### COP4331 Object Oriented Design and Programming

# 3 Dimensional Tic-Tac-Toe Game with Chat Feature

Group Number: 23

Professor: Ionut Cardei

Team Members:

Bryan Barreto, Abir Faisal, Jamahl Farrington, Ruth Jimenez

Deliverable 3
Due: 12/07

## Table of Contents

Application Requirements	Pages 3-7
CRC Cards	Pages 8-12
Sequence Diagrams (Scenarios)	Page 13
UML Diagrams General Application UML MVC Detail (Model/View) MVC Detail (Controller) MVC Detail (Chat/Util/Server)	Pages 14-17 Page 14 Page 15 Page 16 Page 17
Sequence Diagrams (Program)	Pages 18-19
State Diagrams	Page 19
Java Code	Pages 21-85

### **Application Requirements**

@Authors Abir Fasial, Byran Barreto

Scenario: User Application, the software is used by end users as a desktop GUI application.

Use Case 1 - Application - Single Player

Description: Application is launched, and a user plays against a computer algorithm.

Actors: User

Preconditions:

- a. User is present
- b. Graphical User Interface (GUI) is available
- c. Application is running

### Flow:

- 1. Application presents options to the User
- 2. User selects the 'Single Player' option
- 3. Application assigns a SinglePlayerController Class to the GUI
- 4. User plays against an algorithm implemented by the Class
- 5. Application checks if there is a winner/tie after each move.

### Terminations:

- a. User Wins
  - 1. Application informs User that it has won the game
  - 2. Application asks User if they would like to play again
- b. Computer Wins
  - 1. Application informs User that it has lost the game
  - 2. Application asks User if they would like to play again
- c. Game is tied
  - 1. Application informs User that the game is tied
  - 2. Application asks User if they would like to play again

@Authors Abir Fasial, Byran Barreto
Use Case 2 - Application - Single Client Host

Description: Application is launched and used as a host for a single client to join as player 2.

### Actors:

- a. User1
- b. User2

### Preconditions:

- a. User1 is present
- b. User2 is present
- c. User1 is connected to a network
- d. User2 is connected to a network
- e. Graphical User Interface (GUI) is available
- f. Application is running

### Flow:

- 1. Application presents options to the User1
- 2. User1 selects the 'Host Game' option
- 3. Application assigns a SingleHostController Class to the GUI
- 4. Application displays connection information and waits for another user to join it
- 5. User2's instance of Application presents options to the User2
- 6. User2 selects the 'Multiplayer' option
- 7. User2 enters connection information related to User1's instance.
- 8. Application assigns a MultiPlayerClientController Class to the GUI using the connection information.
- 9. User1 plays against User2 over a computer network
- 10. Host Application (User1) checks if there is a winner/tie after each move.

### Terminations:

- d. User1 Wins
  - 1. Application informs User1 that it has won the game
  - 2. Application informs User2 that it has lost the game
  - 3. Application asks User1 and User2 if they would like to play again
- e. User2 Wins
  - 1. Application informs User2 that it has won the game
  - 2. Application informs User1 that it has lost the game
  - 3. Application asks User1 and User2 if they would like to play again
- f. Game is tied
  - 1. Application informs User1 and User2 that the game is tied
  - 2. Application asks User1 and User2 if they would like to play again

@Authors Abir Fasial, Byran Barreto
Use Case 3 - Application - Multiplayer Client

Description: Application is launched and used as a client and joins a host.

Actors: User, Server

### Preconditions:

- a. User is present
- b. Server is present
- c. Graphical User Interface (GUI) is available
- d. Application is running

### Flow:

- 1. Application presents options to the User
- 2. User selects the 'Join Server' option
- 3. The application allows the user to connect to the default server or a server or host of their choice.
- 4. Application connects to the server as a client.
- 5. Server matches User with another client (User2).
  - a. If there are no other players on the server, the client will play against an algorithm implemented by the server.
  - b. If there is an odd number of of players, the client will play against an algorithm implemented by the server.
- 6. Server tracks each set of players and the state of each game.
- 7. Server ingests actions from each client.
- 8. Server interprets the state of each game and informs clients of wins, losses, and ties.

#### Terminations:

- a. User disconnects
  - i. The server will attempt to match the User with a different User.
- b. Server(host) disconnects
  - i. The clients will attempt to recover the connection.
- c. Server runtime error
  - i. The server will restart.

@Authors Abir Fasial, Byran Barreto

Scenario: Server. The software is used in a server environment to host players as clients.

Use Case 4 - Server - Multiplayer Host

Description: Program is run as a game server and used as a host.

Actors: Admin

### Preconditions:

a. Command line is available

### Flow:

- 1. Program is launched in server mode using the --server command line option.
- 2. Program presents itself as a server on the network
- 3. A number of players connect to the server
- 4. Server matches players
  - a. If a player cannot be matched it will play against an algorithm implemented by the program.
- 5. Server tracks each set of players and the state of each game
- 6. Server ingests actions from each client and informs corresponding clients of said actions clients.
- 7. Server interprets the state of each game and informs clients of wins, losses, and ties.

## CRC Cards

### @Authors Abir Fasial

class, responsibilities, collaborators

Main	
load settings launch App or Server	App Server SettingsManager

package app/

Арр	
setup the main window initialize the MVC	Model View Controller

abstract class Model	
manage data for a view notify view of changes	View

abstract class View	
user input notify controller of input	Controller

abstract class Controller	
update the model	Model

## package startscreen/

StartScreenModel extends Model	
define data keys store data values store server info	View

StartScreenView extends View	
show the start screen user selects game type user inputs server if needed	StartScreenController

StartScreenController extends Controller	
tells App what type of game to launch handle user interaction	Арр

# package game/

GameModel	
define data keys store data values store gamestats	GameView
GameView	
define UI elements	GameControllers
enum GameType	
Provide strategy pattern Single,multi, or host game	Арр
SinglePlayerGameController	
run game internally handle game logic handle game input	GameView GameModel
MultiPlayerClientController	
handle game input send input to server receive from server	GameView GameModel
MultiPlayerHostController	
handle game input provide server for other player handle game logic	GameView GameModel

# package util/

Solver	
provide a solver for the game	GameControllers Server

SettingsManager	
Load settings from file Save settings to file	Everything

## package server/

Server	
provide a way for clients to connect and communicate	Clients

# package chat/

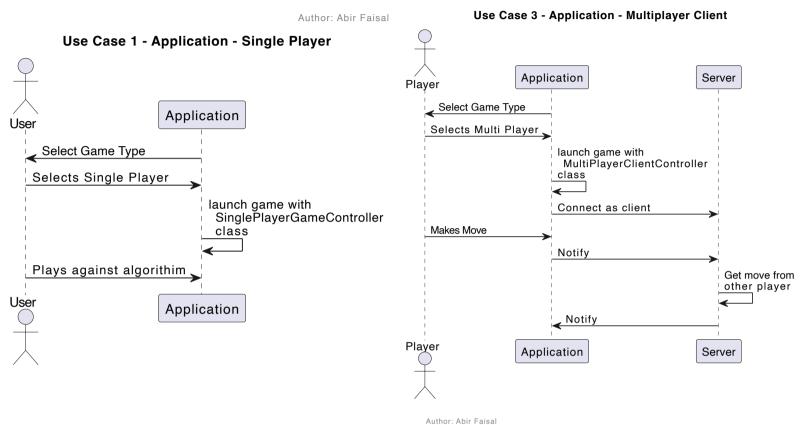
ChatModel	
hold chat messages	
<u> </u>	
ChatView	
Display chat messages input new messages	
ChatController	
Handle the Chat window	ChatBotController ChatClientController
ChatBotController	
respond to chat messages in single player mode	
ChatClientController	
send receive messages from server	

### Sequence Diagrams for scenarios

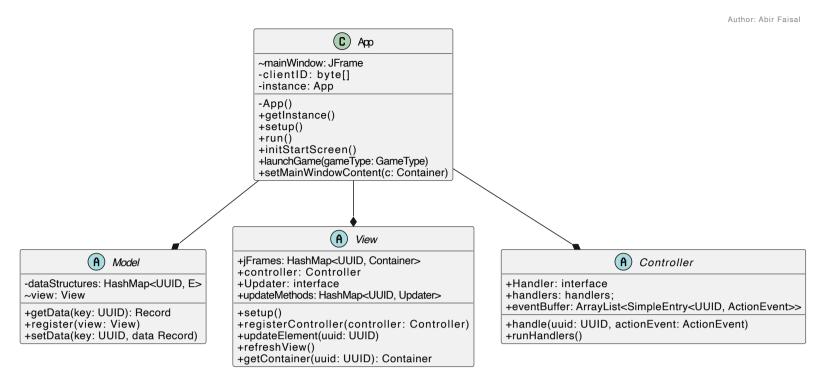
Use Case 2 - Application - Single Client Host

Description: Sequence diagrams for our use case scenarios

Author: Abir Faisal



Author: Abir Faisal Use Case 4 - Server - Multiplayer Host ApplicationA ApplicationB User2 Present launch options Selects Host Game Option Server Clients Admin SingleHostController class Launches with Present launch options "--server" option Enters connection info connects Selects Multiplayer Option match players MultiPlayerClientController class makes move connects as client Makes move process move notifies Notify Makes move notifies Admin Server Clients ApplicationA **ApplicationB** 

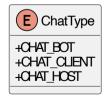


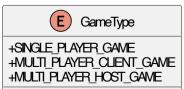
Description: The App class is implimented as a singleton pattern

It declares a Model, View, and Controller.

When the application is launched it initializes an empty window and puts whatever View type the programmer specifies into the mainWindow JFrame

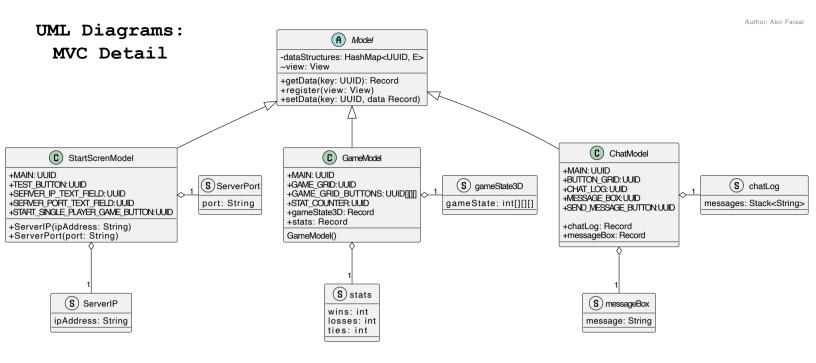
### More detail on next page

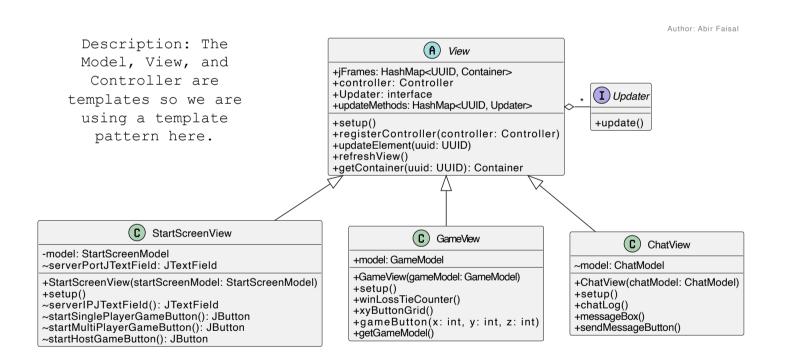




Description: These
Enums are used by
the launch
controller to tell
the App what type of
Game and Chat to
launch.

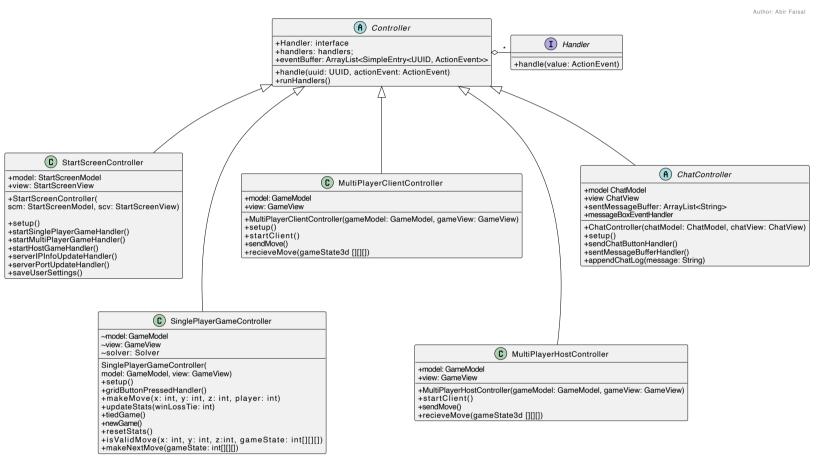
This is a strategy pattern.





