VALUE=0;
    /**
     * Takes the FEN String (standard representation of chess board as
input)
     * @param fenString The FEN string. For explanation of FEN format,
view https://www.chess.com/terms/fen-chess
     */
    public ChessBoardState (String fenString){
        this.fenString=fenString;hasFenBeenCreated=true;
        String[]fenSplit = fenString.split(" ");//splits the FEN String
into its fields
        //generates board based on FEN String
        String board[]=fenSplit[0].split("/");//splits the board data
into ranks
        int xIndex;//stores current x index of piece
        for(int y=0;y<8;y++){//generates board data for each rank
            xIndex=0;
            for(int ch=0;ch<board[y].length();ch++){
                switch (board[y].charAt(ch)) {
                    case 'r':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*ROOK_VALUE;
                        xIndex++;
                        break;
                    case 'n':
```

```
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*KNIGHT_VALUE;
                        xIndex++;
                        break;
                    case 'b':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*BISHOP_VALUE;
                        xIndex++;
                        break;
                    case 'q':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*QUEEN_VALUE;
                        xIndex++;
                        break;
                    case 'k':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*KING_VALUE;
                        xIndex++;
                        break;
                    case 'p':
                        BOARD[y][xIndex]=BLACK_MULTIPLIER*PAWN_VALUE;
                        xIndex++;
                        break;
                    case 'N':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*KNIGHT_VALUE;
                        xIndex++;
                        break;
                    case 'B':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*BISHOP_VALUE;
                        xIndex++;
                        break;
                    case 'Q':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*QUEEN_VALUE;
                        xIndex++;
                        break;
                    case 'K':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*KING_VALUE;
                        xIndex++;
                        break;
                    case 'R':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*ROOK_VALUE;
                        xIndex++;
                        break;
                    case 'P':
                        BOARD[y][xIndex]=WHITE_MULTIPLIER*PAWN_VALUE;
                        xIndex++;
                        break;
                    default:
                        for(int
emp=0;emp<Character.getNumericValue(board[y].charAt(ch));emp++){
                            BOARD[y][xIndex]=EMPTY_VALUE;
                            xIndex++;
                        }   break;
                }
            }
        }
        IS_BLACK_TURN_TO_PLAY_NEXT=fenSplit[1].equals("b");//gets who's
```

```
move it is
        //gets castling rights
        //stored in format White kingside, white queenside, black
kingside, black queenside
        CASTLING_RIGHTS[0]=false;CASTLING_RIGHTS[1]=false;CASTLING_RIGHT
S[2]=false;CASTLING_RIGHTS[3]=false;
        if(!fenSplit[2].equals("-")){
            for(int ch =0;ch<fenSplit[2].length();ch++){
                switch (fenSplit[2].charAt(ch)) {
                    case 'K':
                        CASTLING_RIGHTS[0]=true;
                        break;
                    case 'Q':
                        CASTLING_RIGHTS[1]=true;
                        break;
                    case 'k':
                        CASTLING_RIGHTS[2]=true;
                        break;
                    default:
                        CASTLING_RIGHTS[3]=true;
                        break;
                }
            }
        }
        //gets possible en passant targets
        if(fenSplit[3].equals("-")){
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT=new int[0];//no
available squares
        }else{
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT =
getYAndXOfSquare(fenSplit[3]);//one available en passant square
        }
        //gets half move clock
        HALF_MOVE_CLOCK=Integer.parseInt(fenSplit[4]);
        //gets full move number
        FULL_MOVE_NUMBER=Integer.parseInt(fenSplit[5]);
    }
    /**
     * Creates a new chess board state
     * @param gameBoard The board array
     * @param castlingRights The castling rights. In the form, white
kingside, white queen side, black kingside, black queen side
     * @param isBlackToPlayNext whether black is to make the next move
     * @param fullMoveNumber the number of moves black has played
     * @param halfMoveClock the number of moves both players have made
since the last pawn advance or capture
     * @param enPassantSquare the y and x coordinates of a square that
can be captured via en passant. If no such square exists, use new int[0]
     */
    public ChessBoardState(int[][]gameBoard,boolean
castlingRights[],boolean isBlackToPlayNext, int fullMoveNumber,int
halfMoveClock,int[]enPassantSquare){
        //stores board
```

```
        for(int y=0;y<8;y++){//generates board
            for(int x =0;x<8;x++){
                BOARD[y][x]=gameBoard[y][x];
            }
        }
        CASTLING_RIGHTS[0]=castlingRights[0];CASTLING_RIGHTS[1]=castling
Rights[1];CASTLING_RIGHTS[2]=castlingRights[2];CASTLING_RIGHTS[3]=castli
ngRights[3];
        IS_BLACK_TURN_TO_PLAY_NEXT=isBlackToPlayNext;
        FULL_MOVE_NUMBER=fullMoveNumber;
        HALF_MOVE_CLOCK=halfMoveClock;
        if(enPassantSquare.length==0){
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT= new int[0];
        }else{
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT=new int [2];
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[0]=enPassantSquare[0]
;
            SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]=enPassantSquare[1]
;
        }
    }
    public int[]getNeuralNetInputReducedSize(){
        int data[]=new int[64+4+2+1];
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                data[8*y +x]=BOARD[y][x];
            }
        }
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            data[64]=1;
        }
        if(CASTLING_RIGHTS[0]){
            data[65]=1;
        }
        if(CASTLING_RIGHTS[1]){
            data[66]=1;
        }
        if(CASTLING_RIGHTS[2]){
            data[67]=1;
        }
        if(CASTLING_RIGHTS[3]){
            data[68]=1;
        }
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            data[69]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[0];
            data[70]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1];
        }
        return data;
    }
    public double[]getNeuralNetInputDouble(){
        double data[]= new double [777];//creates the array that stores
the neural network attributes
        //stores castling rights
```

```
        if(CASTLING_RIGHTS[0]){
            data[768]=1;
        }
        if(CASTLING_RIGHTS[1]){
            data[769]=1;
        }
        if(CASTLING_RIGHTS[2]){
            data[770]=1;
        }
        if(CASTLING_RIGHTS[3]){
            data[771]=1;
        }
        //stores who is going to play next
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            data[772]=1;
        }
        //sets half move data
        data[773]=HALF_MOVE_CLOCK;
        //sets en passant data
        data[775]=9;data[776]=9;
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            data[774]=1;data[775]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT
[0];data[776]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1];
        }
        //generates bitboard data
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                switch (BOARD[y][x]) {
                    case WHITE_MULTIPLIER*KING_VALUE:
                        data[x+(8*y)]=1;
                        break;
                    case BLACK_MULTIPLIER*KING_VALUE:
                        data[x+(8*y)+64]=1;
                        break;
                    case WHITE_MULTIPLIER*ROOK_VALUE:
                        data[x+(8*y)+128]=1;
                        break;
                    case BLACK_MULTIPLIER*ROOK_VALUE:
                        data[x+(8*y)+192]=1;
                        break;
                    case WHITE_MULTIPLIER*BISHOP_VALUE:
                        data[x+(8*y)+256]=1;
                        break;
                    case BLACK_MULTIPLIER*BISHOP_VALUE:
                        data[x+(8*y)+320]=1;
                        break;
                    case WHITE_MULTIPLIER*QUEEN_VALUE:
                        data[x+(8*y)+384]=1;
                        break;
                    case BLACK_MULTIPLIER*QUEEN_VALUE:
                        data[x+(8*y)+448]=1;
                        break;
                    case WHITE_MULTIPLIER*PAWN_VALUE:
```

```
                        data[x+(8*y)+512]=1;
                        break;
                    case BLACK_MULTIPLIER*PAWN_VALUE:
                        data[x+(8*y)+576]=1;
                        break;
                    case WHITE_MULTIPLIER*KNIGHT_VALUE:
                        data[x+(8*y)+640]=1;
                        break;
                    case BLACK_MULTIPLIER*KNIGHT_VALUE:
                        data[x+(8*y)+704]=1;
                        break;
                    default:
                        break;
                }
            }
        }
        return data;
    }
    /**
     * Gets the string used for the neural network input given the
current board state
     * @return The input to the neural network
     */
    public int[] getNeuralNetInput(){
        int data[]= new int [777];//creates the array that stores the
neural network attributes
        //stores castling rights
        if(CASTLING_RIGHTS[0]){
            data[768]=1;
        }
        if(CASTLING_RIGHTS[1]){
            data[769]=1;
        }
        if(CASTLING_RIGHTS[2]){
            data[770]=1;
        }
        if(CASTLING_RIGHTS[3]){
            data[771]=1;
        }
        //stores who is going to play next
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            data[772]=1;
        }
        //sets half move data
        data[773]=HALF_MOVE_CLOCK;
        //sets en passant data
        data[775]=9;data[776]=9;
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            data[774]=1;data[775]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT
[0];data[776]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1];
        }
        //generates bitboard data
        for(int y=0;y<8;y++){
```

```
            for(int x=0;x<8;x++){
                switch (BOARD[y][x]) {
                    case WHITE_MULTIPLIER*KING_VALUE:
                        data[x+(8*y)]=1;
                        break;
                    case BLACK_MULTIPLIER*KING_VALUE:
                        data[x+(8*y)+64]=1;
                        break;
                    case WHITE_MULTIPLIER*ROOK_VALUE:
                        data[x+(8*y)+128]=1;
                        break;
                    case BLACK_MULTIPLIER*ROOK_VALUE:
                        data[x+(8*y)+192]=1;
                        break;
                    case WHITE_MULTIPLIER*BISHOP_VALUE:
                        data[x+(8*y)+256]=1;
                        break;
                    case BLACK_MULTIPLIER*BISHOP_VALUE:
                        data[x+(8*y)+320]=1;
                        break;
                    case WHITE_MULTIPLIER*QUEEN_VALUE:
                        data[x+(8*y)+384]=1;
                        break;
                    case BLACK_MULTIPLIER*QUEEN_VALUE:
                        data[x+(8*y)+448]=1;
                        break;
                    case WHITE_MULTIPLIER*PAWN_VALUE:
                        data[x+(8*y)+512]=1;
                        break;
                    case BLACK_MULTIPLIER*PAWN_VALUE:
                        data[x+(8*y)+576]=1;
                        break;
                    case WHITE_MULTIPLIER*KNIGHT_VALUE:
                        data[x+(8*y)+640]=1;
                        break;
                    case BLACK_MULTIPLIER*KNIGHT_VALUE:
                        data[x+(8*y)+704]=1;
                        break;
                    default:
                        break;
                }
            }
        }
        return data;
    }
    /**
     * Gets FEN string from board state
     * @return The FEN String
     */
    public String getFenString(){
        if(this.hasFenBeenCreated){
            return this.fenString;
        }
```

```
        String fenStringBoard="";//stores board as FEN string
        int noOfEmptySquares;//stores the number of empty squares next
to each other in a row
        for(int y=0;y<8;y++){//generates board and FEN String for the
board
            if(y!=0){
                fenStringBoard=fenStringBoard+"/";
            }
            noOfEmptySquares=0;
            for(int x =0;x<8;x++){
                switch (BOARD[y][x]) {
                    case EMPTY_VALUE:
                        noOfEmptySquares++;
                        if(x==7){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                        }   break;
                    case BLACK_MULTIPLIER*ROOK_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"r";
                        break;
                    case BLACK_MULTIPLIER*KNIGHT_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"n";
                        break;
                    case BLACK_MULTIPLIER*BISHOP_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"b";
                        break;
                    case BLACK_MULTIPLIER*QUEEN_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"q";
                        break;
                    case BLACK_MULTIPLIER*KING_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"k";
                        break;
                    case BLACK_MULTIPLIER*PAWN_VALUE:
```

```
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"p";
                        break;
                    case WHITE_MULTIPLIER*ROOK_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"R";
                        break;
                    case WHITE_MULTIPLIER*KNIGHT_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"N";
                        break;
                    case WHITE_MULTIPLIER*BISHOP_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"B";
                        break;
                    case WHITE_MULTIPLIER*QUEEN_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"Q";
                        break;
                    case WHITE_MULTIPLIER*KING_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"K";
                        break;
                    case WHITE_MULTIPLIER*PAWN_VALUE:
                        if(noOfEmptySquares!=0){
                            fenStringBoard=fenStringBoard+String.valueOf
(noOfEmptySquares);
                            noOfEmptySquares=0;
                        }   fenStringBoard=fenStringBoard+"P";
                        break;
                    default:
                        break;
                }
            }
        }
```

```
        String fenCastlingRights="";
        if(CASTLING_RIGHTS[0]){
            fenCastlingRights="K";
        }
        if(CASTLING_RIGHTS[1]){
            fenCastlingRights=fenCastlingRights+"Q";
        }
        if(CASTLING_RIGHTS[2]){
            fenCastlingRights=fenCastlingRights+"k";
        }
        if(CASTLING_RIGHTS[3]){
            fenCastlingRights=fenCastlingRights+"q";
        }
        if(fenCastlingRights.equals("")){
            fenCastlingRights="-";
        }
        String fenColourToPlay="";
        if(IS_BLACK_TURN_TO_PLAY_NEXT){
            fenColourToPlay="b";
        }else{
            fenColourToPlay="w";
        }
        String enPassantSquareFen="-";
        if(SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length==2){
            enPassantSquareFen=getStringOfSquare(SQUARE_THAT_CAN_BE_CAPT
URED_EN_PASSANT[0], SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[1]);
        }
        this.fenString=fenStringBoard+" "+fenColourToPlay+"
"+fenCastlingRights+" "+enPassantSquareFen+"
"+String.valueOf(HALF_MOVE_CLOCK)+" "+String.valueOf(FULL_MOVE_NUMBER);
        this.hasFenBeenCreated=true;
        return this.fenString;
    }
    /**
     * Gets the string value of a square given its y and x coordinates.
E.g. getStringOfSquare(1,0) = "a7"
     * @param y The y index in the board array
     * @param x The x index in the board array
     * @return The square
     */
    private static String getStringOfSquare(int y, int x){
        switch (x) {
            case 0:
                return "a"+String.valueOf(8-y);
            case 1:
                return "b"+String.valueOf(8-y);
            case 2:
                return "c"+String.valueOf(8-y);
            case 3:
                return "d"+String.valueOf(8-y);
            case 4:
                return "e"+String.valueOf(8-y);
            case 5:
```

```
                return "f"+String.valueOf(8-y);
            case 6:
                return "g"+String.valueOf(8-y);
            default:
                return "h"+String.valueOf(8-y);
        }
    }
    /**
     * Takes a chess square, such as "a3" as input and returns the y and
x coordinates of the square on the board
     * @param square The square
     * @return An int array of length [2], with the y and x index for
the board array
     */
    private static int[] getYAndXOfSquare(String square){
        int[]ret = new int[2];
        ret[0]=8-Character.getNumericValue(square.charAt(1));
        switch (square.charAt(0)) {
            case 'a':
                ret[1]=0;
                break;
            case 'b':
                ret[1]=1;
                break;
            case 'c':
                ret[1]=2;
                break;
            case 'd':
                ret[1]=3;
                break;
            case 'e':
                ret[1]=4;
                break;
            case 'f':
                ret[1]=5;
                break;
            case 'g':
                ret[1]=6;
                break;
            default:
                ret[1]=7;
                break;
        }
        return ret;
    }
    /**
     * Gets a copy of the board array. use this when writing with board
     * @return A copy of the board array
     */
    public int[][]getCopyOfBoard(){
        int[][]board=new int[8][8];
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
```

```
                board[y][x]=BOARD[y][x];
            }
        }
        return board;
    }
    /**
     * Returns a copy of the castling rights array
     * @return The castling rights
     */
    public boolean []getCopyOfCastlingRights(){
        boolean rights []=new boolean[4];
        for(int i=0;i<4;i++){
            rights[i]=CASTLING_RIGHTS[i];
        }
        return rights;
    }
    /**
     * Gets a copy of the en passant square array and returns it.
     * @return The copy of the array
     */
    public int[]getCopyOfEnPassantSquare(){
        int ret[]=new
int[SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length];
        for(int i=0;i<ret.length;i++){
            ret[i]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[i];
        }
        return ret;
    }
    /**
     * Returns whether this game is a win. Note this variable may not be
initialised and can only be set by external methods. Hence, only
     * call this method if you are certain that the board state has been
set.
     * @return Whether the board is a win for the current user
     */

    public boolean getIsWin(){
        return this.isWin;
    }
    /**
     * Returns whether this game is a draw. Note: this variable is not
automatically initialised and is set externally.
     * @return Whether the game is a draw
     */
    public boolean getIsDraw(){
        return this.isDraw;
    }
    /**
     * Returns whether this game is a loss for the user to play
currently. Note this variable is not automatically initialised, so only
call this method if the variable has been initialised.
     * @return Whether the game is a loss
     */
```

```
    public boolean getIsLoss(){
        return this.isLoss;
    }
    /**
     * Sets the game as win
     */
    public void setIsWinTrue(){
        this.isWin=true;this.isLoss=false;this.isDraw=false;
    }
    /**
     * Sets the game as a draw
     */
    public void setIsDrawTrue(){
        this.isDraw=true;this.isLoss=false;this.isWin=false;
    }
    /**
     * Sets the game as a loss
     */
    public void setIsLossTrue(){
        this.isLoss=true;this.isDraw=false;this.isWin=false;
    }
    /**
     * Sets the FEN string if it has already been created
     * @param fen The FEN
     */
    private void setFen(String fen){
        this.fenString=fen;
    }
    /**
     * Sets whether the FEN has been created
     * @param hasFenBeenCreated Has the FEN been created
     */
    private void setHasFenBeenCreated(boolean hasFenBeenCreated){
        this.hasFenBeenCreated=hasFenBeenCreated;
    }
    /**
     * Sets whether the game is a loss
     * @param isLoss Whether the game is a loss
     */
    private void setIsLoss(boolean isLoss){
        this.isLoss=isLoss;
    }
    /**
     * Sets whether the game is a draw
     * @param isDraw Whether the game is a draw
     */
    private void setIsDraw(boolean isDraw){
        this.isDraw=isDraw;
    }
    /**
     * Sets whether the game is a win
     * @param isWin Whether the game is a win
     */
```

```
    private void setIsWin(boolean isWin){
        this.isWin=isWin;
    }
    public ChessBoardState getCopyOfState(){
        int[][]board=new int[8][8];//creates a copy of the board
        for(int y=0;y<8;y++){
            for(int x=0;x<8;x++){
                board[y][x]=BOARD[y][x];
            }
        }
        int[]enPassant = new
int[SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length];
        for(int
i=0;i<SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT.length;i++){
            enPassant[i]=SQUARE_THAT_CAN_BE_CAPTURED_EN_PASSANT[i];
        }
        boolean castling[]=new boolean[CASTLING_RIGHTS.length];
        for(int i=0;i<CASTLING_RIGHTS.length;i++){
            castling[i]=CASTLING_RIGHTS[i];
        }
        //creates a new object copying the board
        ChessBoardState newState = new ChessBoardState(board, castling,
IS_BLACK_TURN_TO_PLAY_NEXT, FULL_MOVE_NUMBER, HALF_MOVE_CLOCK,
enPassant);
        newState.setHasFenBeenCreated(hasFenBeenCreated);
        newState.setFen(fenString);
        newState.setIsWin(isWin);
        newState.setIsLoss(isLoss);
        newState.setIsDraw(isDraw);
        return newState;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.neural.network.train;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

/**
 * This is a neural network trained by me to play chess.
 * @author mortimer
```

```
 */
public class ChessNeuralNetwork {
    private static final String
BEST_NET_FILE_NAME="bestNet.txt";//stores the location of the text file
that stores the best network architecture
    private final double[] ALL_WEIGHTS_AND_BIASES;//stores all weights
and biases of the neural network
    private final int[]NETWORK_STRUCTURE;//stores the network structure
    private final int NET_LENGTH;//stores the length of the net
architecture
    private final double[] NET_ACTIVATIONS;//stores the network
activation values
    /**
     * Loads the neural network stored in the "bestNet.txt" file.
     * @param NETWORK_STRUCTURE The network structure of the neural
network
     * @throws java.io.IOException An IOException may be thrown when
reading the values of the network's weights and biases from
"bestNet.txt"
     */
    public ChessNeuralNetwork(final int[] NETWORK_STRUCTURE)throws
IOException{
        this.NETWORK_STRUCTURE= NETWORK_STRUCTURE;
        NET_LENGTH=NETWORK_STRUCTURE.length;
        int totNeurons=NETWORK_STRUCTURE[0]+NETWORK_STRUCTURE[1];
        int
noOfWeightsAndBiases=NETWORK_STRUCTURE[0]*NETWORK_STRUCTURE[1]+NETWORK_S
TRUCTURE[1];//stores the number of weights and biases in the network
        for(int i=2;i<NETWORK_STRUCTURE.length;i++){//calculates the
total number of weights and biases
            noOfWeightsAndBiases=noOfWeightsAndBiases+NETWORK_STRUCTURE[
i]*NETWORK_STRUCTURE[i-1]+NETWORK_STRUCTURE[i];
            totNeurons=totNeurons+NETWORK_STRUCTURE[i];
        }
        if(NETWORK_STRUCTURE.length>2){
            totNeurons=totNeurons-NETWORK_STRUCTURE[NET_LENGTH-1];
        }
        NET_ACTIVATIONS=new double[totNeurons];
        ALL_WEIGHTS_AND_BIASES=new
double[noOfWeightsAndBiases];//creates array storing all weights and
biases
        //reads network architecture from text file
        FileReader read = new FileReader(BEST_NET_FILE_NAME);
        int i=0;//stores the index of all weights and biases to be
accessed
        BufferedReader buffRead = new BufferedReader(read);
        String lineRead;//stores the line read
        while((lineRead=buffRead.readLine())!=null){//loads values of
neural network
            ALL_WEIGHTS_AND_BIASES[i]=Double.parseDouble(lineRead);
            i++;
        }
        buffRead.close();read.close();
```

```
    }
    /**
     * Feeds data through the network and returns the network's output
     * @param input The input
     * @return The output
     */
    public double[] feedThroughNet(int[]input){
        int currentIndexForAll=0;//stores the current index for all
weights and biases
        int activsIndex=0;
        double out[]=new double[NETWORK_STRUCTURE[NET_LENGTH-1]];
        int startOfPrevLayer=0;
        for(;activsIndex<input.length;activsIndex++){
           NET_ACTIVATIONS[activsIndex]=input[activsIndex];
        }
        double sum;//stores the sum that amounts to the input to the
node
        for(int layerIndex=1;layerIndex<NET_LENGTH-1;layerIndex++){
            for(int
neuronInLayer=0;neuronInLayer<NETWORK_STRUCTURE[layerIndex];neuronInLaye
r++){//loops through every neuron in current layer
                sum=ALL_WEIGHTS_AND_BIASES[currentIndexForAll];//sets
the sum equal to the bias
                currentIndexForAll++;
                for(int
neuronInPrevLayer=0;neuronInPrevLayer<NETWORK_STRUCTURE[layerIndex-
1];neuronInPrevLayer++){//sums weights in previous layer to calculate
input value to node
                    sum=sum+NET_ACTIVATIONS[startOfPrevLayer+neuronInPre
vLayer]*ALL_WEIGHTS_AND_BIASES[currentIndexForAll];
                    currentIndexForAll++;
                }
                NET_ACTIVATIONS[activsIndex]=reLu(sum);
                activsIndex++;
            }
            startOfPrevLayer=startOfPrevLayer+NETWORK_STRUCTURE[layerInd
ex-1];
        }
        //calculates output
        for(int
neuronInLayer=0;neuronInLayer<NETWORK_STRUCTURE[NET_LENGTH-
1];neuronInLayer++){//loops through every neuron in output layer
            sum=ALL_WEIGHTS_AND_BIASES[currentIndexForAll];//sets the
sum equal to the bias
            currentIndexForAll++;
            for(int
neuronInPrevLayer=0;neuronInPrevLayer<NETWORK_STRUCTURE[NET_LENGTH-
2];neuronInPrevLayer++){//sums weights in previous layer to calculate
input value to node
                sum=sum+NET_ACTIVATIONS[startOfPrevLayer+neuronInPrevLay
er]*ALL_WEIGHTS_AND_BIASES[currentIndexForAll];
                currentIndexForAll++;
            }
```