# UML Diagrams: MVC Detail

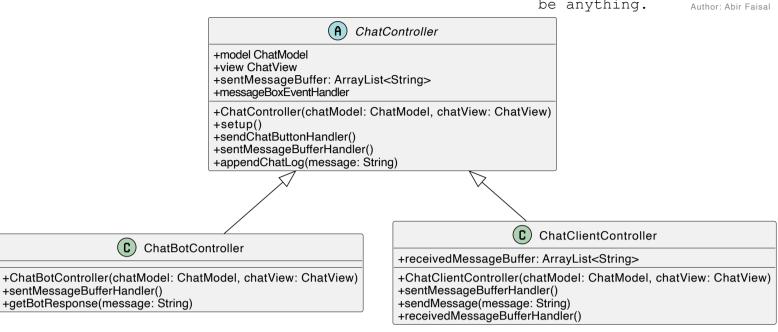
Description: The Model, View, and Controller details continued



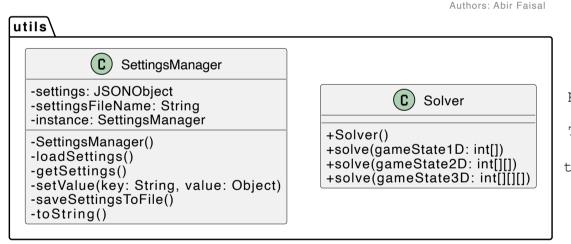
# UML Diagrams: MVC Detail Chat Feature

Description: The chat feature was not a part of our original design, but we thought it would show how our MVC architecture is extensible, you can just extend the MVC classes and make your own thing. It doesn't have to be a game it can be anything.

Author: Abir Faisal

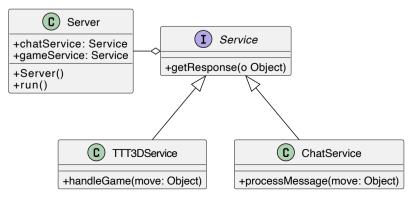


### UML Diagrams: Other



Description: These are utilities that various parts of the program can use as needed.

The solver can check the game for a winner, and the SettingsManager loads and saves program settings from file.



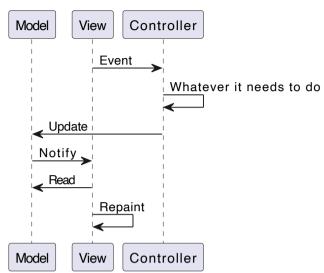
Description: The server is suppose to provide services for clients.

We didn't get to it but basically the idea was that the server could provide any service to any client as long as there exists a service handler.

### Sequence Diagrams for program

Author: Abir Faisal

### **Model View Controller**

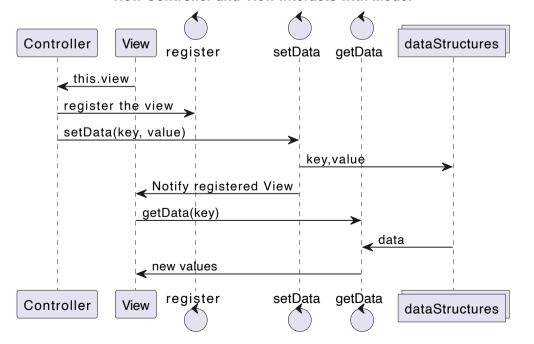


Description: The MVC components interact with each other using UUIDs that are defined in models that extend the abstract Model. (Template pattern)

The templates contain everything needed forthe MVC to work. You just have to define the UI components in your View subclass, event handlers in your Controller subclass, and UUID constants in the Model as well as data strctures in the form of record classes.

Author: Abir Faisal

### **How Controller and View interacts with Model**

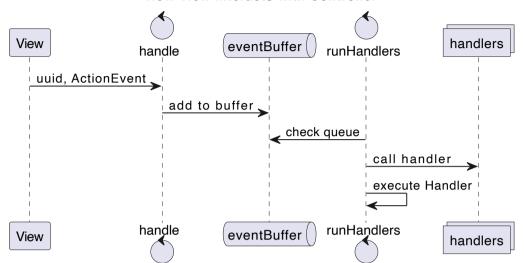


Description: A View is registered to a Model.
When the model is updated by the Controller the view is notified that the model has changed. From there the view will update itself from the model.

# Sequence Diagrams for program continued

Author: Abir Faisal

#### **How View interacts with Controller**

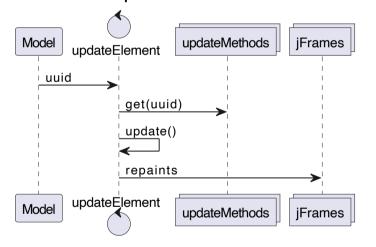


Description: When an event occurs in the view the Controller checks the handlers HashMap to see if it contains a corresponding Handler.

If so then the handler is executed.

Author: Abir Faisal

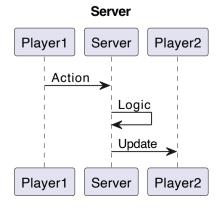
### How View updates UI elements from Model



Description: Each Swing component has an by declaring an new Updater with the Updater interface and putting it into the updateMethods hashmap.

When the model notifies the view the View calls the corresponding update method executes

Author: Abir Fasial

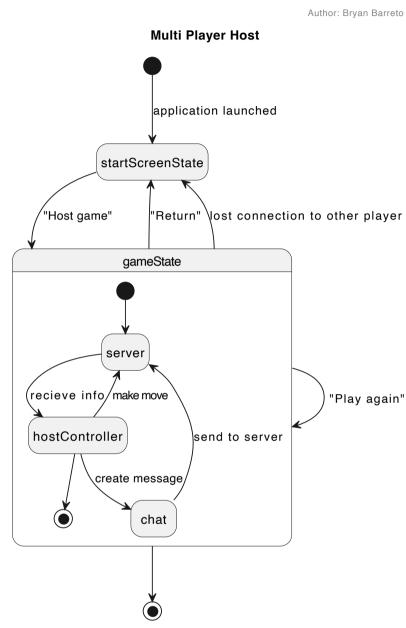


Description: Server
We didn't get to this
but basically it's a
server that would have
players connect to it
and it would mediate
between them.

### State Diagrams:

Author: Bryan Barreto Single Player application launched startScreenState "Single Player" )"return" gameState "play again" Author: Bryan Barreto Multi Player Join application launched startScreenState 'Join game" "Return" Vost connection to host gameState server (recieve info/make move "Play again" clientController send to server create message, chat

Description: General Application State Diagrams



```
File - Main.java
 1 package edu.fau.eng.cop4331.ttt3d;
 3 import edu.fau.eng.cop4331.ttt3d.app.App;
 4 import edu.fau.eng.cop4331.ttt3d.server.Server;
 5 import edu.fau.eng.cop4331.ttt3d.util.SettingsManager;
 7 import java.io.IOException;
 8 import java.util.HashMap;
 9 import java.util.Map;
10
11 public class Main {
12
       public static void main(String[] args) throws IOException {
13
           Map<String, Integer> argmap = new HashMap<>();
14
15
           //interpret command line arguments
16
            //For example --f0 1234 --f1 5678
            for (int i = 0; i < args.length; i++) {
17
18
                String argument = args[i];
19
20
                if (argument.startsWith("--")) {
                    String key = argument.substring(2); //remove -- from key
21
22
                    String value = args[i+1]; //get value of key
23
                    argmap.put(key, Integer.parseInt(value)); //put key and value into map for use
                    System.out.println(key + "=" + value);//TODO remove when no longer needed.
24
25
                }
           }
26
27
28
29
           //load settings
30
           SettingsManager settingsManager = SettingsManager.getInstance();
            settingsManager.loadSettings();
31
32
            //if --server then launch game server instead of user application
33
           if (argmap.get("--server") != null) {
34
35
                System.out.println("Running Server");
36
                //port for server
37
38
                Server server = new Server();
39
                server.run();
40
41
                //TODO ip and port for load balance and failover
42
43
                //TODO Server server = new Server(port, secondaryServerIP, );
44
                //server.run();
45
           } else {
46
                System.out.println("Launch Start Screen");
47
                App instance = App.getInstance();
48
                instance.setup();
49
                instance.run();
50
           }
       }
51
52 }
```

```
File - App.java
 1 package edu.fau.eng.cop4331.ttt3d.app;
 3 import edu.fau.eng.cop4331.ttt3d.app.chat.*;
 4 import edu.fau.eng.cop4331.ttt3d.app.game.*;
 5 import edu.fau.eng.cop4331.ttt3d.app.startscreen.StartScreenController;
 6 import edu.fau.eng.cop4331.ttt3d.app.startscreen.StartScreenModel;
 7 import edu.fau.eng.cop4331.ttt3d.app.startscreen.StartScreenView;
 8 import edu.fau.eng.cop4331.ttt3d.util.SettingsManager;
 9 import org.json.JSONArray;
10
11 import javax.swing.*;
12 import java.awt.*;
13 import java.util.Random;
14
15
16 public class App {
17
       JFrame mainWindow;
18
       private byte[] clientID; //128 bit client id
19
20
       //Singleton Pattern
21
       private static App instance;
22
       private App() {
            this.mainWindow = new JFrame("TTT3D");
23
24
           this.clientID = getClientID();
25
26
       public static synchronized App getInstance() {
           if (instance == null) instance = new App();
27
28
           return instance;
29
       }
30
31
32
        * Set up the components of the main window and/or application
33
        * @author Abir Faisal
34
       public void setup() {
35
36
           initStartScreen();
           this.mainWindow.setSize(800,600);//400 width and 500 height
37
38
           this.mainWindow.setDefaultCloseOperation(JFrame.EXIT ON CLOSE);
39
            ///TODO this.mainWindow.setLayout();
40
           this.mainWindow.setVisible(true);
41
       }
42
43
        /**
44
        * run the application
45
46
        * <u>@author</u> Abir Faisal
47
48
       public void run() {
49
            //TODO this might be useless...
50
51
52
53
        * Setup the initial MVC you want to show the user
54
55
        * @author Abir Faisal
56
        */
       public void initStartScreen(){
57
            StartScreenModel startScreenModel = new StartScreenModel();
58
59
           StartScreenView startScreenView = new StartScreenView(startScreenModel);
           StartScreenController startScreenController = new StartScreenController(startScreenMov
60
            setMainWindowContent(startScreenView.getContainer(startScreenModel.MAIN));
61
62
       }
63
        /**
64
           Launch the application specified by the initial screen
65
66
```

```
File - App.java
 67
          * <u>@author</u> Abir Faisal
 68
          * @param gameType the type of game you want to launch
 69
 70
         public void launchGame(GameType gameType) {
             GameModel gameModel = new GameModel();
 71
 72
             GameView gameView = new GameView(gameModel);
 73
 74
             ChatModel chatModel = new ChatModel();
 75
             ChatView chatView = new ChatView(chatModel);
 76
 77
             switch (gameType) {
 78
                 case SINGLE PLAYER GAME -> {
 79
                     SinglePlayerGameController gameController = new SinglePlayerGameController(gameController)
 80
                     ChatController chatController = new ChatBotController(chatModel, chatView);
 81
 82
                     //show game and chat side by side
                     JSplitPane jSplitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
 83
 84
                     jSplitPane.add(gameView.getContainer(gameModel.MAIN));
 85
                     iSplitPane.add(chatView.getContainer(chatModel.MAIN));
 86
                     setMainWindowContent(jSplitPane);
 87
                 }
 88
 89
                 case MULTI PLAYER CLIENT GAME -> {
 90
                     MultiPlayerClientController gameController = new MultiPlayerClientController
 91
                     ChatController chatController = new ChatClientController(chatModel, chatView
 92
                     //TODO ChatClientController
 93
 94
 95
                     //show game and chat side by side
 96
                     JSplitPane | SplitPane = new JSplitPane(JSplitPane.HORIZONTAL SPLIT);
                     jSplitPane.add(gameView.getContainer(gameModel.MAIN));
 97
 98
                     jSplitPane.add(chatView.getContainer(chatModel.MAIN));
 99
100
                     setMainWindowContent(jSplitPane);
101
                 case MULTI PLAYER HOST GAME -> {}
102
             }
103
104
             System.out.println("Launching Game " + gameType);
105
         }
106
107
108
109
         * Generate a client ID or try to load from settings
110
111
          * <u>@author</u> Abir Faisal
112
          * <u>Qreturn</u> 128bit Client ID as byte[16], 16 * 8bit = 128bits
113
         public byte[] getClientID() {
114
115
             if (this.clientID == null) {
                 this.clientID = new byte[16];
116
117
                 SettingsManager sm = SettingsManager.getInstance();
118
119
                 //if no clientID in settings.json then generate and save
120
                 //else load from configureation
                 if (sm.getSettings().opt("clientID") == null) {
121
122
                     Random r = new Random();
123
                     r.nextBytes(this.clientID);
124
                     //save to settings
                     sm.setValue("clientID", this.clientID);
125
126
                 } else {
127
                     //load from settings
128
                     JSONArray clientIDJSONArray = sm.getSettings().getJSONArray("clientID");
129
130
                     for (int i = 0; i < clientIDJSONArray.length(); i++) {</pre>
                         this.clientID[i] = (byte) clientIDJSONArray.getInt(i);
131
132
                     }
```

```
}
133
134
135
            return clientID;
        }
136
137
138
        /**
         * set the content of the main window, replace existing content
139
140
         * <u>@author</u> Abir Faisal
141
         * Oparam c a JPanel that contains the contents you want to display
142
143
         */
144
        public void setMainWindowContent(Container c) {
            this.mainWindow.getContentPane().removeAll();
145
            this.mainWindow.setContentPane(c);
146
147
            this.mainWindow.revalidate();
148
        }
149
150
         * add the content to the main window
151
152
153
         * @author Abir Faisal
154
         * Oparam c a JPanel that contains the contents you want to display
155
        public void addMainWindowContent(Container c) {
156
            this.mainWindow.add(c);
157
            this.mainWindow.revalidate();
158
159
        }
160
161 }
162
163
```