```
            out[neuronInLayer]=sum;
        }
        softmax(out);
        return out;
    }
    /**
     * Executes the reLu function
     * @param x The input
     * @return The output
     */
    private static double reLu(double x){
        if(x>0.0){
            return x;
        }else{
            return 0.0;
        }
    }
    /**
     * Performs the softmax function on the inputted array
     * @param input The array to perform the softmax function on
     */
    private static void softmax(double[]input){
        input[0]=Math.exp(input[0]);
        double sum=input[0];//stores the sum of the exponentials
        for(int i=1;i<input.length;i++){
            input[i]=Math.exp(input[i]);
            sum=sum+input[i];
        }
        for(int i=0;i<input.length;i++){
            input[i]=input[i]/sum;
        }
    }
     /* Tests the network's accuracy
     * @param testData The test data to use when testing accuracy
     * @return The accuracy and loss of the system as an array of length
2. Element 0 corresponds to the accuracy, whilst element 1 is the loss.
     */
    public double[]testAccuracy(int testData[][]){
        double time = System.currentTimeMillis();//used to calculate how
long it takes to test the system
        int totalCorrect=0;double[]out;
        double[]desiredOut = new double[2];
        double averageLoss=0.0;int greatestIndex;double err;
        double diff;
        int[]netInput=new
int[PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain.WIDTH_OF_DATA-
2];//stores the network input
        for(int y=0;y<testData.length;y++){//loops trhough every item in
the test dataset
            desiredOut[0]=testData[y][0];//gets the desired output of
the neural network
            desiredOut[1]=testData[y][1];
            for(int x=2;x<testData[0].length;x++){//generates the input
```

```
to the neural network
                netInput[x-2]=testData[y][x];
            }
            out=this.feedThroughNet(netInput);//passes the network input
throught the neural network
            greatestIndex=0;//determines whether out[0] or out[1] is
biggest
            if(out[1]>out[0]){
                greatestIndex=1;
            }
            if(desiredOut[greatestIndex]==1.0){//checks to see if
network is correct
                totalCorrect++;
            }
           /* diff=out[0]-desiredOut[0];
            err=diff*diff;*/
           err=0;
            //calculates the loss of the network
            for(int i=0;i<out.length;i++){
                err = err+(desiredOut[i]*Math.log(out[i]));//categorical
cross entropy loss is employed -  this makes the assumption that the
desired output is a one hot vector

            }err=err*-1.0;
            averageLoss=averageLoss+(err);//adds loss to total loss
        }

        double accuracy =
(double)totalCorrect/(double)PhilipMCS5SoftwareDevelopmentNeuralNetworkT
rain.NO_OF_TEST_BOARDS;//calculates the accuracy of the network
        accuracy= accuracy*100.0;
        averageLoss=averageLoss/(double)PhilipMCS5SoftwareDevelopmentNeu
ralNetworkTrain.NO_OF_TEST_BOARDS;//calculates the average loss
        System.out.println("Testing time:
"+((System.currentTimeMillis()-time)/1000.0)+" seconds . Total correct:
"+totalCorrect+" out of
"+PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain.NO_OF_TEST_BOARDS);
        double ret[]={accuracy,averageLoss};
        return ret;
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package philipm.cs5.software.development.neural.network.train;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

/**
 * This class is used to train my chess neural network
 * @author mortimer
 */
public class NeuralNetwork {
    private final int NET_STRUCTURE_LENGTH;//stores the number of
weights and biases in the neural network
    private final int NET_STRUCTURE_LENGTH_MINUS_ONE;
    private final int ALL_LENGTH;//stores the length of all weights and
biases array
    private final int[]TOT_NEURON;
    private final int[]CURRENT_ALL;
    private final Random RAND = new Random();//creates a random object
to be used in network
    private final int[]INDEX_FOR_ALL_WEIGHTS_ERROR;//stores the indexes
to be used in backpropogation to optimise the process
    private final int[]INDEX_FOR_ACTIVS_ERROR;//stores the indexes to be
used when calculating the error of the network
    private final int ACTIV_IN;
    private final int[]NET_STRUCTURE;//stores the network structure
    private final double[]ALL_WEIGHTS_AND_BIASES;// stores all weights
and biases in the network
    private final int TOTAL_NEURONS;
    private final int TOTAL_NEURONS_MINUS_LAST_LAYER;//stores the number
of nodes in the network that are not in the output layer
    private double[] nodeValuePreActivation;//stores the input to nodes
in a given layer before the activation function is applied
    private double []nodeValuePostActivation;//stores the post
activation values of a nodes in a layer in the network
    private static final String FILE_NAME = "networkInfo.txt";//stores
file name where network data is stored
    private double timeStep; //stores the time step
    private double previousMHat[];//stores previous values of MHat -
used to allow training using ADAM optimiser over multiple training
method calls
    private double previousVHat[];//stores previous values of VHat -
used to allow training using ADAM optimiser over multiple training
method calls
    private static final String
BEST_NET_FILE_NAME="bestNet.txt";//stores the name of the text file that
is used to store the best neural network
    private final boolean []NEURONS_DROPPED;//stores which neurons are
dropped
    private final int LENGTH_OF_NEURONS_DROPPED_ARRAY;
```

```
    private final int INDEX_CACHE[];
    /**
     * Creates a neural network of the desired architecture using reLu
in all layer bar the output layer, where the softmax function is used
     * @param netStructure The network structure, with element 0
specifying the number of nodes in the input layer and the final element
specifying the number of nodes in the output layer
     */
    public  NeuralNetwork(int[]netStructure){
        //initilaises various constants used to speed up the
backpropogation process
        this.NET_STRUCTURE = netStructure;
        this.NET_STRUCTURE_LENGTH=this.NET_STRUCTURE.length;
        this.NET_STRUCTURE_LENGTH_MINUS_ONE=this.NET_STRUCTURE_LENGTH-1;
        int totalWeightsAndBiases=0;
        int totalNeurons=0;
        INDEX_CACHE=new int[netStructure.length-1];
        //calculates the total number of neurons as well as the total
number of weights and biases in the network
        for(int i =0;i<netStructure.length;i++){
            if(i!=netStructure.length-1 && i!=0){
                INDEX_CACHE[i]=totalNeurons-netStructure[0];
            }
            totalNeurons=totalNeurons+this.NET_STRUCTURE[i];
            if(i!=0){
                totalWeightsAndBiases =
totalWeightsAndBiases+this.NET_STRUCTURE[i]+(this.NET_STRUCTURE[i]*this.
NET_STRUCTURE[i-1]);
            }
        }
        NEURONS_DROPPED=new boolean[totalNeurons-netStructure[0]-
netStructure[netStructure.length-1]];
        LENGTH_OF_NEURONS_DROPPED_ARRAY=NEURONS_DROPPED.length;
        //initilaises more constants used for optimisation purposes
        this.TOTAL_NEURONS = totalNeurons;
        this.TOTAL_NEURONS_MINUS_LAST_LAYER=this.TOTAL_NEURONS-
this.NET_STRUCTURE[this.NET_STRUCTURE_LENGTH_MINUS_ONE];
        this.ALL_WEIGHTS_AND_BIASES=new double[totalWeightsAndBiases];
        this.ALL_LENGTH=totalWeightsAndBiases;
        this.ACTIV_IN=this.TOTAL_NEURONS-
this.NET_STRUCTURE[this.NET_STRUCTURE.length-1];
        int currentIndex=0;
        //initialises the value of weights and biases in the network to
values that make learning optimal
        for(int
layerIndex=0;layerIndex<this.NET_STRUCTURE.length;layerIndex++){
            if(layerIndex!=0){
                for(int
currentLayerIndex=0;currentLayerIndex<this.NET_STRUCTURE[layerIndex];cur
rentLayerIndex++){
                    this.ALL_WEIGHTS_AND_BIASES[currentIndex]=0.0;//all
biases are initiliased to 0
                    currentIndex++;
```

```
                    for(int
prevLayerIndex=0;prevLayerIndex<this.NET_STRUCTURE[layerIndex-
1];prevLayerIndex++){//weigths leading into bias
                        this.ALL_WEIGHTS_AND_BIASES[currentIndex]=getRan
dDoubleBetweenOneAndMinusOne(this.NET_STRUCTURE[layerIndex-
1],this.NET_STRUCTURE[layerIndex]);//use xavier initialisation to
intialise the network
                        currentIndex++;
                    }
                }
            }
        }
        //the backpropogation process is perfromed on made up data to
allow for fast backpropogation during the training stage of the network
        this.feedThroughNet(new double[this.NET_STRUCTURE[0]]);//feeds a
random input into the network
        double[]desiredOutput=new
double[this.NET_STRUCTURE[this.NET_STRUCTURE_LENGTH_MINUS_ONE]];//create
s a random desired output
        double[]errorInLayer= new double[1];//stores the network error
        double[]errorInLayerAbove=new double[1];//stores error in layer
above
        double sumOfErrorAndWeights;
        double grad[]=new double[this.ALL_WEIGHTS_AND_BIASES.length];
        int indexForAllWhenDoingWeightsAndError;
        int indexWeightErrorChange;
        int indexForActivsError;
        int totalAllExplored;
        int indexForAllGrad;
        int totNeuronsExplored;
        int indexForActivsGrad;
        int activIndex;
        //this process gets the indexs accessed during training to avoid
repeating the same calculation over and over again
        int[] arrayIndexForAllWhenDoingWeightsAndError=new
int[this.NET_STRUCTURE.length];//these arrays are used to speed of the
backpropogation process
        int[]arrayForIndexForActivsError= new
int[this.NET_STRUCTURE.length];
        int[]arrayForIndexForAllGrad=new int[this.NET_STRUCTURE.length];
        int []arrayForTotNeuronsExplored=new
int[this.NET_STRUCTURE.length];
        for(int layerIndex=this.NET_STRUCTURE.length-
1;layerIndex>0;layerIndex--){
            errorInLayer=new double[this.NET_STRUCTURE[layerIndex]];
            if(layerIndex==this.NET_STRUCTURE.length-1){
                activIndex=this.TOTAL_NEURONS-
this.NET_STRUCTURE[this.NET_STRUCTURE.length-1];
                for(int
neuronInLayer=0;neuronInLayer<this.NET_STRUCTURE[layerIndex];neuronInLay
er++){
                    errorInLayer[neuronInLayer]=(this.nodeValuePostActiv
ation[activIndex]-
```