File - App.java

```
1 package edu.fau.eng.cop4331.ttt3d.app;
 3 import java.awt.*;
 4 import java.util.HashMap;
 5 import java.util.UUID;
 6 import java.util.function.BiConsumer;
8 public abstract class View {
9
10
11
       //Objects of the view
12
       public HashMap<UUID, Container> jFrames = new HashMap<>();
       public Controller controller;
13
14
15
16
       //methods that are called when update is called on a UUID mapped to jFrames
       public HashMap<UUID, Updater> updateMethods = new HashMap<>();
17
18
19
       //TODO remove, it seems like its not used
20 //
         public Model model;
         public View(Model model) {
21 //
22 //
             this.model = model;
23 //
             this.model.register(this);
24 //
         }
25
26
       /**
        * Used to setup the view, setup the main view and add elements to it
27
28
        * This should be called in the constructor
29
30
        * @author Abir Faisal
31
32
       public abstract void setup();
33
34
35
       //register a controller for the view
36
37
       /**
38
        * Registers a controller with the view so that the view
        * is aware of where it needs to send actions and events.
39
40
        * The view will call it's handle(UUID) method when soemthing happens.
41
42
        * @author Abir Faisal
        * Oparam controller A subclass that extends the abstract Controller
43
44
45
       public void registerController(Controller controller){
46
           this.controller = controller;
47
48
49
50
51
        * Updates an element of the view given its corresponding UUID
52
53
        * <u>@author</u> Abir Faisal
54
        * @param uuid UUID as defined in the model of the view
55
56
       public void updateElement(UUID uuid) {
           if (this.updateMethods.get(uuid) != null)
57
58
               this.updateMethods.get(uuid).update();
59
       }
60
61
62
        * Refresh/Update the whole view.
63
64
        * @author Abir Faisal
65
       public void refreshView(){
66
```

File - View.java

```
File - View.java
             BiConsumer<? <pre>super UUID, ? super Updater> biConsumer = (uuid, updater) -> updater.up
 67
             updateMethods.forEach(biConsumer);
 68
 69
         }
 70
 71
         /**
         * Get a component of the view
 72
 73
 74
          * <u>@author</u> Abir Faisal
          * @param uuid UUID as defined in the model of the view
 75
 76
         public Container getContainer(UUID uuid){
 77
             return this.jFrames.get(uuid);
 78
 79
         }
 80
 81 }
 82
```

```
1 package edu.fau.eng.cop4331.ttt3d.app;
 3 import java.util.HashMap;
 4 import java.util.UUID;
 6 public abstract class Model <E> {
8
       // Contains data structures that will be
9
       // updated by the controller or read by the view
       HashMap<UUID, E> dataStructures = new HashMap<>();
10
11
12
       //Just a refrence to the view that should be notified
       //when data is updated in of this model
13
14
       View view;
15
16
       /**
17
        * The view will request a dataStructure from the Model
18
19
        * @author Abir Faisal
20
        * @param key UUID as defined in a subclass of this Model
21
22
       public Record getData(UUID key) {
23
           return (Record) dataStructures.get(key);
24
       }
25
26
        * Register a view with the model so that setData()
27
28
        * can call its notify method after updating a value
29
30
        * @author Abir Faisal
        * Oparam view the view that should be notified of changes to this model
31
32
       public void register(View view) {
33
34
           this.view = view;
35
36
37
38
39
        * Allows the controller to set/update a dataStructure
40
        * and the model to notify the view
41
        * @author Abir Faisal
42
        * Oparam key UUID as defined in a subclass of this Model
43
        * Oparam data record object as defined a subclass of this Model
44
45
        */
46
       public synchronized void setData(UUID key, Record data) {
           if (dataStructures.containsKey(key)){
47
48
               //replace the object
               dataStructures.replace(key, (E) data);
49
50
               //notify the view that data has changed
51
               this.view.updateElement(key);
52
           } else {
53
               //add the object
54
               this.dataStructures.put(key, (E) data);
55
56
               //notify the view that data has changed
               this.view.updateElement(key);
57
58
           }
59
       }
60
61 }
```

File - Model.java

62

```
File-Handler.java
1 package edu.fau.eng.cop4331.ttt3d.app;
2
3 import java.awt.event.ActionEvent;
4
5 public interface Handler {
6    void handle(ActionEvent value);
7 }
8
```

```
File-Updater.java
1 package edu.fau.eng.cop4331.ttt3d.app;
2
3 /**
4 * Updater interface.
5 * the method update() is called when something in the view needs to be updated.
6 *
7 * @author Abir Faisal
8 */
9 public interface Updater {
10 void update();
11 }
12
```

```
File - Controller.java
 1 package edu.fau.eng.cop4331.ttt3d.app;
 3 import java.awt.event.ActionEvent;
 4 import java.util.AbstractMap.SimpleEntry;
 5 import java.util.ArrayList;
 6 import java.util.HashMap;
 7 import java.util.UUID;
 9 import static java.lang.Thread.sleep;
10
11 public abstract class Controller {
12
13
       //Contains a set of UUID and handlers implimenting the Handler interface
14
       public HashMap<UUID, Handler> handlers = new HashMap<>();
15
16
17
        * When the user interacts with the View,
         * the View will notify the Controller that a (UUID, actionEvent) has occurred,
18
19
         * then the (UUID, ActionEvent) will go into a handlerBuffer
20
        * later it will be handled by a Thread launched by runHandlers().
21
22
         * @author Abir Faisal
23
        */
24
25
26
       public ArrayList<SimpleEntry<UUID, ActionEvent>> eventBuffer = new ArrayList<>();
27
28
       /**
29
        * passes events from the UI into the event buffer.
30
         * It is handled when the runHandlers thread checks it.
31
32
         * @author Abir Faisal
33
         * @param uuid
34
         * @param actionEvent
35
       public void handle(UUID uuid, ActionEvent actionEvent) {
36
            SimpleEntry<UUID, ActionEvent> tuple = new SimpleEntry<>(uuid, actionEvent);
37
38
            eventBuffer.add(tuple);
39
       }
40
41
42
         * This will monitor the event buffer and handle any events
43
44
        * <u>@author</u> Abir Faisal
45
46
       //TODO convert to iterator pattern
       public void runHandlers() {
47
            new Thread(() -> {
48
49
                while (true) {
50
                    int i = 0;
51
                    try {
52
                        for (i = 0; i < eventBuffer.size(); i++) {</pre>
53
                             //get the UUID and ActionEvent
54
                            SimpleEntry<UUID, ActionEvent> simpleEntry = eventBuffer.get(i);
55
56
                             //Handle the event
                            UUID uuid = simpleEntry.getKey();
57
58
                            ActionEvent actionEvent = simpleEntry.getValue();
59
                            handlers.get(uuid).handle(actionEvent);
60
                             //remove from buffer
61
62
                            eventBuffer.remove(i);
63
                        }
                        sleep(50); //prevent using CPU cycles for no reason.
64
65
                    } catch (InterruptedException e) {
                        throw new RuntimeException(e);
66
```

### 

```
File - ChatView.java
 1 package edu.fau.eng.cop4331.ttt3d.app.chat;
 3 import edu.fau.eng.cop4331.ttt3d.app.Updater;
 4 import edu.fau.eng.cop4331.ttt3d.app.View;
 6 import javax.swing.*;
 7 import javax.swing.event.DocumentEvent;
 8 import javax.swing.event.DocumentListener;
 9 import javax.swing.text.DefaultCaret;
10 import java.awt.*;
11 import java.awt.event.ActionEvent;
12 import java.time.Instant;
13 import java.util.ArrayList;
14 import java.util.Stack;
15 import java.util.UUID;
16
17 public class ChatView extends View {
18
19
20
        ChatModel model;
21
22
        /**
23
        * Constructor
24
25
         * @param chatModel ChatModel
26
         */
27
        public ChatView(ChatModel chatModel){
28
            this.model = chatModel;
29
            this.model.register(this);
30
            setup();
        }
31
32
33
34
        /**
35
        * Setup the view
36
         */
        @Override
37
38
        public void setup() {
39
            JPanel mainJPanel = new JPanel();
            mainJPanel.setLayout(new BoxLayout(mainJPanel, BoxLayout.Y_AXIS));
40
41
            this.jFrames.put(this.model.MAIN, mainJPanel);
42
43
44
            this.jFrames.get(this.model.MAIN).add(chatLog());
45
            this.jFrames.get(this.model.MAIN).add(messageBox());
46
            this.jFrames.get(this.model.MAIN).add(sendMessageButton());
        }
47
48
49
50
        ////UI elements///////
51
52
         * The chat log where the user can see the send and
53
         * recieved messages
54
55
         * @author Abir Faisal
         * <u>@return</u> JScrollPane
56
57
58
        JScrollPane chatLog() {
59
            UUID uuid = this.model.CHAT_LOG;
60
            JTextArea jTextArea = new JTextArea("");
61
62
            jTextArea.setEditable(false);
63
            DefaultCaret dc = (DefaultCaret) jTextArea.getCaret();
64
            dc.setUpdatePolicy(DefaultCaret.ALWAYS_UPDATE);
65
            JScrollPane jScrollPane = new JScrollPane(jTextArea);
66
                                               Page 1 of 3
```

```
67
            jScrollPane.setPreferredSize(new Dimension(800,600));
 68
            jScrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
 69
 70
 71
            Updater updater = () -> {
 72
                //get record from model
 73
                     ChatModel.chatLog cl =
 74
                             (ChatModel.chatLog) this.model.getData(uuid);
 75
                Stack<String> messages = cl.messages();
 76
                jTextArea.append(messages.peek() + "\n\n");
 77
            };
 78
            updateMethods.put(uuid, updater);
 79
 80
            return jScrollPane;
        }
 81
 82
 83
        /**
 84
         * The message box where the user
 85
         * types in a message that they want to send.
 86
 87
         * <u>@author</u> Abir Faisal
 88
         * @return JTextArea
 89
         */
 90
        JTextArea messageBox() {
 91
            UUID uuid = this.model.MESSAGE_BOX;
 92
 93
            JTextArea jTextArea = new JTextArea();
 94
            jTextArea.setPreferredSize(new Dimension(100,50));
 95
 96
            DocumentListener dl = new DocumentListener() {
 97
 98
                @Override
 99
                public void insertUpdate(DocumentEvent e) {
100
                     controller.handle(uuid,
101
                             new ActionEvent(jTextArea, 0, jTextArea.getText())
102
                     );
103
                }
104
                @Override
105
                public void removeUpdate(DocumentEvent e) {
106
                     controller.handle(uuid,
                             new ActionEvent(jTextArea, 0, jTextArea.getText())
107
108
                     );
109
                }
110
                @Override
111
                public void changedUpdate(DocumentEvent e) {}
112
113
            jTextArea.getDocument().addDocumentListener(dl);
114
115
            Updater updater = () -> {
116
117
                ChatModel.messageBox message =
118
                         (ChatModel.messageBox) this.model.getData(this.model.MESSAGE_BOX);
119
                String strMessage = message.message();
120
121
                //if the text is different then update it, else do nothing
122
                if (!jTextArea.getText().equals(strMessage)) {
123
                     //set text without triggering event
                     jTextArea.getDocument().removeDocumentListener(dl);
124
125
                     jTextArea.setText(strMessage);
126
                     //restore the change listener
                     jTextArea.getDocument().addDocumentListener(dl);
127
                }
128
129
130
            this.updateMethods.put(uuid, updater);
131
132
```

File - ChatView.java

```
133
            return jTextArea;
134
135
136
        /**
137
         * <u>@author</u> Abir Faisal
138
         * <u>@return</u> JButton
139
         */
140
        JButton sendMessageButton(){
            UUID uuid = this.model.SEND_MESSAGE_BUTTON;
141
            JButton jButton = new JButton("Send");
142
143
144
            jButton.addActionListener(actionEvent -> {
                 this.controller.handle(uuid, actionEvent);
145
146
            });
147
148
            return jButton;
        }
149
150
151
152 }
153
```

File - ChatView.java

```
1 package edu.fau.eng.cop4331.ttt3d.app.chat;
 3 import edu.fau.eng.cop4331.ttt3d.app.Model;
5 import java.util.ArrayList;
 6 import java.util.Stack;
7 import java.util.UUID;
9 public class ChatModel extends Model {
10
11
       /**
12
13
       * The View uses these constants to get data from the Model
14
        * The Controller uses these constants to update data in the Model
15
        * Every element in a view that needs to be updated
16
17
        * needs to have a UUID refrence to it here.
18
19
        * These are non-static so the UUID will
20
        * be unique to each instance of the class
21
22
        */
23
       public UUID MAIN = UUID.randomUUID();
24
       public UUID HELLO_WORLD_JLABEL = UUID.randomUUID();
25
       public UUID CHAT_LOG = UUID.randomUUID();
26
       public UUID MESSAGE_BOX = UUID.randomUUID();
       public UUID SEND_MESSAGE_BUTTON = UUID.randomUUID();
27
28
29
       //data structures
30
31
32
       * Holds an stack array of String messages
33
        * to be displayed by the view or updated by the controller
34
35
        * @param messages Stack<String>
36
37
       public record chatLog(Stack<String> messages){}
38
39
40
       * Holds the text that the user types into the message box
41
42
        * @param message String
43
        */
44
       public record messageBox(String message){}
45
46
47 }
48
```

File - ChatModel.java

```
1 package edu.fau.eng.cop4331.ttt3d.app.chat;
 3 import edu.fau.eng.cop4331.ttt3d.app.Controller;
 4 import edu.fau.eng.cop4331.ttt3d.app.Handler;
 6 import java.util.ArrayList;
 7 import java.util.Stack;
8 import java.util.UUID;
9
10 public abstract class ChatController extends Controller {
11
12
       ChatModel model:
13
       ChatView view;
14
       ArrayList<String> sentMessageBuffer;
15
16
        * Constructor
17
18
        * @param chatModel ChatModel
19
        * @param chatView ChatView
20
       public ChatController(ChatModel chatModel, ChatView chatView) {
21
22
           this.model = chatModel;
           this.view = chatView;
23
24
           this.view.registerController(this);
25
           this.sentMessageBuffer = new ArrayList<>();
26
27
           runHandlers();
28
           setup();
       }
29
30
31
32
        * Setup the controller
33
34
       void setup() {
35
           handlers.put(this.model.SEND_MESSAGE_BUTTON, sendChatButtonHandler());
           handlers.put(this.model.MESSAGE_BOX, messageBoxEventHandler());
36
37
38
           //init the chat log datastrcture
39
           Stack<String> s = new Stack<String>();
40
           s.push("");
41
           this.model.setData(this.model.CHAT_LOG, new ChatModel.chatLog(s));
42
           this.model.setData(this.model.MESSAGE_BOX, new ChatModel.messageBox(""));
43
       }
44
45
46
       //event handlers////////
47
48
49
        * Handles what happens when the send chat button is pressed
50
        * @author Abir Faisal
51
52
        * @return
53
54
       Handler sendChatButtonHandler() {
55
           UUID messageBoxUUID = this.model.MESSAGE_BOX;
56
57
           return value -> {
58
               System.out.println("send button pressed");
59
               //get the text from the message
60
               ChatModel.messageBox mb =
                       (ChatModel.messageBox) this.model.getData(messageBoxUUID);
61
62
               String message = mb.message();
63
64
               //clear the message in the model
               this.model.setData(messageBoxUUID, new ChatModel.messageBox(""));
65
66
```

File - ChatController.java

```
67
                //append the message to the chat
 68
                appendChatLog("Player 1: " + message);
 69
 70
                 //put the message in the message buffer for the chat bot
                this.sentMessageBuffer.add(message);
 71
 72
            };
        }
 73
 74
 75
        /**
 76
         * Updates the data in the model
 77
         * when the text in the message box changes
 78
 79
         * @author Abir Faisal
 80
         * Oreturn
 81
         */
 82
        Handler messageBoxEventHandler() {
 83
            UUID uuid = this.model.MESSAGE_BOX;
 84
            return actionEvent -> {
 85
                 //update the model
 86
                this.model.setData(uuid,
                        new ChatModel.messageBox(actionEvent.getActionCommand())
 87
 88
                );
 89
            };
 90
        }
 91
        //controller logic/////////
 92
 93
 94
        /**
 95
         * Monitors the message buffer for any messages from the user
 96
         * if so then it responds to it
 97
 98
         * This can be a chat bot or client it should be implimented such that it
 99
         * reads the message buffer, handles it, then clear the message from the buffer
100
101
         * Preferable it should be in it's own thread.
102
103
         * <u>@author</u> Abir Faisal
104
105
        public abstract void sentMessageBufferHandler();
106
107
        /**
108
         * Append a message to the chatLog data structure in the model
         * This should be called when your messageBufferHandler produces response
109
110
111
         * <u>@author</u> Abir Faisal
112
         * Oparam message String message you want to append
113
         */
114
        void appendChatLog(String message) {
115
            UUID chatLogUUID = this.model.CHAT_LOG;
116
117
            //append the message to the chat
118
            ChatModel.chatLog cl =
                     (ChatModel.chatLog) this.model.getData(chatLogUUID);
119
120
            Stack<String> messages = cl.messages();
121
122
            //put the new message on the top of the stack
123
            messages.push(message);
124
125
            //update the chatlog datastructure in the model
126
            this.model.setData(chatLogUUID, new ChatModel.chatLog(messages));
127
        }
128
129 }
```

File - ChatController.java

```
1 package edu.fau.eng.cop4331.ttt3d.app.chat;
 3 import edu.fau.eng.cop4331.ttt3d.app.Handler;
5 import java.util.*;
 7 import static java.lang.Thread.sleep;
9 public class ChatBotController extends ChatController {
10
11
       /**
12
        * Constructor
13
        * @param chatModel ChatModel
14
        * @param chatView ChatView
15
        */
       public ChatBotController(ChatModel chatModel, ChatView chatView) {
16
17
           super(chatModel, chatView);
18
           sentMessageBufferHandler();
19
       }
20
21
       //controller logic/////
22
23
24
        * Monitors the message buffer for any messages from the user
25
        * if so then it responds to it
26
        * @author Abir Faisal
27
28
        */
29
       @Override
30
       public void sentMessageBufferHandler() {
           new Thread(() -> {
31
32
               while (true) {
                   for (int i = 0; i < this.sentMessageBuffer.size(); i++) {</pre>
33
34
                        //allow the bot to respond
                       getBotResponse(this.sentMessageBuffer.get(i));
35
36
                        //remove from buffer
37
                       this.sentMessageBuffer.remove(i);
38
                   }
39
                   try {
40
41
                        sleep(100); //prevent using CPU cycles for no reason.
42
                    }catch (InterruptedException e) {
43
44
45
           }).start();
46
       }
47
48
49
       /**
50
        * gets a computer generated response and puts it into the chat
51
52
        * @author Abir Faisal
53
54
       void getBotResponse(String message) {
55
           //TODO make more advanced
           String[] responses = {"Ok", "I understand", "Sure"};
56
           Random r = new Random();
57
58
           int i = r.nextInt(responses.length);
59
           appendChatLog("Bot: " + responses[i]);
60
       }
61
62
63 }
64
```