```
desiredOutput[neuronInLayer])*activationDerivative(this.nodeValuePreActi
vation[activIndex]);
                    activIndex++;
                }
            }else{
                indexForAllWhenDoingWeightsAndError=0;
                for(int layer=1;layer<layerIndex+1;layer++){//calcs
first index in all of next layer
                    indexForAllWhenDoingWeightsAndError=indexForAllWhenD
oingWeightsAndError+(this.NET_STRUCTURE[layer]*this.NET_STRUCTURE[layer-
1]+this.NET_STRUCTURE[layer]);
                }
                arrayIndexForAllWhenDoingWeightsAndError[layerIndex]=ind
exForAllWhenDoingWeightsAndError;
                indexForActivsError=0;
                for(int layer=0;layer<layerIndex;layer++){//frist index
of activ in current layer
                    indexForActivsError=indexForActivsError+this.NET_STR
UCTURE[layer];
                }
                arrayForIndexForActivsError[layerIndex]=indexForActivsEr
ror;
                for(int neuronInLayer =
0;neuronInLayer<this.NET_STRUCTURE[layerIndex];neuronInLayer++){
                    indexWeightErrorChange=indexForAllWhenDoingWeightsAn
dError+1+neuronInLayer;//first index of weight
                    sumOfErrorAndWeights=0;
                    for(int
neuronInLayerAfter=0;neuronInLayerAfter<this.NET_STRUCTURE[layerIndex+1]
;neuronInLayerAfter++){
                        sumOfErrorAndWeights=sumOfErrorAndWeights+(error
InLayerAbove[neuronInLayerAfter]*this.ALL_WEIGHTS_AND_BIASES[indexWeight
ErrorChange]);
                        indexWeightErrorChange=indexWeightErrorChange+1+
this.NET_STRUCTURE[layerIndex];
                    }
                    errorInLayer[neuronInLayer]=sumOfErrorAndWeights*act
ivationDerivative(this.nodeValuePreActivation[indexForActivsError]);
                    indexForActivsError++;
                }
            }
            //errorInLayer has been generated
            totalAllExplored=0;
            for(int layer = this.NET_STRUCTURE.length-
1;layer>layerIndex-1;layer=layer-1){
                totalAllExplored=totalAllExplored +
(this.NET_STRUCTURE[layer]*this.NET_STRUCTURE[layer-
1]+this.NET_STRUCTURE[layer]);
            }
            indexForAllGrad=this.ALL_WEIGHTS_AND_BIASES.length-
totalAllExplored;//first all of biases in this layer
            arrayForIndexForAllGrad[layerIndex]=indexForAllGrad;
            totNeuronsExplored=0;
```

```
            for(int layer=0;layer<layerIndex-1;layer++){//first index of
activs in layer before
                totNeuronsExplored=totNeuronsExplored+this.NET_STRUCTURE
[layer];
            }
            arrayForTotNeuronsExplored[layerIndex]=totNeuronsExplored;
            for(int
neuronInLayer=0;neuronInLayer<this.NET_STRUCTURE[layerIndex];neuronInLay
er++){
                grad[indexForAllGrad]=errorInLayer[neuronInLayer];//sets
grad for bias;
                indexForAllGrad++;
                indexForActivsGrad=totNeuronsExplored;
                for(int
neuronInLayerBefore=0;neuronInLayerBefore<this.NET_STRUCTURE[layerIndex-
1];neuronInLayerBefore++){
                    grad[indexForAllGrad]=errorInLayer[neuronInLayer]*(t
his.nodeValuePostActivation[indexForActivsGrad]);
                    indexForActivsGrad++;
                    indexForAllGrad++;
                }
            }
            errorInLayerAbove=new double[errorInLayer.length];
            for(int i =0;i<errorInLayer.length;i++){
                errorInLayerAbove[i]=errorInLayer[i];
            }
        }
        //the backpropogation mock trial run is finished and now the
indexes accessed are stored to avoid needless calculations when
backpropogation is carried out during the training phase
        this.INDEX_FOR_ALL_WEIGHTS_ERROR=arrayIndexForAllWhenDoingWeight
sAndError;
        this.INDEX_FOR_ACTIVS_ERROR=arrayForIndexForActivsError;
        this.CURRENT_ALL=arrayForIndexForAllGrad;
        this.TOT_NEURON=arrayForTotNeuronsExplored;
    }
    /**
     * Loads the best neural network with the architecture of the
network that is stored in the network info text file
     * @return The weights and biases for the best network.
     */
    public double[]loadBestNetWithThisArchitecture(){
        double []bestNet=new double[this.ALL_WEIGHTS_AND_BIASES.length];
        double bestAcc=-0.5;
        int lineNoEnd;
        int lineNoStart=0;
        double currentAcc=0;
        boolean isBest=true;
        boolean isSameArch=false;
        try{
            //loads text file into memory
            FileReader read = new FileReader(this.FILE_NAME);
            BufferedReader buff = new BufferedReader(read);
```

```
            Object[]a=buff.lines().toArray();
            String []s = new String[a.length];
            for(int i=0;i<a.length;i++){
                s[i]=String.valueOf(a[i]);
            }
            //loops through evert line of the text file
            for(int i=0;i<s.length;i++){
                if(s[i].contains("Network Structure")){//checks to see
if line is describing neural network structure
                    isSameArch=false;
                    String str="Network Structure ";
                    for(int
index=0;index<this.NET_STRUCTURE.length;index++){
                        str=str+String.valueOf(this.NET_STRUCTURE[index]
)+"  ";
                    }
                    if(s[i].equals(str)){//checks to see whether the
architecture is the same as the current network
                        isSameArch=true;
                    }
                }
                if(s[i].contains("Best network (with
accuracy")&&isSameArch==true){//checks to see if new network is
appropriate
                    isBest=false;
                    String num="";
                    //gets accuracy of best network
                    for(int c=0;c<s[i].length();c++){
                        boolean isNumOrDot=true;
                        try{
                            double
f=Double.parseDouble(String.valueOf((s[i].charAt(c))));
                        }catch(NumberFormatException e){
                            if(s[i].charAt(c)!='.'){
                                isNumOrDot=false;
                            }
                        }
                        if(isNumOrDot==true){
                            num =num+s[i].charAt(c);
                        }
                    }
                    currentAcc=Double.parseDouble(num);
                    if(currentAcc>bestAcc&&isSameArch==true){//updates
best accuracy if appropriate
                        bestAcc=currentAcc;
                        isBest=true;
                    }
                    if(isBest==true&&isSameArch==true){//stores network
as best network if appropraite
                        lineNoStart=i+1;
                        lineNoEnd=i+bestNet.length;
                        int bestIn=0;
                        for(int
```

```
element=lineNoStart;element<=lineNoEnd;element++){
                                bestNet[bestIn]=Double.parseDouble(s[element
]);
                                bestIn++;
                            }
                        }
                    }
                }
            }catch(IOException e){
                System.out.println("errro "+e+" with file "+FILE_NAME);
                e.printStackTrace();
            }
            return bestNet;
        }
        /**
         * Loads the array stored in the best net text file, assuming it is
intended for a network with the architecture of the current network.
         * @return The weights and biases array
         * @throws Exception An exception may occur if the best net file
does not match the network structure.
         */
        public double[]loadBestNetFile() throws Exception{
            double[]best = new double[ALL_WEIGHTS_AND_BIASES.length];
            FileReader read = new FileReader(BEST_NET_FILE_NAME);
            BufferedReader buffRead = new BufferedReader(read);
            int index=0;
            String lineRead;
            while((lineRead=buffRead.readLine())!=null){
                best[index]=Double.parseDouble(lineRead);index++;
            }
            buffRead.close();read.close();
            return best;
        }
        /**
         * This is the hyperbolic tangent function.
         * @param x The input
         * @return the output of tanH
         */
         private static double tanH(double x){
             return ((Math.pow(Math.E, x)-Math.pow(Math.E,-
x))/(Math.pow(Math.E,x)+Math.pow(Math.E,-x)));
         }
         /**
          * This calculates the the first order derivative of tanH
          * @param x The input
          * @return The output
          */
         private static double gradTanH(double x){
             return(1.0-(Math.pow(tanH(x),2)));
         }
         /**
          * This function executes the reLu function
          * @param x The input
```

```
      * @return The output
      */
     private static double reLU(double x){
         if(x>0.0){
             return x;
         }else{
             return 0.0;
         }
     }
     /**
      * This calculates the first oder derivative of reLu. Note
((dy/dx)reLU(0)) is handled as 0 (although reLU is not differentiable at
x = 0)
      * @param x The input
      * @return The output
      */
     private static double gradRELU(double x){
         if(x>0.0){
             return 1.0;
         }else{
             return 0.0;
        }
     }
     /**
      * Performs the softmax function on an input vector
      * @param input The input vector
      * @return The output vector
      */
     private static double[]softmax(double[]input){
         double sum=0.0;//calculates the sum of the e^x for each element
in the array
         double[]ret=new double[input.length];//stores the vector to
return
         double[]exp=new double[input.length];
         for(int i =0;i<input.length;i++){
             exp[i]=Math.exp(input[i]);
             sum =sum+exp[i];
         }
         for(int i =0;i<input.length;i++){
             ret[i]=exp[i]/sum;
         }
         return ret;
     }
     /**
      * This method calculates the gradient of each parameter with
respect to the loss function. This is an optimised implementation of the
backpropagation algorithm. This method assumes that the training item
has already been fed through the network.
      * @param desiredOutput The desired output of the neural network
      * @return The gradient of each parameter's error with respect to
the loss function.
      */
    private double[]calcGradientAttemptOptimised(double[]desiredOutput){
```

```
        double[]errorInLayer= new double[1];//stores the error in the
current layer
        double[]errorInLayerAbove=new double[1];//stores the error in
the layer above
        double sumOfErrorAndWeights;//stores the sum of the error
multiplied by the weigths in a give layer
        double grad[]=new double[this.ALL_LENGTH];//stores the error
gradient for each weight and bias in the network
        //stores various indices used during the backpropogation process
        int indexWeightErrorChange;
        int indexForActivsError;
        int indexForAllGrad;
        int changeD;
        int totNeuronsExplored;
        int indexForActivsGrad;
        int activIndex;
        for(int
layerIndex=this.NET_STRUCTURE_LENGTH_MINUS_ONE;layerIndex>0;layerIndex--
){//loops through the entire network
            errorInLayer=new
double[this.NET_STRUCTURE[layerIndex]];//stores the error of each node
in the given layer
            if(layerIndex==this.NET_STRUCTURE_LENGTH_MINUS_ONE){//for
the output layer, the error of each neuron is calculated
                activIndex=this.ACTIV_IN;
                for(int
neuronInLayer=0;neuronInLayer<this.NET_STRUCTURE[layerIndex];neuronInLay
er++){
                    errorInLayer[neuronInLayer]=
this.nodeValuePostActivation[activIndex] -
desiredOutput[neuronInLayer];//this calcualtes the error of the network
when a softmax activation is sued in conjunction with categorical cross
entropoy loss
                    //errorInLayer[neuronInLayer]=(this.nodeValuePostAct
ivation[activIndex]-
desiredOutput[neuronInLayer])*gradTanH(this.nodeValuePreActivation[activ
Index]);//this is the method used to calculate the error of the network
when using the mean squared error loss
                    activIndex++;
                }
            }else{
                indexForActivsError=this.INDEX_FOR_ACTIVS_ERROR[layerInd
ex];
                changeD=1+this.NET_STRUCTURE[layerIndex];
                //loops through each neuron in the current layer and
calculates the error of the weights and biase of the neurons
                for(int neuronInLayer =
0;neuronInLayer<this.NET_STRUCTURE[layerIndex];neuronInLayer++){
                    if(NEURONS_DROPPED[neuronInLayer+INDEX_CACHE[layerIn
dex]]){
                        errorInLayer[neuronInLayer]=0;
                        indexForActivsError++;
                        continue;
```