File - ChatBotController.java

```
File - ChatClientController.java
 1 package edu.fau.eng.cop4331.ttt3d.app.chat;
 3 import java.util.ArrayList;
 5 import static java.lang.Thread.sleep;
 7 public class ChatClientController extends ChatController {
 8
 9
        ArrayList<String> receivedMessageBuffer;
10
11
        /**
12
        * Constructor
13
         * @param chatModel ChatModel
14
         * @param chatView ChatView
15
        */
        public ChatClientController(ChatModel chatModel, ChatView chatView) {
16
17
            super(chatModel, chatView);
18
            sentMessageBufferHandler();
19
        }
20
        /**
21
22
         * Handles the messages in the message buffer
23
         * Sends the message to the server
        */
24
25
        @Override
26
        public void sentMessageBufferHandler() {
27
            new Thread(() -> {
28
                while (true) {
29
                    for (int i = 0; i < this.sentMessageBuffer.size(); i++) {</pre>
30
                         //send the message
                         System.out.println("Sending Message: " + sentMessageBuffer.get(i));
31
32
                         sendMessage(this.sentMessageBuffer.get(i));
33
                         //remove from buffer
34
35
                         this.sentMessageBuffer.remove(i);
36
                    }
37
38
                    try {
39
                         sleep(100); //prevent using CPU cycles for no reason.
40
                    }catch (InterruptedException e) {
41
42
43
            }).start();
        }
44
45
46
        * Handles sending the message to the server
47
48
         * @param message String
49
50
        void sendMessage(String message) {
51
52
        }
53
54
        * Handles recieved messages in the recieved message buffer
55
56
        void receivedMessageBufferHandler() {
57
            new Thread(() -> {
58
59
                while (true) {
                    for (int i = 0; i < this.receivedMessageBuffer.size(); i++) {</pre>
60
                         System.out.println("Recieved Message: " + receivedMessageBuffer.get(i));
61
62
                         //put the recieved message into the view
63
                         //remove from buffer
64
                         this.receivedMessageBuffer.remove(i);
65
                    }
66
```

```
File - ChatClientController.java
  67
                            try {
                            sleep(100); //prevent using CPU cycles for no reason.
}catch (InterruptedException e) {
  68
  69
  70
                      }
  71
                 }).start();
  72
           }
  73
  74
  75 }
  76
```

```
File-GameType.java
1 package edu.fau.eng.cop4331.ttt3d.app.game;
2
3 /**
4 * The type of game to launch
5 */
6 public enum GameType {
7    SINGLE_PLAYER_GAME,
8    MULTI_PLAYER_CLIENT_GAME,
9    MULTI_PLAYER_HOST_GAME
10 }
11
```

```
File - GameView.java
 1 package edu.fau.eng.cop4331.ttt3d.app.game;
 3 import edu.fau.eng.cop4331.ttt3d.app.Updater;
 4 import edu.fau.eng.cop4331.ttt3d.app.View;
 6 import javax.swing.*;
 7 import java.awt.*;
 8 import java.awt.event.ActionEvent;
 9 import java.time.Instant;
10 import java.util.UUID;
11
12 public class GameView extends View {
13
14
       GameModel model;
15
        /**
16
17
        * Constructor
18
19
        * @param gameModel GameModel
20
21
       public GameView(GameModel gameModel) {
22
           this.model = gameModel; //make view aware of model
           this.model.register(this); //make model aware of view
23
24
            setup(); //setup the view
       }
25
26
27
        /**
28
        * Setup the view
29
        */
30
       @Override
31
       public void setup() {
32
            JPanel mainJPanel = new JPanel();
33
           this.jFrames.put(model.MAIN, mainJPanel);
34
           mainJPanel.setLayout(new BoxLayout(mainJPanel, BoxLayout.Y_AXIS));
35
36
           this.jFrames.get(model.MAIN).add(winLossTieCounter());
37
38
           this.jFrames.get(model.MAIN).add(new JLabel("Layer1"));
39
           this.jFrames.get(model.MAIN).add(xyButtonGrid(0));
40
41
            this.jFrames.get(model.MAIN).add(new JLabel("Layer2"));
42
           this.jFrames.get(model.MAIN).add(xyButtonGrid(1));
43
44
           this.jFrames.get(model.MAIN).add(new JLabel("Layer3"));
45
           this.jFrames.get(model.MAIN).add(xyButtonGrid(2));
46
       }
47
48
       ////UI elements///////
49
50
51
       JLabel winLossTieCounter() {
52
            JLabel jLabel = new JLabel("Win: 0 Loss: 0 Tie: 0");
53
           UUID uuid = this.model.STAT_COUNTER;
54
55
           Updater updater = new Updater() {
56
                @Override
                public void update() {
57
58
                    GameModel.stats stats = (GameModel.stats) model.getData(uuid);
59
                    int win = stats.wins();
60
                    int loss = stats.losses();
61
                    int tie = stats.ties();
62
                    String statStr = "Win:" + win + " Loss:" + loss + " Tie:"+ tie;
63
                    jLabel.setText(statStr);
                }
64
65
            };
           updateMethods.put(model.STAT_COUNTER, updater);
66
```

```
File - GameView.java
 67
 68
             return jLabel;
 69
         }
 70
 71
 72
         /**
          * Grid that contains 3x3 button array
 73
 74
          * The 1 value is used to deterimine
          * which layer of the cube this grid corresponds to
 75
 76
          * Oparam layer the layer also known as the z axis
 77
 78
          * @return the grid
 79
          */
 80
         JPanel xyButtonGrid(int layer) {
 81
             JPanel grid = new JPanel();
 82
             grid.setLayout(new GridLayout(3,3));
             UUID gameGridUUID = this.model.GAME_GRID;
 83
 84
             UUID[][][] buttonUUIDS = this.model.GAME_GRID_BUTTONS;
 85
             int index = 0;
 86
 87
             //generate the buttons
 88
             for (int y = 0; y < 3; y++) {
                 for (int x = 0; x < 3; x++) {
 89
 90
                     grid.add(gameButton(x, y, layer, index));
 91
                     index +=1;
 92
                 }
             }
 93
 94
 95
             //refreshes the buttons
 96
             Updater updater = new Updater() {
 97
                 @Override
 98
                 public void update() {
 99
                     for (int z = 0; z < 3; z++) {
100
                          for (int y = 0; y < 3; y++) {
101
                              for (int x = 0; x < 3; x++) {
102
                                  UUID uuid = buttonUUIDS[x][y][z];
103
                                  GameModel.gameState3D gs3d = (GameModel.gameState3D) model.getDa
104
                                  int [][][] gs = gs3d.gameState3D();
105
                                  gs[x][y][z] = 0;
106
                                  model.setData(uuid, new GameModel.gameState3D(gs));
                              }
107
                         }
108
                     }
109
                 }
110
111
             };
112
             updateMethods.put(gameGridUUID, updater);
113
114
             return grid;
         }
115
116
117
         /**
118
         * generates the game button given the x,y,z cordinates and index
119
          * @param x cordinate
120
121
          * @param y cordinate
          * @param z layer
122
123
          * @param index counter
124
          * @return
125
         JButton gameButton(int x, int y, int z, int index) {
126
127
             UUID gameGridUUID = this.model.GAME_GRID;
128
             UUID buttonUUID = this.model.GAME_GRID_BUTTONS[x][y][z];
129
130
             JButton jButton = new JButton("-");
             jButton.setPreferredSize(new Dimension(50,50));
131
             jButton.setFont(new Font(null, Font.PLAIN, 40));
132
```

```
File - GameView.java
133
134
            //event handler will recieve this string "x,y"
135
             //optionally it can use index to identify which button was pressed
            String coordinates = x + "," + y + "," + z;
136
137
138
            //action event to be passed to the controller
139
            ActionEvent ae = new ActionEvent(jButton, index, coordinates);
140
            jButton.addActionListener(e -> this.controller.handle(gameGridUUID, ae));
141
             //if model is updated with a new gameState then do this
142
143
            int xf = x; //final
144
            int yf = y; //final
145
            int zf = z; //final
            Updater updater = new Updater() {
146
147
                 @Override
148
                 public void update() {
                       System.out.println("xyz" + xf + yf + zf);
149 //
                     //read the state from the game state record int the model datastructures
150
151
                     GameModel.gameState3D gs3d = (GameModel.gameState3D) model.getData(gameGridUl
152
                     int[][][] gs = gs3d.gameState3D();
153
                     int state = gs[xf][yf][zf];
154
155
                     //if 1 then "X" if -1 then "O" else "-"
                     if (state == 1) jButton.setText("X");
156
                     else if (state == -1) jButton.setText("0");
157
158
                     else jButton.setText("-");
                 }
159
160
            };
            updateMethods.put(buttonUUID, updater);
161
162
            return jButton;
163
        }
164
165
166
167
        /**
168
         *
169
         * <u>@return</u> GameModel
170
171
        public GameModel getGameModel() {
172
            return model;
173
        }
174 }
```

```
1 package edu.fau.eng.cop4331.ttt3d.app.game;
 3 import edu.fau.eng.cop4331.ttt3d.app.Model;
5 import java.util.UUID;
 7 public class GameModel extends Model {
8
9
       /**
10
        * The View uses these constants to get data from the Model
11
        * The Controller uses these constants to update data in the Model
12
13
        * Every element in a view that needs to be updated
14
        * needs to have a UUID refrence to it here.
15
16
        * These are non-static so the UUID will
17
        * be unique to each instance of the class
18
19
        */
20
       public UUID MAIN = UUID.randomUUID();
21
       public UUID HELLO_WORLD_JLABEL = UUID.randomUUID();
22
       public UUID GAME GRID = UUID.randomUUID();
23
       public UUID[][][] GAME_GRID_BUTTONS;
24
       public UUID STAT_COUNTER;
25
26
        * initializes the UUIDs for GAME_GRID_BUTTONS
27
28
        */
29
       public GameModel() {
30
           this.GAME GRID BUTTONS = new UUID[3][3][3];
           for (int z = 0; z < this.GAME_GRID_BUTTONS.length; z++) {</pre>
31
32
               for (int y = 0; y < this.GAME_GRID_BUTTONS.length; y++) {</pre>
                    for (int x = 0; x < this.GAME_GRID_BUTTONS.length; x++) {</pre>
33
                        this.GAME_GRID_BUTTONS[x][y][z] = UUID.randomUUID();
34
35
36
               }
37
           }
38
       }
39
40
41
42
        * Holds the state of the game
43
        * 1 = X
44
        * 0 = empty
45
        * -1 = 0
46
        * @param gameState3D int[][][]
47
48
49
       public record gameState3D(int[][][] gameState3D){}
50
51
52
        * Holds the number of wins, losses, and ties
53
        * to be displayed in the view
54
55
        * <a href="#">Oparam</a> wins int
        * Oparam losses int
56
        * @param ties int
57
58
        */
       public record stats(
59
               int wins,
60
               int losses,
61
62
               int ties
63
       ){}
64
65
66 }
```