```
                    }
                    indexWeightErrorChange=this.INDEX_FOR_ALL_WEIGHTS_ER
ROR[layerIndex]+1+neuronInLayer;//first index of weight
                    sumOfErrorAndWeights=0;//sums the weights mulitplied
by the error in the layer above
                    for(int
neuronInLayerAfter=0;neuronInLayerAfter<this.NET_STRUCTURE[layerIndex+1]
;neuronInLayerAfter++){
                        sumOfErrorAndWeights=sumOfErrorAndWeights+(error
InLayerAbove[neuronInLayerAfter]*this.ALL_WEIGHTS_AND_BIASES[indexWeight
ErrorChange]);
                        indexWeightErrorChange=indexWeightErrorChange+ch
angeD;
                    }
                    errorInLayer[neuronInLayer]=sumOfErrorAndWeights*act
ivationDerivative(this.nodeValuePreActivation[indexForActivsError]);//st
ores the error of the neuron by muliplying the sum of the error and
wieghts for the neuron by the pre activation node input
                    indexForActivsError++;
                }
            }
            //errorInLayer has been generated - now the error of each
bias and weight needs to be determined
            indexForAllGrad=this.CURRENT_ALL[layerIndex];
            for(int
neuronInLayer=0;neuronInLayer<this.NET_STRUCTURE[layerIndex];neuronInLay
er++){//loops through all neurons in layer
                if(errorInLayer[neuronInLayer]==0){
                    grad[indexForAllGrad]=0;indexForAllGrad++;
                    for(int
neuronInLayerBefore=0;neuronInLayerBefore<this.NET_STRUCTURE[layerIndex-
1];neuronInLayerBefore++){//generates error for weights
                        grad[indexForAllGrad]=0;//calculates error for
weight
                        indexForAllGrad++;
                    }
                    continue;
                }
                grad[indexForAllGrad]=errorInLayer[neuronInLayer];//sets
grad for bias (which is just equal to the error of the node)
                indexForAllGrad++;
                indexForActivsGrad=this.TOT_NEURON[layerIndex];//stores
the index to use for POST_ACTIVATION array access
                for(int
neuronInLayerBefore=0;neuronInLayerBefore<this.NET_STRUCTURE[layerIndex-
1];neuronInLayerBefore++){//generates error for weights
                    grad[indexForAllGrad]=errorInLayer[neuronInLayer]*(t
his.nodeValuePostActivation[indexForActivsGrad]);//calculates error for
weight
                    indexForActivsGrad++;
                    indexForAllGrad++;
                }
            }
```

```
            if(layerIndex==1){//added in for optimisation
                return grad;
            }
            //sets the error in the current layer as the error in the
layer above (this is used when calculating the error of the next layer)
            errorInLayerAbove=new double[errorInLayer.length];
            for(int i =0;i<errorInLayer.length;i++){
                errorInLayerAbove[i]=errorInLayer[i];
            }
        }
        return grad;//returns the gradients
    }
    /**
     * Shuffles the training data array.
     * @param array The array to shuffle
     */
     private static void shuffleTrainingDataArray(int[][]array){
        Random rnd = new Random();int in;
        int buff;
        for(int y=0;y<array.length;y++){//loops through every row in
array and swaps it with another row randomly
            in = rnd.nextInt(array.length);
            for(int x=0;x<array[0].length;x++){
                buff=array[y][x];
                array[y][x]=array[in][x];
                array[in][x]=buff;
            }
        }
        //as objects are passed by reference in java, the array is
shuffled without needing to return it
    }
     /**
      * Trains the neural network on a dataset using backpropagation in
conjuction with the ADAM optimiser.
      * @param LEARNING_RATE The learning rate of the network - this is
recommended to be 0.001 as the ADAM optimiser is used.
      * @param NO_OF_EPOCHS The number of epochs in the training cycle
(each epoch is one complete pass of the dataset).
      * @param SAMPLE_SIZE The batch size - the number of training items
used per network update.
      * @param TOTAL_TRAIN_SIZE The total size of the training data
      * @param PRINT_AND_TEST_AFTER_THIS_MANY_ITERATIONS How many
iterations to test the accuracy and print it. Each iteration is one
parameter update of the network.
     * @param DROPOUT_RATE The dropout rate of each neuron
      */
     public void trainNew( final double LEARNING_RATE, final int
NO_OF_EPOCHS,final int SAMPLE_SIZE,final int TOTAL_TRAIN_SIZE,final int
PRINT_AND_TEST_AFTER_THIS_MANY_ITERATIONS, final double DROPOUT_RATE){
        double changeBy[];//stores how much to change each parameter by
        final long TIME_BEFORE=System.currentTimeMillis();//used to
calculate how long the network has been running for
        double timeTemp = System.currentTimeMillis();//used to calculate
```

```
the time between each testing of the network
        final int
NO_OF_ITERATIONS_IN_ONE_EPOCH=(TOTAL_TRAIN_SIZE/SAMPLE_SIZE);//stores
number of iterations per epoch
        final double
SAMPLE_SIZE_DOUBLE=Double.parseDouble(String.valueOf(SAMPLE_SIZE));
        double[]bestNet = new double[this.ALL_LENGTH];//stores the best
network configuration
        double accuracy;//stores the accuracy of the network
        double time = System.currentTimeMillis();
        double mT[]=new double[this.ALL_LENGTH];//stores the mT values -
part of the ADAM optimisation formula
        double vT[]=new double[this.ALL_LENGTH];//stores the vT values -
part of the ADAM optimisation formula
        double mHatT[]=new double[this.ALL_LENGTH];//stores the mHatT
values - part of the ADAM optimisation formula
        double vHatT[]=new double[this.ALL_LENGTH];//stores the vHatT
values - part of the ADAM optimisation formula
        double prevMT[]=new double[this.ALL_LENGTH];// stores the
pervious MT values - part of the ADAM optimisation formula
        double prevVT[]=new double[this.ALL_LENGTH];// stores the
pervious VT values - part of the ADAM optimisation formula
        double t =0.0;//stores the time step - also used in the ADAM
forumla
        try{//checks to see if any previous learning iterations have
been run with this model and allows for ADAM training to continue
            double l =this.previousVHat[0];
            for(int i =0;i<this.previousMHat.length;i++){
                prevMT[i]=this.previousMHat[i];
                prevVT[i]=this.previousVHat[i];
            }
            t=this.timeStep;
        }catch(java.lang.NullPointerException e){
        }
        //stores constants used in ADAM optimisation method
        final double BETA_ONE=0.9;
        final double BETA_TWO=0.999;
        Random rnd = new Random();
        final double EPSILON = Math.pow(10,-8);
        double gradOfOne[];//stores the gradient of one item of training
data
        int indexOfTrain=0;//stores the index that accesses the
appropriate training data
        int testData[][]=
PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain.getTestData();//laods
the test data
        int[][]trainingData=PhilipMCS5SoftwareDevelopmentNeuralNetworkTr
ain.getTrainingData();//loads the training data
        int indexPrintAndTest=1;//stores the the number of iterations
since the previous print of accuracy
        double start[]=this.testAccuracy(testData);//calculates the
initial network accuracy before any training
        double highestAcc=start[0];//stores the highest network accuracy
```

```
        for(int i =0;i<this.ALL_LENGTH;i++){
            bestNet[i]=this.ALL_WEIGHTS_AND_BIASES[i];//initialises the
best network as the current network
        }
        double desiredOut[]= new double[2];//stores the desired output
of the neural network
        double[]inputToNet=new double[this.NET_STRUCTURE[0]];//stores
the array used to make an input to the net
        int indexData=0;//stores the index of training data to be
accessed within an iteration
        System.out.println("Accuracy before training "+highestAcc+" %"+"
loss of "+start[1]);
        for(int epoch=1;epoch<=NO_OF_EPOCHS;epoch++){//performs the
specified number of epochs
                shuffleTrainingDataArray(trainingData);//shuffles the
training data at the start of every epoch
            indexOfTrain=0;
            for(int
iteration=1;iteration<=NO_OF_ITERATIONS_IN_ONE_EPOCH;iteration++){//perf
orms every iteration in the epoch
                changeBy = new double[this.ALL_LENGTH];//sets change by
to an empty array of the appropriate length
                indexData=indexOfTrain;
                indexOfTrain=indexOfTrain+SAMPLE_SIZE;
                for (int sampleNo = 0; sampleNo < SAMPLE_SIZE;
sampleNo++) {//loops thorugh every data sample in the iteration
                    for(int i=0;i<this.NET_STRUCTURE[0];i++){//loads the
network input
                        inputToNet[i]=trainingData[indexData][i+2];
                    }
                    //drops out neurons for regularisation if
appropriate
                    for(int
node=0;node<LENGTH_OF_NEURONS_DROPPED_ARRAY;node++){
                        NEURONS_DROPPED[node]=rnd.nextFloat()<DROPOUT_RA
TE;
                    }
                    this.feedThroughNet(inputToNet);//feeds the input
through the network
                    desiredOut[0]=trainingData[indexData][0];//stores
the desired network output
                    desiredOut[1]=trainingData[indexData][1];
                    indexData++;
                    gradOfOne =
this.calcGradientAttemptOptimised(desiredOut);//calculates the error of
each parameter
                    for (int i = 0; i <
this.ALL_WEIGHTS_AND_BIASES.length; i++) {//stores how much each
paramater should be changed by
                        changeBy[i]=changeBy[i]+gradOfOne[i];
                    }
                }
                t=t+1.0;//increments the time step - used in the ADAM
```

```
formula
                for(int i=0;i<changeBy.length;i++){//updates all weights
and biases using the ADAM optimisation technique
                    changeBy[i]=changeBy[i]/SAMPLE_SIZE_DOUBLE;//change
by is normalised to reflect the sample size
                    //the ADAM optimisation formula is implemented
                    mT[i]=BETA_ONE*prevMT[i]+(1.0-BETA_ONE) *
changeBy[i];
                    vT[i]=BETA_TWO*prevVT[i]+(1.0-
BETA_TWO)*(changeBy[i]*changeBy[i]);
                    mHatT[i]=mT[i]/(1.0-Math.pow(BETA_ONE,t));
                    vHatT[i]=vT[i]/(1.0-Math.pow(BETA_TWO,t));
                    prevMT[i]=mT[i];
                    prevVT[i]=vT[i];
                    this.ALL_WEIGHTS_AND_BIASES[i]=this.ALL_WEIGHTS_AND_
BIASES[i] - (LEARNING_RATE*mHatT[i]/(Math.sqrt(vHatT[i])+EPSILON));//the
network values are configured appropriately
                   //this.allWeightsAndBiases[i]=this.allWeightsAndBiase
s[i]-(changeBy[i]*learningRate); // this method is the commented out but
is the gradient descent formula
                }
                if(indexPrintAndTest==PRINT_AND_TEST_AFTER_THIS_MANY_ITE
RATIONS){//checks to see if they system should test the network and
print the result of this
                    //tests network, displays this and updates the best
accuracy and best network variables if appropriate
                    System.out.println("Time training between training
"+((System.currentTimeMillis()-time)/1000.0)+" seconds");
                    //ensures that no nodes are dropped when testing
accuracy
                    for(int n=0;n<LENGTH_OF_NEURONS_DROPPED_ARRAY;n++){
                        NEURONS_DROPPED[n]=false;
                    }
                    double[]accAndLoss=this.testAccuracy(testData);
                    time = System.currentTimeMillis();
                    accuracy = accAndLoss[0];
                    if (accuracy >= highestAcc || highestAcc <= 0) {
                        for (int i = 0; i <
this.ALL_WEIGHTS_AND_BIASES.length; i++) {
                            bestNet[i] = this.ALL_WEIGHTS_AND_BIASES[i];
                        }
                        highestAcc = accuracy;
                    }
                    System.out.println("Accuracy " + accuracy + "% after
" + epoch + " epochs out of " + NO_OF_EPOCHS + " epochs. Sample size of
" + SAMPLE_SIZE + " with a learning rate of " + LEARNING_RATE+" loss of
"+accAndLoss[1]+ "iteration "+(iteration)+" out of
"+NO_OF_ITERATIONS_IN_ONE_EPOCH+" best accuracy of "+highestAcc);
                    indexPrintAndTest=0;
                    if((System.currentTimeMillis()-
timeTemp)/1000.0>=36000){//this clause is used to ensure that the
network only takes a certain amount of time to train
                        System.out.println("ending training as time has
```

```
elapsed "+ ((System.currentTimeMillis()-timeTemp)/1000.0));
                        epoch=NO_OF_EPOCHS+1;
                        break;
                    }
                }
                indexPrintAndTest++;
            }
        }
        //stores the values of the ADAM technique - this allows the same
neural network to call the train method multiple times
        this.timeStep=t;
        this.previousMHat=new double[this.ALL_LENGTH];
        this.previousVHat=new double[this.ALL_LENGTH];
        for(int i =0;i<prevMT.length;i++){
            this.previousMHat[i]=prevMT[i];
            this.previousVHat[i]=prevVT[i];
        }
        //calclates the final network accuracy
        accuracy = this.testAccuracy(testData)[0];
        System.out.println("Accuracy "+accuracy+" with loss of
"+this.testAccuracy(testData)[1]);
        //writes all network details to a text file
        long timeAtEnd = System.currentTimeMillis();
        long timeTaken=timeAtEnd-TIME_BEFORE;
        try{
            FileWriter writer = new FileWriter(FILE_NAME,true);
            BufferedWriter bufferedWriter = new BufferedWriter(writer);
            bufferedWriter.write("----------------------------");
            bufferedWriter.newLine();
            bufferedWriter.write("Network Structure ");
            for(int i =0;i<this.NET_STRUCTURE.length;i++){
                bufferedWriter.write(String.valueOf(this.NET_STRUCTURE[i
])+"  ");
            }
            bufferedWriter.newLine();
            bufferedWriter.write("All weights and biases");
            bufferedWriter.newLine();
            for(int i =0;i<this.ALL_WEIGHTS_AND_BIASES.length;i++){
                bufferedWriter.write(String.valueOf(this.ALL_WEIGHTS_AND
_BIASES[i]));
                bufferedWriter.newLine();
            }
            bufferedWriter.write("No of epochs "+NO_OF_EPOCHS);
            bufferedWriter.newLine();
            bufferedWriter.write("Final accuracy of "+accuracy);
            bufferedWriter.newLine();
            bufferedWriter.write("time taken in milliseconds
"+timeTaken);
            bufferedWriter.newLine();
            bufferedWriter.write("Learning rate of "+LEARNING_RATE);
            bufferedWriter.newLine();
            bufferedWriter.write("Best network (with accuracy
"+highestAcc+"% with this sturcture had a configuration of");
```

```
            bufferedWriter.newLine();
            for(int i=0;i<bestNet.length;i++){
                bufferedWriter.write(String.valueOf(bestNet[i]));
                bufferedWriter.newLine();
            }
            bufferedWriter.newLine();
            bufferedWriter.write("Using a sample size of "+SAMPLE_SIZE+"
for each epoch out of a total of 60000 samples. Note this doesnt
neccesarly apply if this is true:  false");
            bufferedWriter.newLine();
            bufferedWriter.write("----------------------------");
            bufferedWriter.close();
            writer.close();
        }catch(IOException e){
            System.out.println("error with file "+FILE_NAME+" error of
"+e);
            System.out.println("Error trace");
            e.printStackTrace();
        }
    }
     /**
      * Gets the training data
      * @return The training data
      */
    private static int[][]getTrainingData(){
        return
PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain.getTrainingData();
    }
    /**
     * Gets the test data
     * @return The test data
     */
    private static int[][]getTestData(){
        return
PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain.getTestData();
    }
    /**
     * Tests the network's accuracy
     * @param testData The test data to use when testing accuracy
     * @return The accuracy and loss of the system as an array of length
2. Element 0 corresponds to the accuracy, whilst element 1 is the loss.
     */
    public double[]testAccuracy(int testData[][]){
        double time = System.currentTimeMillis();//used to calculate how
long it takes to test the system
        int totalCorrect=0;double[]out;
        double[]desiredOut = new double[2];
        double averageLoss=0.0;int greatestIndex;double err;
        double diff;
        double[]netInput=new
double[PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain.WIDTH_OF_DATA-
2];//stores the network input
        for(int y=0;y<testData.length;y++){//loops trhough every item in
```

```
the test dataset
            desiredOut[0]=testData[y][0];//gets the desired output of
the neural network
            desiredOut[1]=testData[y][1];
            for(int x=2;x<testData[0].length;x++){//generates the input
to the neural network
                netInput[x-2]=testData[y][x];
            }
            out=this.feedThroughNet(netInput);//passes the network input
throught the neural network
            greatestIndex=0;//determines whether out[0] or out[1] is
biggest
            if(out[1]>out[0]){
                greatestIndex=1;
            }
            if(desiredOut[greatestIndex]==1.0){//checks to see if
network is correct
                totalCorrect++;
            }
           /* diff=out[0]-desiredOut[0];
            err=diff*diff;*/
           err=0;
            //calculates the loss of the network
            for(int i=0;i<out.length;i++){
                err = err+(desiredOut[i]*Math.log(out[i]));//categorical
cross entropy loss is employed -  this makes the assumption that the
desired output is a one hot vector

            }err=err*-1.0;
            averageLoss=averageLoss+(err);//adds loss to total loss
        }

        double accuracy =
(double)totalCorrect/(double)PhilipMCS5SoftwareDevelopmentNeuralNetworkT
rain.NO_OF_TEST_BOARDS;//calculates the accuracy of the network
        accuracy= accuracy*100.0;
        averageLoss=averageLoss/(double)PhilipMCS5SoftwareDevelopmentNeu
ralNetworkTrain.NO_OF_TEST_BOARDS;//calculates the average loss
        System.out.println("Testing time:
"+((System.currentTimeMillis()-time)/1000.0)+" seconds . Total correct:
"+totalCorrect+" out of
"+PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain.NO_OF_TEST_BOARDS);
        double ret[]={accuracy,averageLoss};
        return ret;
    }
    /**
     * This feeds a desired input through the neural network and returns
the output. This is sometimes referred to as forward propagation.
     * @param input The input array to the network
     * @return The neural network output
     */
    public double[]feedThroughNet(double[]input){
        int currentAllIndex=0;//stores the current index used to access
```

```
all weights and biases of the neural network
        double []nodeValuesAfterActivation;//stores the activations in
the current layer
        double []out = new
double[this.NET_STRUCTURE[this.NET_STRUCTURE_LENGTH_MINUS_ONE]];//stores
the output of the neural network
        int indexForNodesInThisLayer;//stores the index used to access
the nodes in a given layer
        double valueOfBias;//stores the value of the current bias
        double nodeValPreActivation;//stores the value of the node
before the activation function is applied to it
        double[]nodeValuesAfterActivationPrevLayer=input;//stores the
value of activated nodes in the previous layer
        int indexForPreActivs=0;//stores the index for pre activation
arrays
        int indexForPostActivs=0;//stores the index for post actviation
array
        this.nodeValuePreActivation=new double[this.TOTAL_NEURONS];
        this.nodeValuePostActivation=new double[this.TOTAL_NEURONS];
        for(int i=0;i<this.NET_STRUCTURE[0];i++){//sets values for the
input layer - all nodes in input layer = input[i]
            this.nodeValuePreActivation[indexForPreActivs]=input[i];
            this.nodeValuePostActivation[indexForPostActivs]=input[i];
            indexForPostActivs++;
            indexForPreActivs++;
        }
        for(int
layerIndex=1;layerIndex<this.NET_STRUCTURE_LENGTH;layerIndex++){//loops
through network and calculates values of each layer
            indexForNodesInThisLayer=0;
            nodeValuesAfterActivation=new
double[this.NET_STRUCTURE[layerIndex]];
            for(int
indexOfLayer=0;indexOfLayer<this.NET_STRUCTURE[layerIndex];indexOfLayer+
+){//neurons in current layer
                if(layerIndex!=NET_STRUCTURE_LENGTH_MINUS_ONE&&NEURONS_D
ROPPED[INDEX_CACHE[layerIndex]+indexOfLayer]){//check to see if node is
dropped out
                    currentAllIndex=currentAllIndex+1+this.NET_STRUCTURE
[layerIndex-1];
                    this.nodeValuePreActivation[indexForPreActivs]=0;ind
exForPreActivs++;
                    this.nodeValuePostActivation[indexForPostActivs]=0;i
ndexForPostActivs++;
                    nodeValuesAfterActivation[indexForNodesInThisLayer]=
0;
                    indexForNodesInThisLayer++;
                    continue;
                }
                valueOfBias=this.ALL_WEIGHTS_AND_BIASES[currentAllIndex]
;//stores bias of node
                currentAllIndex++;
                nodeValPreActivation=0;
```

```
                for(int
prevLayerIndex=0;prevLayerIndex<this.NET_STRUCTURE[layerIndex-
1];prevLayerIndex++){//sums wieghts and previous node values to get
preActivation value
                    nodeValPreActivation=nodeValPreActivation +
nodeValuesAfterActivationPrevLayer[prevLayerIndex] *
this.ALL_WEIGHTS_AND_BIASES[currentAllIndex];
                    currentAllIndex++;

                }
                this.nodeValuePreActivation[indexForPreActivs]=nodeValPr
eActivation+valueOfBias;//sets value before actviation in this layer
                indexForPreActivs++;
                if(layerIndex!=this.NET_STRUCTURE_LENGTH_MINUS_ONE){//se
ts  the post activation value
                    nodeValuesAfterActivation[indexForNodesInThisLayer]=
activation(this.nodeValuePreActivation[indexForPreActivs-1]);//for non
output layers the choosen actviation is used - normally this is reLu
                    indexForNodesInThisLayer++;
                    this.nodeValuePostActivation[indexForPostActivs]=nod
eValuesAfterActivation[indexForNodesInThisLayer-1];
                    indexForPostActivs++;
                }else if(indexOfLayer==this.NET_STRUCTURE[layerIndex]-
1){//for the output layer, the softmax activation is used
                    double preSigLayer[]=new
double[this.NET_STRUCTURE[layerIndex]];
                    for(int
indexVal=0;indexVal<preSigLayer.length;indexVal++){
                        preSigLayer[indexVal]=this.nodeValuePreActivation
[this.TOTAL_NEURONS_MINUS_LAST_LAYER+indexVal];
                    }
                    nodeValuesAfterActivation=softmax(preSigLayer);
                    for(int i=0;i<nodeValuesAfterActivation.length;i++){
                        this.nodeValuePostActivation[indexForPostActivs]
=nodeValuesAfterActivation[i];
                        indexForPostActivs++;
                    }
                }
                /*else{
                    nodeValuesAfterActivation[indexForNodesInThisLayer]=
tanH(this.nodeValuePreActivation[indexForPreActivs-1]);//tanh used for
output layer of non classification model
                    indexForNodesInThisLayer++;
                    this.nodeValuePostActivation[indexForPostActivs]=nod
eValuesAfterActivation[indexForNodesInThisLayer-1];
                    indexForPostActivs++;
                }*/
            }
            nodeValuesAfterActivationPrevLayer=nodeValuesAfterActivation
;
            if(layerIndex==this.NET_STRUCTURE_LENGTH_MINUS_ONE){
                out = nodeValuesAfterActivation;
            }
```

```
        }
        //returns the network output
        return out;
    }
    /**
     * Sets all weights and biases in the network to the chosen value
     * @param allWAndB The new set of weights and biases
     */
    public void setAllWeightsAndBiases(double[]allWAndB){
        if(allWAndB.length!=this.ALL_WEIGHTS_AND_BIASES.length){
            System.out.println("Attempted to change weights and biases
to an array of different length ");
        }
        for(int i =0;i<allWAndB.length;i++){
            this.ALL_WEIGHTS_AND_BIASES[i]=allWAndB[i];
        }
    }
    /**
     * Uses the specified activation function. Currently this is reLU.
     * @param x The input
     * @return The return
     */
    private static double activation(double x){
        if(x>0.0){
            return x;
        }else{
            return 0.0;
        }
    }
    /**
     * Returns the gradient of the chosen activation. Currently this
returns the gradient for reLU.
     * @param x The input
     * @return The return value
     */
    private static double activationDerivative(double x){
       if(x>0.0){
            return 1.0;
        }else{
            return 0.0;
        }
    }
    /**
     * Gets a random number between one and minus one using the xavier
technique.
     * @param numberOfNeuronsInLayerBelow The number of neurons in the
layer below
     * @param numberOfNeuronsInLayer The number of neurons in the
current layer
     * @return The random number
     */
    private double getRandDoubleBetweenOneAndMinusOne(int
numberOfNeuronsInLayerBelow,int numberOfNeuronsInLayer){
```