File - GameModel.java

```
File - MultiPlayerHostController.java
 1 package edu.fau.eng.cop4331.ttt3d.app.game;
 3 import edu.fau.eng.cop4331.ttt3d.app.Controller;
 5 public class MultiPlayerHostController extends Controller {
 7
        GameModel model;
 8
       GameView view;
 9
10
        /**
11
         * Constructor
12
13
         * @param gameModel GameModel
14
         * @param gameView GameView
15
         */
       public MultiPlayerHostController(GameModel gameModel, GameView gameView) {
16
17
            this.model = gameModel;
18
            this.model.register(gameView);
19
            this.view = gameView;
20
            this.view.registerController(this);
21
22
            setup();
       }
23
24
25
        /**
        * Setup the controller
26
27
        */
28
       void setup(){
29
       };
30
31
32
        //used when hosting a game for another player and yourself
33
34 }
35
```

```
File - SinglePlayerGameController.java
 1 package edu.fau.eng.cop4331.ttt3d.app.game;
 3 import edu.fau.eng.cop4331.ttt3d.app.Controller;
 4 import edu.fau.eng.cop4331.ttt3d.app.Handler;
 5 import edu.fau.eng.cop4331.ttt3d.util.Solver;
 7 import javax.swing.*;
 8 import java.awt.event.ActionEvent;
 9 import java.util.Random;
10 import java.util.UUID;
11
12 public class SinglePlayerGameController extends Controller {
13
       //controller that connects the view with a single player game model
14
       GameModel model;
15
16
       GameView view;
17
       public SinglePlayerGameController(GameModel model, GameView view) {
18
19
            this.model = model;
20
            this.view = view;
21
            this.view.registerController(this);
22
23
            runHandlers();
24
            setup();
25
       }
26
27
       void setup() {
28
            newGame();
29
            resetStats();
30
            this.handlers.put(model.GAME_GRID, gridButtonPressedHandler());
31
       }
32
33
       //Event Handlers////////////
34
35
36
       /**
37
         * This handler recieves x,y cordinates of the button that was pressed
         * Oreturn A Handler that reacts to button presses on it's grid.
38
39
         */
40
       Handler gridButtonPressedHandler() {
41
            return new Handler() {
42
                @Override
                public void handle(ActionEvent value) {
43
                      System.out.println(value.getID());
44 //
                    //get int x,y and z from String "x,y,z"
45
46
                    String[] s = value.getActionCommand().split(",");
47
                    int x = Integer.parseInt(s[0]);
48
                    int y = Integer.parseInt(s[1]);
                    int z = Integer.parseInt(s[2]);
49
50
                    System.out.println(x + "," + y + "," + z);
51
                    makeMove(x, y, z, 1);
52
                }
53
            };
       }
54
55
56
       //Game logic//////////
57
       Solver solver = new Solver();
58
59
        /**
60
        *
         * Validates and makes a move and updates the model
61
62
         * Also tells user if the game was won and if so resets the game
63
         * @author Abir Faisal
64
65
         * Oparam X
66
         * @param y
```

```
File - SinglePlayerGameController.java
 67
          * @param z
 68
          * @param player
 69
 70
         void makeMove(int x, int y, int z, int player) {
             System.out.format("interpreting move xyz=%d,%d,%d player=%d", x, y, z, player);
 71
 72
             GameModel.gameState3D gs3d = (GameModel.gameState3D) this.model.getData(this.model.GameModel.gameState3D)
 73
 74
             //make sure the postion was empty
 75
             boolean isValidMove = isValidMove(x, y, z, gs3d.gameState3D());
 76
             UUID buttonUUID = this.model.GAME GRID BUTTONS[x][y][z];
 77
 78
 79
             //if valid then update model
 80
             if (isValidMove) {
                 System.out.println(" validMove");
 81
 82
                 //update the model
 83
                 int[][][] gs = gs3d.gameState3D();
                 gs[x][y][z] = (player == 1) ? 1 : -1; //X=1 0=-1
 84
 85
                 this.model.setData(buttonUUID, new GameModel.gameState3D(gs));
 86
                 //check if there is a winner
 87
                 gs3d = (GameModel.gameState3D) this.model.getData(this.model.GAME GRID);
 88
 89
                 int winner = solver.solve(gs3d.gameState3D());
 90
 91
                 //if no winner, make next move
 92
                 if (winner == 3) \{ //X \}
 93
                     System.out.println("X wins");
 94
                     updateStats(1);
 95
                     JOptionPane.showMessageDialog(null, "You won");
 96
                     newGame();
 97
                 } else if (winner == -3) { //0
 98
                     System.out.println("O wins");
 99
                     updateStats(-1);
100
                     JOptionPane.showMessageDialog(null, "You lost");
101
                     newGame();
102
103
                 else if (tiedGame()) {
104
                     System.out.println("Tied Game");
105
                     updateStats(0);
                     JOptionPane.showMessageDialog(null, "The game was tied");
106
                     newGame();
107
108
                 else if (player == 1) makeNextMove(gs);
109
110
111
             } else System.out.println(" invalidMove");
112
         }
113
114
115
         /**
          * Updates the game stats in the model with the new values
116
117
118
          * @author Abir Faisal
119
          * @param winLossTie 1=win -1=loss 0=tie
120
          */
121
         void updateStats(int winLossTie){
122
             //get data from model
             GameModel.stats stats = (GameModel.stats) this.model.getData(this.model.STAT_COUNTER
123
124
             GameModel.stats newStats = null;
125
             switch (winLossTie){
126
127
                 case 1: {
128
                     //update the stats
129
                     newStats = new GameModel.stats(stats.wins() + 1, stats.losses(), stats.ties(
130
                     break;
                 }
131
132
                 case -1: {
```

```
File - SinglePlayerGameController.java
133
                      //update the stats
134
                     newStats = new GameModel.stats(stats.wins(), stats.losses() + 1, stats.ties(
135
                     break;
                 }
136
137
                 case 0: {
138
                     //update the stats
                     newStats = new GameModel.stats(stats.wins(), stats.losses(), stats.ties() + 1
139
140
                     break;
                 }
141
142
             }
             //update the model with the new stats
143
144
             this.model.setData(this.model.STAT_COUNTER, newStats);
145
         }
146
147
         /**
148
         * Check if the game is tied
149
150
         * <u>Oreturn</u> true = tied, false = not tied
151
         */
152
         boolean tiedGame(){
153
             return false; //TODO
154
         }
155
156
         /**
157
         * Setup a new game
          * <u>@author</u> Abir Faisal
158
159
         * setup a new game
160
         */
161
         void newGame() {
162
             //empty game grid
             //should init to zeros automatically
163
             int[][][] newGameState = new int[3][3][3];
164
             this.model.setData(model.GAME_GRID,
165
166
                     new GameModel.gameState3D(newGameState)
167
             );
         }
168
169
170
         void resetStats(){
171
             this.model.setData(model.STAT_COUNTER, new GameModel.stats(0,0,0));
         }
172
173
174
         * check if the move is a valid move
175
176
         *
177
         * <u>@author</u> Abir Faisal
178
          * @param x coordinate
179
          * @param y coordinate
          * @param z coordinate
180
181
          * @param gameState
          * Oreturn true if the move is valid, false if it is invalid.
182
183
184
         boolean isValidMove(int x, int y, int z, int[][][] gameState) {
             if (gameState[x][y][z] == 0) return true;
185
186
             else return false;
187
         }
188
189
          * Single player mode
190
191
          * Computer makes next move
192
         *
193
          * <u>@author</u> Abir Faisal
194
         */
195
         void makeNextMove(int[][][] gameState) {
196
             //select random position
197
             Random r = new Random();
198
             int x = r.nextInt(gameState.length);
```

```
File - SinglePlayerGameController.java
199
             int y = r.nextInt(gameState[x].length);
             int z = r.nextInt(gameState[x][y].length);
200
201
202
             System.out.println("\ncomputer move " + x + "," + y + "," + z);
203
204
             //validate decision
             while (gameState[x][y][z] != 0) {
205
206
                 System.out.println("NOT VALID RECALCULATING");
207
                 x = r.nextInt(gameState.length);
208
                 y = r.nextInt(gameState[x].length);
209
                 z = r.nextInt(gameState[x][y].length);
210
211
             makeMove(x, y, z, 0); //player 0 is always opponent
212
213
        }
214 }
```

```
File - MultiPlayerClientController.java
 1 package edu.fau.eng.cop4331.ttt3d.app.game;
 3 import edu.fau.eng.cop4331.ttt3d.app.Controller;
 5 public class MultiPlayerClientController extends Controller {
 7
        GameModel model;
 8
       GameView view;
 9
        public MultiPlayerClientController(GameModel gameModel, GameView gameView) {
10
11
            this.model = gameModel;
12
            this.model.register(gameView);
13
            this.view = gameView;
            this.view.registerController(this);
14
15
16
            setup();
       }
17
18
19
20
       void setup(){
21
       }
22
23
24
25
        //Use cases
        //When the user wants to connect to a multipleyer server
26
27
        //When the user wants to connect to single host
28
29 }
30
```

```
File - StartScreenView.java
 1 package edu.fau.eng.cop4331.ttt3d.app.startscreen;
 3 import edu.fau.eng.cop4331.ttt3d.app.App;
 4 import edu.fau.eng.cop4331.ttt3d.app.Updater;
 5 import edu.fau.eng.cop4331.ttt3d.app.View;
 6 import edu.fau.eng.cop4331.ttt3d.app.game.GameType;
 8 import javax.swing.*;
 9 import javax.swing.event.DocumentEvent;
10 import javax.swing.event.DocumentListener;
11 import java.awt.*;
12 import java.awt.event.ActionEvent;
13 import java.time.Instant;
14 import java.util.UUID;
16 public class StartScreenView extends View {
17
        //The model that this view will reference
18
19
        //when it needs to update
20
       StartScreenModel model;
21
22
        /**
23
        * Instantiate and setup the View
24
25
        * @author Abir Faisal
26
        * @param startScreenModel StartScreenModel
27
28
       public StartScreenView(StartScreenModel startScreenModel) {
29
           this.model = startScreenModel; //make view aware of model
30
            this.model.register(this); //make model aware of view
31
            setup(); //setup the view
32
       }
33
        //set up the view
34
35
       @Override
       public void setup() {
36
            JPanel mainJPanel = new JPanel();
37
38
           this.jFrames.put(model.MAIN, mainJPanel);
39
40
            //centering panel for asthetic purposes.
41
            JPanel centeringPanel = new JPanel();
42
            centeringPanel.setLayout(new BoxLayout(centeringPanel, BoxLayout.Y_AXIS));
43
44
           centeringPanel.add(new JLabel("Server IP"));
45
           centeringPanel.add(serverIPJTextField());
46
           centeringPanel.add(new JLabel("Server Port"));
47
48
           centeringPanel.add(serverPortJTextField());
49
50
           centeringPanel.add(startSinglePlayerGameButton());
51
           centeringPanel.add(startMultiPlayerGameButton());
52
           centeringPanel.add(startHostGameButton());
53
54
55
            //put centering panel in mainJpanel
56
           this.jFrames.get(model.MAIN).add(centeringPanel);
       }
57
58
59
       //NOTE: Try to keep these methods in order as they appear visually
60
61
62
        * Text field where the user enters the server IP
63
64
        * @author Abir Faisal
         * Oreturn a JTextField for the user to type in the server IP and port
65
66
```

```
File - StartScreenView.java
 67
        JTextField serverIPJTextField() {
 68
             JTextField serverIPTextField = new JTextField("0.0.0.0");
 69
             serverIPTextField.setMaximumSize(new Dimension(300, 25));
 70
             UUID uuid = this.model.SERVER_IP_TEXT_FIELD;
 71
 72
             //when the text field is changed
 73
             //notify the controller of the change
 74
             DocumentListener dl1 = new DocumentListener() {
 75
                 @Override
                 public void insertUpdate(DocumentEvent e) {
 76
                     controller.handle(uuid,
 77
 78
                             new ActionEvent(serverIPTextField, 0, serverIPTextField.getText())
 79
                     );
 80
                 }
 81
                 @Override
                 public void removeUpdate(DocumentEvent e) {
 82
 83
                     controller.handle(uuid,
 84
                             new ActionEvent(serverIPTextField, 0, serverIPTextField.getText())
 85
                     );
                 }
 86
                 @Override
 87
 88
                 public void changedUpdate(DocumentEvent e) {}
 89
             };
 90
             serverIPTextField.getDocument().addDocumentListener(dl1);
 91
 92
             //updates the UI if there is a change in the Model
 93
             Updater updater = () -> {
 94
 95
                 //get the data from the model as ServerInfo
 96
                 StartScreenModel.ServerIP ip = (StartScreenModel.ServerIP) this.model.getData(uu:
 97
 98
                 //if the text is different then update it, else do nothing
 99
                 if (!serverIPTextField.getText().equals(ip.ipAddress())) {
100
                     //set text without triggering listner
101
                     serverIPTextField.getDocument().removeDocumentListener(dl1);
102
                     serverIPTextField.setText(ip.ipAddress());
103
                     //restore the change listener
104
                     serverIPTextField.getDocument().addDocumentListener(dl1);
105
                 }
             };
106
             this.updateMethods.put(uuid, updater);
107
108
109
             return serverIPTextField;
        }
110
111
112
         /**
113
114
          * Text field where the user enters the server port number
115
          * @author Abir Faisal
116
117
          * @return
118
          */
119
        JTextField serverPortJTextField() {
120
             JTextField iTextField = new JTextField("1234");
121
             jTextField.setMaximumSize(new Dimension(300, 25));
122
             UUID uuid = this.model.SERVER_PORT_TEXT_FIELD;
123
124
             //when the text field is changed
125
             //notify the controller of the change
126
             DocumentListener dl = new DocumentListener() {
127
                 @Override
128
                 public void insertUpdate(DocumentEvent e) {
129
                     controller.handle(uuid,
130
                             new ActionEvent(jTextField, 0, jTextField.getText())
                     );
131
                 }
132
```

```
133
                @Override
134
                public void removeUpdate(DocumentEvent e) {
135
                    controller.handle(uuid,
                             new ActionEvent(jTextField, 0, jTextField.getText())
136
137
                     );
138
                }
139
                @Override
140
                public void changedUpdate(DocumentEvent e) {}
141
            ¡TextField.getDocument().addDocumentListener(dl);
142
143
144
145
            //updates the UI if there is a change in the Model
146
            Updater updater = () -> {
147
                //get the data from the model as ServerInfo
148
149
                StartScreenModel.ServerPort port =
150
                         (StartScreenModel.ServerPort) this.model.getData(uuid);
151
152
                //if the text is different then update it, else do nothing
153
                if (!jTextField.getText().equals(port.port())) {
154
                     //set text without triggering event
155
                    jTextField.getDocument().removeDocumentListener(dl);
156
                    jTextField.setText(port.port());
157
                     //restore the change listener
158
                    jTextField.getDocument().addDocumentListener(dl);
                }
159
160
161
162
            };
            this.updateMethods.put(uuid, updater);
163
164
            return jTextField;
165
        }
166
167
168
169
170
        //TODO convert to a loop
171
172
        /**
173
         * Button that starts a single player game
174
175
         * @author Abir Faisal
176
         * <u>@return</u>
177
         */
178
        JButton startSinglePlayerGameButton() {
179
            //instantiate the button
            JButton jButton = new JButton("Single Player");
180
181
            UUID uuid = this.model.START_SINGLE_PLAYER_GAME_BUTTON;
182
183
            jButton.addActionListener(actionEvent -> {
184
                this.controller.handle(uuid, actionEvent);
185
            });
186
187
            return jButton;
        }
188
189
190
191
         * Button that starts a multi player game
192
193
         * @author Abir Faisal
194
         * @return
195
196
        JButton startMultiPlayerGameButton() {
197
            //instantiate the button
198
            JButton jButton = new JButton("Multi Player");
```

File - StartScreenView.java

```
File - StartScreenView.java
199
             UUID uuid = this.model.START_MULTI_PLAYER_GAME_BUTTON;
200
201
             jButton.addActionListener(actionEvent -> {
202
                 this.controller.handle(uuid, actionEvent);
203
             });
204
             return jButton;
205
         }
206
         /**
207
208
          * Button that starts a hosting a game for one other player
209
210
          * <u>@author</u> Abir Faisal
211
          * <u>@return</u>
212
         JButton startHostGameButton() {
213
             //instantiate the button
214
215
             JButton jButton = new JButton("Host Game");
             UUID uuid = this.model.START_MULTI_HOST_GAME_BUTTON;
216
217
218
             jButton.addActionListener(actionEvent -> {
                 this.controller.handle(uuid, actionEvent);
219
220
             });
221
             return jButton;
222
223
         }
224 }
225
```

```
1 package edu.fau.eng.cop4331.ttt3d.app.startscreen;
 3 import edu.fau.eng.cop4331.ttt3d.app.Model;
5 import java.util.UUID;
7
8 public class StartScreenModel extends Model {
9
10
11
        * The View uses these constants to get data from the Model
        * The Controller uses these constants to update data in the Model
12
13
14
        * Every element in a view that needs to be updated
        * needs to have a UUID refrence to it here.
15
16
        * These are non-static so the UUID will
17
18
        * be unique to each instance of the class
19
20
        */
       public UUID MAIN = UUID.randomUUID();
21
22
       public UUID HELLO WORLD JLABEL = UUID.randomUUID();
23
       public UUID TEST BUTTON = UUID.randomUUID();
24
       public UUID SERVER_IP_TEXT_FIELD = UUID.randomUUID();
25
       public UUID SERVER_PORT_TEXT_FIELD = UUID.randomUUID();
       public UUID START_SINGLE_PLAYER_GAME_BUTTON = UUID.randomUUID();
26
       public UUID START_MULTI_PLAYER_GAME_BUTTON = UUID.randomUUID();
27
28
       public UUID START_MULTI_HOST_GAME_BUTTON = UUID.randomUUID();
29
30
       //example data strcuture holding some information to be
31
32
       //used by the view or updated by the controller
       public record ExampleDataStruct(
33
34
               String s,
35
               double n,
36
               int i,
37
               int[] arrayList
38
       ){}
39
40
       public record ServerIP(String ipAddress){}
41
       public record ServerPort(String port){}
42
43 }
44
```

File - StartScreenModel.java

```
File - StartScreenController.java
 1 package edu.fau.eng.cop4331.ttt3d.app.startscreen;
 3 import edu.fau.eng.cop4331.ttt3d.app.App;
 4 import edu.fau.eng.cop4331.ttt3d.app.Controller;
 5 import edu.fau.eng.cop4331.ttt3d.app.Handler;
 6 import edu.fau.eng.cop4331.ttt3d.app.game.GameType;
 7 import edu.fau.eng.cop4331.ttt3d.util.SettingsManager;
 9 import java.awt.event.ActionEvent;
10 import java.util.UUID;
11
12 public class StartScreenController extends Controller {
13
14
       StartScreenModel model;
       StartScreenView view;
15
16
17
        /**
        * Constructor
18
19
        * Oparam scm StartScreenModel
20
        * @param scv StartScreenView
21
       public StartScreenController(StartScreenModel scm, StartScreenView scv) {
22
23
           this.model = scm;
24
            this.view = scv;
25
           this.view.registerController(this);
26
27
           runHandlers();
28
           System.out.println("running event handlers");
29
            setup();
30
       }
31
32
       /**
33
        * Setup the view
34
35
       void setup() {
           handlers.put(model.START_SINGLE_PLAYER_GAME_BUTTON, startSinglePlayerGameHandler());
36
37
           handlers.put(model.START_MULTI_PLAYER_GAME_BUTTON, startMultiPlayerGameHandler());
38
           handlers.put(model.START_MULTI_HOST_GAME_BUTTON, startHostGameHandler());
39
40
           handlers.put(model.SERVER_IP_TEXT_FIELD, serverIPInfoUpdateHandler());
41
           handlers.put(model.SERVER_PORT_TEXT_FIELD, serverPortUpdateHandler());
42
            //load settings
43
44
           SettingsManager sm = SettingsManager.getInstance();
           String ipAddress = sm.getSettings().getString("userDefinedServer");
45
46
           String port = sm.getSettings().getString("userDefinedPort");
47
48
            //data type used by model
           StartScreenModel.ServerIP serverIP = new StartScreenModel.ServerIP(ipAddress);
49
50
           StartScreenModel.ServerPort serverPort = new StartScreenModel.ServerPort(port);
51
52
            //set the data in the model
53
           this.model.setData(model.SERVER_IP_TEXT_FIELD, serverIP);
54
           this.model.setData(model.SERVER_PORT_TEXT_FIELD, serverPort);
55
       }
56
57
58
59
       //Action handlers
        /**
60
        * Save server settings and tell Application
61
62
        * to launch GameType.SINGLE_PLAYER_GAME
63
        * @author Abir Faisal
64
        * Oreturn a Handler that launches a single player game
65
66
```

```
67
        Handler startSinglePlayerGameHandler() {
 68
            StartScreenController instance = StartScreenController.this;
 69
            return new Handler() {
 70
                @Override
 71
 72
                public void handle(ActionEvent value) {
                     System.out.println("Start Single Player Button Pressed");
 73
 74
                     //save the settings
 75
                     saveUserSettings();
 76
                     //launch the game
 77
                     App.getInstance().launchGame(GameType.SINGLE PLAYER GAME);
 78
                }
 79
            };
        }
 80
 81
 82
 83
         * Save server settings and tell Application
 84
         * to launch GameType.MULTI_PLAYER_CLIENT_GAME
 85
 86
         * @author Abir Faisal
         * Oreturn a Handler that launches a single player game
 87
 88
         */
 89
        Handler startMultiPlayerGameHandler() {
 90
            StartScreenController instance = StartScreenController.this;
 91
 92
            return new Handler() {
 93
                @Override
 94
                public void handle(ActionEvent value) {
 95
                     System.out.println("Start Multi Player Button Pressed");
 96
                     //save the settings
 97
                     saveUserSettings();
 98
                     //launch the game
 99
                     App.getInstance().launchGame(GameType.MULTI_PLAYER_CLIENT_GAME);
                }
100
101
            };
102
        }
103
104
105
         * Save server settings and tell Application
         * to launch GameType.MULTI_PLAYER_HOST_GAME
106
107
108
         * @author Abir Faisal
         * Oreturn a Handler that launches a single player game
109
110
111
        Handler startHostGameHandler() {
112
            StartScreenController instance = StartScreenController.this;
113
            return new Handler() {
114
115
                @Override
                public void handle(ActionEvent value) {
116
117
                     System.out.println("Start Host Game Button Pressed");
118
                     //save the settings
119
                     saveUserSettings();
120
                     //launch the game
                     App.getInstance().launchGame(GameType.MULTI_PLAYER_HOST_GAME);
121
                }
122
123
            };
        }
124
125
126
127
128
        /**
129
         * When the user changes the server IP
130
131
         * <u>@author</u> Abir Faisal
132
         * Oreturn A handler that updates the model with a new value
```

File - StartScreenController.java

```
File - StartScreenController.java
133
134
        Handler serverIPInfoUpdateHandler(){
135
            UUID uuid = model.SERVER_IP_TEXT_FIELD;
            StartScreenController instance = StartScreenController.this;
136
137
138
            return new Handler() {
139
                @Override
140
                public void handle(ActionEvent value) {
                    String serverIP = value.getActionCommand(); //get the IP:Port
141
                     //Update the model with the IP
142
143
                    StartScreenController.this.model.setData(uuid, new StartScreenModel.ServerIP
144
                    System.out.println(instance.model.getData(uuid));
145
                }
146
            };
147
        }
148
149
        /**
150
         * When the user changes the server Port
151
152
         * @author Abir Faisal
153
         * Oreturn A handler that updates the model with a new value
154
         */
155
        Handler serverPortUpdateHandler(){
156
            UUID uuid = model.SERVER_PORT_TEXT_FIELD;
            StartScreenController instance = StartScreenController.this;
157
158
            return new Handler() {
159
160
                @Override
                public void handle(ActionEvent value) {
161
162
                    String serverPort = value.getActionCommand();
                     //update the model with the port
163
                    model.setData(uuid, new StartScreenModel.ServerPort(serverPort));
164
165
                    System.out.println(instance.model.getData(uuid));
                }
166
167
            };
168
        }
169
170
171
        172
        /**
173
         * Save the user settings from the ServerIP and port text input fields
174
175
         * @author Abir Faisal
176
177
         */
178
        void saveUserSettings() {
            StartScreenController instance = StartScreenController.this;
179
180
181
182
            //save the settings
183
            StartScreenModel.ServerIP serverIPRecord =
184
                     (StartScreenModel.ServerIP) instance.model.getData(model.SERVER_IP_TEXT_FIELI
185
186
            StartScreenModel.ServerPort serverPortRecord =
                     (StartScreenModel.ServerPort) instance.model.getData(model.SERVER_PORT_TEXT_|
187
188
189
            SettingsManager.getInstance().setValue("userDefinedServer", serverIPRecord.ipAddress
            SettingsManager.getInstance().setValue("userDefinedPort", serverPortRecord.port());
190
191
        }
192
193 }
```

```
File - Solver.java
 1 package edu.fau.eng.cop4331.ttt3d.util;
 3 public class Solver {
       //class that contains game solvers
 5
        public Solver() {
 7
 8
 9
10
        * Solves the game given a 1D representation of the gameState
11
         * @param gameState1D
12
13
        public void solve(int[] gameState1D) {
14
            //TODO check winner given a 1D game state array
15
16
17
        /**
18
        * Solves the game given a 2D representation of the gameState
19
         * @param gameState2D
20
        public void solve(int[][] gameState2D) {
21
22
            //TODO check winner given a 2D game state array
23
24
25
        /**
26
         * Solves the game given a 3D representation of the gameState
27
28
         * @param gameState3D
29
30
        public int solve(int[][][] gameState3D) {
31
            int y0;
            int y1;
32
33
            int y2;
34
35
            int x0;
36
            int x1;
            int x2;
37
38
39
            //solve for horizontal and vertical wins
            for (int z = 0; z < 3; z++) { //layer
40
                for (int i = 0; i < 3; i++) {
41
42
                    v0 = gameState3D[i][0][z];
43
                    y1 = gameState3D[i][1][z];
44
                    y2 = gameState3D[i][2][z];
45
                    int hSum = y0 + y1 + y2;
46
47
                    x0 = gameState3D[0][i][z];
48
                    x1 = gameState3D[1][i][z];
49
                    x2 = gameState3D[2][i][z];
50
                    int vSum = x0 + x1 + x2;
51
52
                    if (hSum == 3) return hSum;
53
                    if (hSum == -3) return hSum;
54
                    if (vSum == 3) return vSum;
55
                    if (vSum == -3) return vSum;
56
                }
            }
57
58
59
            //check for diagonal wins
60
            for (int z = 0; z < 3; z++) {
                int topLeft = gameState3D[0][0][z]; int topRight = gameState3D[2][0][z];
61
62
                                   int center = gameState3D[1][1][z];
63
                int bottomLeft = gameState3D[0][2][z];int bottomRight = gameState3D[2][2][z];
64
65
                int diag1 = topLeft + center + bottomRight;
                int diag2 = bottomLeft + center + topRight;
66
```

```
File - Solver.java
  67
                      if (diag1 == 3) return diag1;
if (diag1 == -3) return diag1;
if (diag2 == 3) return diag2;
  68
  69
  70
                       if (diag2 == -3) return diag2;
  71
                 }
  72
  73
                 //check for orthogonal wins TODO
  74
  75
                 //no winners found
  76
                 return 0;
  77
            }
  78
  79
  80 }
  81
```

```
File - SettingsManager.java
   1 package edu.fau.eng.cop4331.ttt3d.util;
   3 import org.json.JSONObject;
   5 import java.io.File;
   6 import java.io.IOException;
   7 import java.io.PrintWriter;
   8 import java.nio.file.Files;
   9 import java.nio.file.Path;
 10
 11 import static java.lang.System.exit;
 12
 13 public class SettingsManager {
 14
 15
                private JSONObject settings;
                private final String settingsFileName = "settings.json";
 16
 17
 18
                //singleton pattern
 19
                private static SettingsManager instance;
 20
                private SettingsManager() {
 21
 22
                public static synchronized SettingsManager getInstance(){
 23
                         if (instance == null) instance = new SettingsManager();
 24
                         return instance;
                }
 25
 26
 27
                public void loadSettings() {
 28
                          //check if settings.json exists
 29
                         File file = new File(settingsFileName);
 30
 31
                         trv {
 32
                                   //if exist then load from file
 33
                                  if (file.exists()) {
 34
                                           String jsonString = Files.readString(Path.of(file.getPath()));
 35
                                           this.settings = new JSONObject(jsonString);
 36
                                  } else {
 37
 38
                                            //get default settings
 39
                                           String jsonString = new String(
 40
                                                             SettingsManager.class.getClassLoader().getResourceAsStream(settingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilestingsFilesting
 41
                                            );
 42
                                            //load into this and save to file
 43
                                           this.settings = new JSONObject(jsonString);
 44
                                           saveSettingsToFile();
 45
                                  }
 46
                         }catch (IOException e) {
 47
                                  System.out.println("Failed to load settings");
 48
 49
                         System.out.println(this.settings);//TODO remove?
 50
                }
 51
 52
                public JSONObject getSettings() {
 53
                         return settings;
 54
                }
 55
 56
                public synchronized void setValue(String key, Object value) {
 57
                         this.settings.put(key, value);
 58
                         saveSettingsToFile();
 59
 60
                synchronized void saveSettingsToFile() {
 61
 62
                          //save the changes to settings.json
 63
                         File file = new File(settingsFileName);
 64
 65
                         try {
                                  PrintWriter writer = new PrintWriter(file);
 66
```

```
File - SettingsManager.java
                   writer.write(this.settings.toString());
 67
                   writer.close();
 68
              }catch (IOException e) {
 69
 70
                   System.out.println(e);
                   System.out.println("failed to save settings");
 71
              }
 72
          }
 73
 74
 75
          @Override
         public String toString() {
    return this.settings.toString();
 76
 77
 78
 79 }
 80
```

```
File - Server.java
   1 package edu.fau.eng.cop4331.ttt3d.server;
   3 import java.io.IOException;
   4 import java.io.ObjectInputStream;
   5 import java.io.ObjectOutputStream;
   6 import java.net.ServerSocket;
   7 import java.net.Socket;
   8 import java.util.ArrayList;
   9
 10 public class Server {
 11
 12
                  /**
 13
                    * This is the server for clients
 14
                    * @author Abir Faisal, Jamahl Farrington
 15
 16
 17
                  ArrayList threads = new ArrayList<>();
 18
 19
                  ServerSocket server;
 20
                  int serverPort = 32034;
 21
 22
                  /**
 23
                   * Constructor
 24
                    * <a href="https://example.com/linearing/">
<a href="https://example.com
 25
 26
                  public Server() throws IOException {
 27
                           server = new ServerSocket(serverPort);
 28
                  }
 29
 30
                  /**
 31
                   * Run the server
 32
                    * @throws IOException
 33
 34
                  public void run() throws IOException {
 35
                           while (true) {
                                    System.out.println("Waiting for connection: " + server);
 36
 37
 38
                                     //created socket waits for connection
 39
                                    Socket socket = server.accept();
                                    System.out.println("Accepted Connection from : " + socket.getInetAddress());
 40
 41
 42
                                     //read from socket
                                    ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
 43
 44
                                    System.out.println(ois.readAllBytes());
 45
 46
                                     //process the input
 47
 48
                                     //respond to the client
 49
                                    ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
 50
                                    oos.writeObject(new String("Hello World"));
 51
 52
 53
                                     //close the connection
 54
                                    ois.close();
 55
                                    oos.close();
 56
                                     socket.close();
 57
                           }
 58
                  }
 59
 60
 61
                  /**
 62
                    * TODO write debug info without blocking
 63
                    * Oparam str String you want to print to terminal
 64
                    * Oparam log true=append to log file, false=do nothing
 65
                  void nonBlockingPrintln(String str, boolean log) {
 66
```

```
File- Service.java
1 package edu.fau.eng.cop4331.ttt3d.server;
2
3 //TODO make abstract
4 public interface Service {
5    Object getResponse();
6 }
7
```

```
File- ChatService.java
1 package edu.fau.eng.cop4331.ttt3d.server.services;
2
3 public class ChatService {
4    /**
5     * Handle the message sent from a client
6     * @param message Object that will be deserialized into a String
7     */
8     void processMessage(Object message){};
9 }
10
```

```
File-TTT3DService.java
1 package edu.fau.eng.cop4331.ttt3d.server.services;
2
3 public class TTT3DService {
4
5    /**
6    * Handle the move sent from a client
7    * @param move Object that will be deserialized into a gameState: int[][][]
8    */
9    void handleGame(Object move){
10    }
11 }
12
```

```
File-settings.json

1 {
2  "defaultServer": "127.0.0.1",
3  "userDefinedServer": "0.0.0.0",
4  "defaultPort": "32034",
5  "userDefinedPort": "1",
6  "rateLimitSeconds": "1"
7 }
```

```
1 import edu.fau.eng.cop4331.ttt3d.app.App;
 4 import org.junit.*;
 5
 6 import javax.swing.*;
 8 public class AppTest {
 9
10
       public AppTest(){
11
12
13
       }
14
15
       @BeforeClass
16
       public static void setUpClass() {
17
18
19
20
       @AfterClass
21
       public static void tearDownClass() {
22
23
24
       }
25
       @Before
26
27
       public void setUp(){
28
       }
29
30
31
       @After
32
       public void tearDown() {
33
       }
34
35
36
37
       //Test player ID generation
38
39
       @Test
       public void playerIDTest() throws Exception {
40
41
           App app = App.getInstance();
           String pidb = "";
42
43
           byte[] playerID = app.getClientID();
44
45
           for (int i = 0; i < playerID.length; i++) {</pre>
46
               pidb += playerID[i] + " ";
47
48
           System.out.println("player ID Bytes: " + pidb);
49
50
       }
51
52
       @Test
53
       public void test() {
54
55
           JOptionPane.showInputDialog("hello");
56
           JOptionPane.showConfirmDialog(null, "message", "tiTle", 1, 2);
57
       }
58
59
60
61 }
```