```
        return
RAND.nextGaussian()*Math.sqrt(2.0/((double)(numberOfNeuronsInLayer+numbe
rOfNeuronsInLayerBelow)));
       // return
(rnd.nextGaussian()*Math.sqrt(1.0/(double)numberOfInputsForWeightsInThis
Layer)); //xavier init for tanh/sigmoid
    }
    /**
     * Gets the best network with this structure and loads it to a text
file
     */
    public void writeBestNetToBestFile(){
        double[]bestNet=this.loadBestNetWithThisArchitecture();
        try{
            FileWriter w = new FileWriter(BEST_NET_FILE_NAME,false);
            BufferedWriter buffWrite = new BufferedWriter(w);
            buffWrite.write(String.valueOf(bestNet[0]));
            for(int i=1;i<bestNet.length;i++){
                buffWrite.newLine();
                buffWrite.write(String.valueOf(bestNet[i]));
            }
            buffWrite.flush();w.flush();
            buffWrite.close();w.close();
        }catch(IOException e){
            System.out.println("An unexpected error occured "+e);
        }
    }
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.neural.network.train;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;

/**
 * This is the main class used to train my chess neural network.
 * @author mortimer
```

```
 */
public class PhilipMCS5SoftwareDevelopmentNeuralNetworkTrain {
    private static final int TOTAL_AMOUNT_OF_DATA=2326888;//stores total
data items
    public static final int NO_OF_TEST_BOARDS=80000;//stores the number
of boards used in test data
    private static final int
NO_OF_TRAINING_BOARDS=2246888/*TOTAL_AMOUNT_OF_DATA-
NO_OF_TEST_BOARDS*/;//stores number of boards used in training data
    public static final int WIDTH_OF_DATA=779;//stores the width of each
data item
    private static final String TEST_FILE_NAME="testData.txt";//stores
name of test data text file
    private static final String TRAIN_FILE_NAME="trainData.txt";//stores
name of training data file
    private static final String SPLIT=",";//stores character used to
split items in file
    private static final String LOSE_FILE_NAME="losestates.txt";//stores
file name for losing states
    private static final String WIN_FILE_NAME="winstates.txt";//stores
file name for winning states
    private static final String DRAW_FILE_NAME="drawstates.txt";//stores
file name for drawing states
    private static final int NO_OF_DRAWS=485843*4;//stores number of
draws in dataset
    private static final int NO_OF_WINS=414929*4;//stores number of wins
in dataset
    private static final int NO_OF_LOSSES=305861*4;//stores number of
losses in dataset
    private static final int NO_OF_WINS_WHITE=208767;
    private static final int NO_OF_WINS_BLACK=206162;
    private static final int NO_OF_LOSE_BLACK=152503;
    private static final int NO_OF_LOSE_WHITE=153358;

    /**
     * The main method used to train the neural network
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        NeuralNetwork net = new NeuralNetwork(new
int[]{777,300,200,100,2});//creates a neural network with this structure
        double a []=new double[777];
        double timeBefore=System.currentTimeMillis();
        for(int i=0;i<200000;i++){
            net.feedThroughNet(a);
        }
        System.out.println("Time to do 200,000:
"+(System.currentTimeMillis()-timeBefore)/1000.0);
        System.out.println("new system");
        double learningRate=0.001;int noOfEpochs=200000;int
batchSize=128;
        int totalSizeOfTrainingData=NO_OF_TRAINING_BOARDS;int
printAfterNoIterations =1000;double dropoutRate=0.0;
```

```
        net.trainNew(learningRate, noOfEpochs,
batchSize,  totalSizeOfTrainingData,printAfterNoIterations,dropoutRate);
//trains network
        net.writeBestNetToBestFile(); //writes best network to a text
file - this is used for the ChessNeuralNetwork Class
        try{
            //tests the accuracy of the chess neural network
            ChessNeuralNetwork c= new ChessNeuralNetwork(new
int[]{777,300,200,100,2});
            double[]aT=c.testAccuracy(getTestData());
            System.out.println(aT[0]+" "+aT[1]);
            net.setAllWeightsAndBiases(net.loadBestNetFile());//sets the
neural network to be the network with the best accuracy
        }catch(Exception e){
            System.out.println(e);
        }
        //tests the accuracy of the trained neural network
        double[] aT=net.testAccuracy(getTestData());
        System.out.println(aT[0]+" "+aT[1]);
    }
    /**
     * This method is used to generate a training and test data set
using stockfish (the world's strongest chess engine) as the ground truth
     * evaluation of a board state. Essentially, the code loops through
all board states stored and stores the stockfish evaluation along with
     * a computer friendly chess board representation. This code was not
used as the code would take too long to run (as stockfish needs a
minimum amount
     * of time to evaluate each board state with any accuracy). This
method would prove effective given more computer resources.
     */
    private static void formatDataForStockfish(){
        try{
            StockFish stock = new StockFish();
            ChessBoardState state;
            int arr[];
            String dat;
            String fens[];
            ArrayList <String> data = new ArrayList(100000);//stores the
data to be written to the text files
            FileReader read = new FileReader(WIN_FILE_NAME);
            BufferedReader buffRead =  new
BufferedReader(read);
            fens=buffRead.lines().toArray(String[]::new);
            buffRead.close();read.close();
            for(int i=0;i<fens.length;i++){
                //gets the value and computer version of each board
state -this process is repeated for winning, drawing and losing board
states
                state = new  ChessBoardState(fens[i]);
                arr=state.getNeuralNetInput();
                try{
                    dat =
```

```
String.valueOf(stock.getCurrentBoardValue(fens[i],state.IS_BLACK_TURN_TO
_PLAY_NEXT));
                }catch(Exception de){
                    continue;
                }
                for(int j=0;j<arr.length;j++){
                    dat=dat+","+String.valueOf(arr[j]);
                }
                if(i%1000==0&& i>1){

                    System.out.println("i "+i);
                }
                data.add(dat);
            }
            FileReader read2 = new FileReader(DRAW_FILE_NAME);
            BufferedReader buffRead2 =  new
BufferedReader(read2);
            fens=buffRead2.lines().toArray(String[]::new);
            buffRead2.close();read2.close();
            for(int i=0;i<fens.length;i++){
                state = new  ChessBoardState(fens[i]);
                arr=state.getNeuralNetInput();
                try{
                    dat =
String.valueOf(stock.getCurrentBoardValue(fens[i],state.IS_BLACK_TURN_TO
_PLAY_NEXT));
                }catch(Exception de){
                    continue;
                }
                for(int j=0;j<arr.length;j++){
                    dat=dat+","+String.valueOf(arr[j]);
                }
                if(i%1000==0&&i>1){

                    System.out.println("i "+i);
                }
                data.add(dat);
            }
            FileReader read3 = new FileReader(LOSE_FILE_NAME);
            BufferedReader buffRead3 =  new
BufferedReader(read3);
            fens=buffRead3.lines().toArray(String[]::new);
            buffRead3.close();read3.close();
            for(int i=0;i<fens.length;i++){
                state = new  ChessBoardState(fens[i]);
                arr=state.getNeuralNetInput();
                try{
                    dat =
String.valueOf(stock.getCurrentBoardValue(fens[i],state.IS_BLACK_TURN_TO
_PLAY_NEXT));
                }catch(Exception de){
                    continue;
                }
```

```
                for(int j=0;j<arr.length;j++){
                    dat=dat+","+String.valueOf(arr[j]);
                }
                if(i%1000==0&&i>1){

                    System.out.println("i "+i);
                }
                data.add(dat);
            }
            System.out.println("Size of data "+data.size());
            stock.exit();
            Collections.shuffle(data);//randomises data
            //writes data to file
            int maxTrainIndex=(data.size()/10)*9;
            FileWriter write = new FileWriter(TRAIN_FILE_NAME,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            for(int i=0;i<maxTrainIndex;i++){
                if(i!=0){
                    buffWrite.newLine();
                }
                buffWrite.write(data.get(i));
            }buffWrite.flush();write.flush();
            buffWrite.close();write.close();
            FileWriter w = new FileWriter(TEST_FILE_NAME,false);
            BufferedWriter buffW = new BufferedWriter(w);
            for(int i= maxTrainIndex;i<data.size();i++){
                if(i!=maxTrainIndex){
                    buffW.newLine();
                }
                buffW.write(data.get(i));
            }
            buffW.flush();w.flush();
            buffW.close();w.close();
        }catch(IOException e){
            System.out.println("Error "+e);
        }
    }
    /**
     * Gets neural network training data.
     * @return The training data
     */
    public static int[][]getTrainingData(){
        //reads the training data from the training data text file and
returns it as an array
        int w=0;int l=0;
        int data[][]=new int[NO_OF_TRAINING_BOARDS][WIDTH_OF_DATA];
        try{
            int i=0;
            String lineRead;String[]line;
            FileReader read = new FileReader(TRAIN_FILE_NAME);
            BufferedReader buffRead =new BufferedReader(read);
            while((lineRead=buffRead.readLine())!=null&&
i<NO_OF_TRAINING_BOARDS){
```

```
                line=lineRead.split(SPLIT);
                for(int ch=0;ch<WIDTH_OF_DATA;ch++){
                    data[i][ch]=Integer.parseInt(line[ch]);
                }
                if(data[i][0]==1){
                    w++;
                }else{
                    l++;
                }
                i++;
            }buffRead.close();
            read.close();
            System.out.println("W: "+w+" l: "+l);
        }catch(IOException e){
            System.out.println("Error "+e);
        }
        return data;
    }
    /**
     * Gets the test data from the test file.
     * @return The test data
     */
    public static int[][]getTestData(){
        //reads the test data from the test file and writes it to an
array
        int data[][]=new int[NO_OF_TEST_BOARDS][WIDTH_OF_DATA];
        int w=0;int l=0;
        try{
            int i=0;
            String lineRead;String[]line;
            FileReader read = new FileReader(TEST_FILE_NAME);
            BufferedReader buffRead =new BufferedReader(read);
            while((lineRead=buffRead.readLine())!=null&&
i<NO_OF_TEST_BOARDS){
                line=lineRead.split(SPLIT);
                for(int ch=0;ch<WIDTH_OF_DATA;ch++){
                    data[i][ch]=Integer.parseInt(line[ch]);
                }
                if(data[i][0]==1){
                    w++;
                }else{
                    l++;
                }
                i++;
            }buffRead.close();
            read.close();
            System.out.println("Win: "+w+" lose: "+l);
        }catch(IOException e){
            System.out.println("Error "+e);
        }
        return data;
    }
    /**
```

```
     * Formats data for a neural network comparing the value of boards.
The idea behind this was that the network would compare
     * two board states and determine which one was better. This process
was not used as it proved too inefficient computationally.
     */
    private static void formatDataCompare(){
        try{
            String[]data = new String[NO_OF_LOSSES];
            int index=0;
            int[]line;
            //loads win data
            FileReader read = new FileReader(WIN_FILE_NAME);
            BufferedReader buffRead = new BufferedReader(read);
            String lineRead;
            ChessBoardState state;
            while((lineRead=buffRead.readLine())!=null){
                state =new ChessBoardState(lineRead);
                line=state.getNeuralNetInput();
                lineRead=String.valueOf(line[0]);
                for(int j=1;j<line.length;j++){
                    lineRead=lineRead+SPLIT+String.valueOf(line[j]);
                }
                if(index==NO_OF_LOSSES){//exits loop once all data is
obtained
                    break;
                }
                data[index]=lineRead;
                index++;
                if(index%50000==0){
                    System.out.println(index);
                }
            }buffRead.close();read.close();
            System.out.println("Done with win file");
            //loads loss data
            index=0;
            FileReader readLose = new FileReader(LOSE_FILE_NAME);
            BufferedReader buffReadLose = new BufferedReader(readLose);
            while((lineRead=buffReadLose.readLine())!=null){
                state =new ChessBoardState(lineRead);
                line = state.getNeuralNetInput();
                lineRead= String.valueOf(line[0]);
                for(int j=1;j<line.length;j++){
                    lineRead=lineRead+SPLIT+line[j];
                }
                if(Math.random()>=0.5){//creates random win,loss
pairings
                    data[index]="1"+SPLIT+"0"+SPLIT+data[index]+SPLIT+li
neRead;
                }else{
                    data[index]="0"+SPLIT+"1"+SPLIT+lineRead+SPLIT+data[
index];
                }
                index++;
```

```
                if(index%50000==0){
                    System.out.println(index);
                }
            }
            buffReadLose.close();readLose.close();
            System.out.println("Done with lose file");
            //writes data to file
            shuffleArray(data);
            System.out.println("Shuffled");
            FileWriter write = new FileWriter(TRAIN_FILE_NAME,false);
            BufferedWriter buffWrite = new BufferedWriter(write);
            buffWrite.write(data[0]);
            for(int i=1;i<NO_OF_TRAINING_BOARDS;i++){
                buffWrite.newLine();
                buffWrite.write(data[i]);
            }buffWrite.flush();write.flush();
            buffWrite.close();write.close();
            FileWriter w = new FileWriter(TEST_FILE_NAME,false);
            BufferedWriter buff = new BufferedWriter(w);
            buff.write(data[NO_OF_TRAINING_BOARDS]);
            for(int i=1;i<NO_OF_TEST_BOARDS;i++){
                buff.newLine();
                buff.write(data[NO_OF_TRAINING_BOARDS+i]);
            }buff.flush();w.flush();
            buff.close();w.close();
            System.out.println("Data created");
        }catch(IOException e){
            System.out.println("Error "+e);
        }
    }
    /**
     * Raw fen strings from each file are converted into data in a
format that can be input to the neural network. This method was used
     * and this method labels board states in a game of chess that was
eventually won as positive and vice versa. This approach produced a high
accuracy
     * although it was limited by the hardware used to train it. It was
also limited by the fact that the dataset of boardstates is not as
strong
     * as previously thought.
     */
    private static void formatData(){
        try{
            int[]line;//stores the input to the neural network as a
double array
            int i=0;//stores the current record number
            String data[]= new String[TOTAL_AMOUNT_OF_DATA];//stores the
data as an array
            ChessBoardState state;//stores the chessboard state
            String lineRead;//stores the lineRead
            System.out.println("Array created");
            FileReader loseRead = new FileReader(LOSE_FILE_NAME);
            BufferedReader buffReadLose=new BufferedReader(loseRead);
```

```
            while((lineRead=buffReadLose.readLine())!=null){
                state = new ChessBoardState(lineRead);//creates a new
chess board using the FEN string
                line=state.getNeuralNetInput();
                lineRead="0"+SPLIT+"1";//used to indicate the desired
network output
                for(int element=0;element<WIDTH_OF_DATA-2;element++){
                    lineRead=lineRead+SPLIT+String.valueOf(line[element]
);
                }
                data[i]=lineRead;
                i++;
                if(i%50000==0){
                    System.out.println(i);
                }
            }
            buffReadLose.close();loseRead.close();
            System.out.println("done with lose");
            FileReader winRead = new FileReader(WIN_FILE_NAME);
            BufferedReader buffReadWin = new BufferedReader(winRead);
            while((lineRead=buffReadWin.readLine())!=null){
                state = new ChessBoardState(lineRead);//creates a new
chess board using the FEN string
                line=state.getNeuralNetInput();
                lineRead="1"+SPLIT+"0";//used to indicate the desired
network output
                for(int element=0;element<WIDTH_OF_DATA-2;element++){
                    lineRead=lineRead+SPLIT+String.valueOf(line[element]
);
                }
                data[i]=lineRead;
                i++;
                if(i==TOTAL_AMOUNT_OF_DATA){
                    break;
                }
                if(i%50000==0){
                    System.out.println(i);
                }
            }
            System.out.println("Done with win");
            buffReadWin.close();winRead.close();
          /*  FileReader drawRead = new FileReader(DRAW_FILE_NAME);
            BufferedReader buffReadDraw=new BufferedReader(drawRead);
            while((lineRead=buffReadDraw.readLine())!=null){
                state = new ChessBoardState(lineRead);//creates a new
chess board using the FEN string
                line=state.getNeuralNetInput();
                lineRead="0"+SPLIT+"1"+SPLIT+"0";//used to indicate the
desired network output
                for(int element=0;element<WIDTH_OF_DATA-3;element++){
                    lineRead=lineRead+SPLIT+String.valueOf(line[element]
);
                }
```

```
                data[i]=lineRead;
                i++;
                if(i%50000==0){
                    System.out.println(i);
                }
            }
            buffReadDraw.close();drawRead.close();
            System.out.println("Done with draw");*/
            //writes data to file
            shuffleArray(data);//shuffles array randomly
            //writes training and test data to file
            FileWriter writeTrain = new
FileWriter(TRAIN_FILE_NAME,false);
            BufferedWriter buffWriteTrain = new
BufferedWriter(writeTrain);
            buffWriteTrain.write(data[0]);//writes the first line
            for(i=1;i<NO_OF_TRAINING_BOARDS;i++){
                buffWriteTrain.newLine();
                buffWriteTrain.write(data[i]);
            }
            buffWriteTrain.flush();writeTrain.flush();
            buffWriteTrain.close();writeTrain.close();
            //writes test data to file
            FileWriter writeTest = new FileWriter(TEST_FILE_NAME,false);
            BufferedWriter buffWriteTest = new
BufferedWriter(writeTest);
            buffWriteTest.write(data[NO_OF_TRAINING_BOARDS]);
            for(i=NO_OF_TRAINING_BOARDS+1;i<NO_OF_TEST_BOARDS+NO_OF_TRAI
NING_BOARDS;i++){
                buffWriteTest.newLine();
                buffWriteTest.write(data[i]);
            }
            buffWriteTest.flush();writeTest.flush();
            buffWriteTest.close();writeTest.close();
        }catch(IOException e){
            System.out.println("Error "+e);
        }
        System.out.println("Done with formatting");
    }
    /**
     * Randomly shuffles inputted array
     * @param arr The array to shuffle
     */
    private static void shuffleArray(String[]arr){
        Random rnd = new Random();
        String buff;int el;
        for(int i=0;i<arr.length;i++){
            el=rnd.nextInt(arr.length);
            buff=arr[el];
            arr[el]=arr[i];
            arr[i]=buff;
        }
    }
```

```
}
```

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package philipm.cs5.software.development.neural.network.train;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

/**
 * This class enables the user to make use of stockfish.
 * @author mortimer
 */
public class StockFish {
    private BufferedReader input;//used to input to the engine
    private BufferedWriter output;//used to read the output of the
engine
    private Process process;//stores the process (i.e. the execution of
the stockfish engine)
    private static final String
STOCK_FISH_LOCATION="stockfish_20090216_x64.exe";//stores the name of
the stockfish location
    /**
     * Creates a new stock fish object that can run Stockfish 12
     * @throws java.io.IOException An exception may occur when
attempting to access Stockfish
     */
    public StockFish()throws IOException{
        try{
            process=Runtime.getRuntime().exec(STOCK_FISH_LOCATION);
            input = new BufferedReader(new
InputStreamReader(process.getInputStream()));
            output = new BufferedWriter(new
OutputStreamWriter(process.getOutputStream()));
        }catch(IOException e){
            throw e;//throws any erros that may occur
        }
    }
    /**
     * Gets the value of a given fen board state
     * @param currentFen The board state
     * @param isBlackTurn Is it black's turn to play
```

```
     * @return The value in centipawns
     * @throws IOException An exception when using stockfish
     */
    public int getCurrentBoardValue(String currentFen,boolean
isBlackTurn)throws IOException, Exception{
        try{
            //sets board state to appropriate FEN
            executeTask("position fen "+currentFen);
            //best move is calculated
            executeTask("go movetime 50");//this gets the best move with
a search time of 1000 milliseconds
            String prevLine="";
            String lineRead;
            while((lineRead=input.readLine())!=null){
                if(lineRead.contains("score cp ")){
                    prevLine=lineRead;
                }else if(lineRead.contains("bestmove")){
                    break;
                }
            }
            if(prevLine.equals("")){
                throw new Exception("score format exception");
            }
            prevLine=prevLine.split("score cp ")[1];
            String val="";
            for(int c=0;c<prevLine.length();c++){
                if(Character.isDigit(prevLine.charAt(c)) ||
prevLine.charAt(c)=='-'){
                    val=val+String.valueOf(prevLine.charAt(c));
                }else{
                    break;
                }
            }
            int multiplier=1;
            if(isBlackTurn){
                multiplier=-1;
            }
            return multiplier*Integer.parseInt(val);
        }catch(IOException e){
            throw e;
        }
    }
    /**
     * Gets the Stockfish 12's move from the given board state.
     * @param currentFen The FEN of the current board state
     * @return The board state after the computer has made its move
     * @throws IOException An exception may occur when attempting to
access Stockfish
     */
    public String getStockFishMove(String currentFen) throws
IOException{
        try{
            //sets board state to appropriate FEN
```

```
            executeTask("position fen "+currentFen);
            //best move is calculated
            executeTask("go movetime 1000");//this gets the best move
with a search time of 1000 milliseconds
            String bestMove= readOutput("bestmove ").split("
")[1];//gets the computer move
            executeTask("position fen "+currentFen+" moves
"+bestMove);//updates the board
            executeTask("d");//gets the computer to draw the current
board and output the corresponding FEN
            String lineWithFen=readOutput("Fen: ");//gets the line
containing the current Fen String
            String fen="";//stores the Fen to return
            for(int c=5;c<lineWithFen.length();c++){//loops through
every character in the FEN and appends it to the String storing the FEN
board
                fen = fen+String.valueOf(lineWithFen.charAt(c));
            }
            return fen;
        }catch(IOException e){
            throw e;
        }
    }
    /**
     * Executes the desired instruction
     * @param task The UCI command to execute
     * @throws java.io.IOException An exception may occur when using
Stockfish
     */
    private void executeTask(String task) throws IOException{
        try{
            output.write(task+System.lineSeparator());
            output.flush();//writes the command and flushes the output
        }catch(IOException e){
            throw e;
        }
    }
    /**
     * Loops through all new lines and checks to see if they begin with
the expectedValue. If they do, the line is returned
     * @param expectedValue The expected start of the line
     * @return The line read
     * @throws IOException An IOException may occur when using Stockfish
     */
    private String readOutput(String expectedValue)throws IOException{
        try{
            String lineRead;//stores the line read
            while((lineRead=input.readLine())!=null){
                if(lineRead.startsWith(expectedValue)){
                    return lineRead;
                }
            }
        }catch(IOException e){
```

```
            throw e;
        }
        return null;
    }
    /**
     * Exits the application and closes the Stockfish application. Once
this process is called, no further calls can be made to the current
StockFish instance
     * @throws IOException An exception may occur when attempting to
access Stockfish
     */
    public void exit()throws IOException{
        try{
            executeTask("quit");//attempts to quit the program
        }finally{
            process.destroy();//closes the task
            input.close();
            output.close();
        }
    }
}
```