File - AppTest.java

```
File-ViewTest.java

1 public class ViewTest {
2 }
3
```

```
1 import edu.fau.eng.cop4331.ttt3d.app.Model;
 3 import org.junit.jupiter.api.Test;
 4 import static org.junit.jupiter.api.Assertions.*;
 5
 7 public class ModelTest {
8
9
       //Serialize Deserialize test
10
       @Test
       public void serDeserTest() throws Exception {
11
             Model m1 = new Model("Model1");
12 //
             System.out.println("Player ID: " + m1.getplayerIDasString());
13 //
14
             ObjectOutputStream objectOutputStream = new ObjectOutputStream();
15 //
16
17 //
             System.out.println(m1.getplayerID());
18
             assertEquals(m1.hashCode(), m1.hashCode());
19 //
20
       }
21
22
23 }
24
25
26
```

File - ModelTest.java

```
File - ServerTest.java
 1 import edu.fau.eng.cop4331.ttt3d.server.Server;
 2 import org.junit.jupiter.api.Test;
 4 import java.io.IOException;
 6 public class ServerTest {
 8
 9
10
11
        @Test
        public void testServer() throws IOException {
12
            Server server = new Server();
13
14 //
              server.run();
15
16
        }
17
18 }
19
```

```
File - AliceAndBob.java
 1 import org.junit.jupiter.api.Test;
 3 import javax.crypto.Cipher;
 4 import javax.crypto.CipherOutputStream;
 5 import javax.crypto.CipherSpi;
 6 import javax.crypto.NoSuchPaddingException;
 7 import javax.crypto.spec.IvParameterSpec;
 8 import java.io.IOException;
 9 import java.io.ObjectOutputStream;
10 import java.io.OutputStream;
11 import java.math.BigDecimal;
12 import java.math.BigInteger;
13 import java.security.NoSuchAlgorithmException;
14 import java.util.Arrays;
15 import java.util.Random;
16
17 public class AliceAndBob {
18
19
        BigInteger[] genPrimesArray(int len) {
20
            BigInteger[] secretNumbers = new BigInteger[len];
21
22
            for (int i = 0; i < len; i++) {
23
                secretNumbers[i] = BigInteger.probablePrime(16, new Random());
24
            }
25
            return secretNumbers;
26
        }
27
28
        BigInteger[] genSecretMods(BigInteger[] commonMods, BigInteger[] commonBases, BigInteger[
29
            BigInteger[] secretSauce = new BigInteger[secretMods.length];
30
31
            for (int i = 0; i < secretMods.length; i++) {</pre>
32
                BigInteger base = commonBases[i];
33
                BigInteger mod = secretMods[i]; //exponent
34
35
                //= base^mod % common mod
                secretSauce[i] = base.modPow(mod, commonMods[i]);
36
37
38
            return secretSauce;
39
        }
40
41
42
        private class TestObject {
43
            //object for encrypt/decrypt test
44
            private int i;
45
            public TestObject(int i){
46
                this.i = i;
47
48
            public int getI() {
49
                return i;
50
51
            public TestObject setI(int i) {
52
                this.i = i;
53
                return this;
54
            }
        }
55
56
57
58
59
        public void simDiffieHellmanKeyExchange() throws Exception {
60
61
            int len = 1;
62
63
            BigInteger[] pubMods = genPrimesArray(len);
64
            BigInteger[] pubBases = genPrimesArray(len);
65
            System.out.println("pubMods: " + Arrays.toString(pubMods));
66
                                               Page 1 of 2
```

```
File - AliceAndBob.java
            System.out.println("pubBases: " + Arrays.toString(pubBases));
 67
 68
 69
             //Alice's Secret
 70
            BigInteger[] alicePrivateMods = genPrimesArray(len);
            System.out.println("aliceSecret: " + Arrays.toString(alicePrivateMods));
 71
 72
            //Bob's Secret
 73
 74
            BigInteger[] bobPrivateMods = genPrimesArray(len);
 75
            System.out.println("bobSecret: " + Arrays.toString(bobPrivateMods));
 76
 77
             //generate a public key given the public and private mods
            BigInteger[] alicePubMods = genSecretMods(pubMods, pubBases, alicePrivateMods);
 78
 79
            System.out.println("alicePubMods: " + Arrays.toString(alicePubMods));
 80
            BigInteger[] bobPubMods = genSecretMods(pubMods, pubBases, bobPrivateMods);
 81
                                              " + Arrays.toString(bobPubMods));
 82
            System.out.println("bobPubMods:
 83
 84
             //Alice x Bob Key Exchange and Mix
 85
            BigInteger[] commonSecret1 = genSecretMods(pubMods, bobPubMods, alicePrivateMods);
 86
            System.out.println("commonSecret1: " + Arrays.toString(commonSecret1));
 87
            //Bob x Alice Key Exchange and Mix
 88
 89
            BigInteger[] commonSecret2 = genSecretMods(pubMods, alicePubMods, bobPrivateMods);
 90
            System.out.println("commonSecret2: " + Arrays.toString(commonSecret2));
 91
 92
 93
             //Test Encryption/Decryption
 94
            TestObject testObject = new TestObject(123);
 95
 96
 97
            String cipherMode = "AES/CBC/PKCS5Padding";
 98
            Cipher cipher;
 99
            cipher = Cipher.getInstance(cipherMode);
            //TODO cipher.init(Cipher.ENCRYPT MODE, secretKey, IvParameterSpec);
100
101
102
            OutputStream outputStream = null; //TODO
103
104
            CipherOutputStream cipherOutputStream = new CipherOutputStream(outputStream, cipher)
105
106
            cipherOutputStream.write(0);
107
108
109
110
            for (int i = 0; i < len; i++) {
                 assert commonSecret1[i].equals(commonSecret2[i]) : "Secrets are not common";
111
112
            }
113
        }
114
115 }
116
```

```
File-ControllerTest.java

1 public class ControllerTest {
2 }
3
```

```
File - PerformanceTest.java
 1 import org.junit.jupiter.api.Test;
 3 import java.util.Random;
 5 public class PerformanceTest {
 7
 8
 9
10
        //Standard array compare
11
        //Built in array compare
12
        //Arrays.equals()
13
        //Arrays.
14
        //Vector
15
        private int[] gameState1D = new int[27];
16
17
        private int[][][] getGameState3D = new int[3][3][3];
18
19
        private void initRandGameState() {
20
            Random r = new Random();
21
            for (int i = 0; i < gameState1D.length; i++) {</pre>
22
23
                //randomly assign a 1, 0, or -1
                int a = (r.nextBoolean()) ? 1:-1; // X or 0
24
25
                this.gameState1D[i] = r.nextBoolean() ? a : 0; //played or empty
26
            }
        }
27
28
29
30
        private void checkWinnerAlgo3DV2(int[][][] gameState3D) {
31
32
33
            for (int z = 0; z < gameState3D.length; <math>z++) {
34
35
                int xSum = 0;
36
37
                for (int x = 0; x < gameState3D[z].length; <math>x++) {
38
39
                     int ySum = 0;
40
41
                     for (int y = 0; y < gameState3D[z][x].length; y++) {</pre>
42
                         int a = gameState3D[x][y][z];
43
                         xSum += a;
44
                         ySum += a;
45
46
                    }
47
                    if (ySum == 3) {
48
                         System.out.println("winner found");
49
50
                     } else ySum = 0; //reset ySum
51
52
                if (xSum == 3) {
53
54
                     System.out.println("winner found");
55
                } else xSum = 0; //reset ySum
56
            }
        }
57
58
59
60
        //TODO try 3d array and compare performance
        private void checkWinnerAlgo3dV1(int[][][] gamestate3D) {
61
62
            //TODO find better names for these variables
63
64
            int originH;
65
            int originV;
            int pos2;
66
```

```
67
            int pos3;
 68
            int pos4;
 69
            int pos5;
 70
 71
            //for each layer as i
 72
            for (int i = 0; i < 3; i++) {
 73
 74
                 //check horizontal and vertical wins
 75
                 for (int j = 0; j < 3; j++) {
                     originH = gamestate3D[0][j][i];
 76
 77
                     pos2 = gamestate3D[1][j][i];
 78
                     pos3 = gamestate3D[2][j][i];
 79
 80
                     originV = gamestate3D[j][0][i];
 81
                     pos4 = gamestate3D[j][1][i];
 82
                     pos5 = gamestate3D[j][2][i];
 83
 84
                     // if these values are 3 or -3 we know
 85
                     // there is a winner
 86
                     // and that either X(3) or O(-3) has won
                     int hWinner = originH + pos2 + pos3;
 87
 88
 89
                     int vWinner = originV + pos4 + pos5;
 90
 91
                 }
            }
 92
 93
 94
            //TODO check diagonals
 95
 96
 97
 98
        }
 99
100
101
        //Check 1D gamestate
102
        private void checkWinnerAlgoV1(int[] gamestate){
103
104
            //check horizontal wins for each layer in the cube
105
            //TODO REFACTOR, maybe each check should be it's own method?
106
107
108
            int gs1 = 0;
109
            int gs2 = 0;
110
            int gs3 = 0;
111
112
            int i = 0;
113
            while (i < 27) {
114
115
                gs1 = gamestate[i];
116
                gs2 = gamestate[i+1];
117
                gs3 = gamestate[i+2];
118
                System.out.println("Game State: " + i + " " + gs1 + gs2 + gs3);
119
120
                if ((gs1 == gs2) && (gs2 == gs3)) {
121
                     System.out.println("Horizontal win: " + gs1 + gs2 + gs3);
                 }
122
123
                 i += 3;
            }
124
125
126
            //check vertical wins
            System.out.println("check Horizontal wins");
127
128
            i = 0;
129
            int j = 0;
            while (i < 27) {</pre>
130
131
                 j = 0;
132
                while (j < 3) {
```

File - PerformanceTest.java

```
File - PerformanceTest.java
133
                                      gs1 = gamestate[i+j];
 134
                                      gs2 = gamestate[i+j+3];
 135
                                      gs3 = gamestate[i+j+6];
 136
                                      System.out.println("Game State: " + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i + i +
 137
 138
 139
                                      if ((gs1 == gs2) && (gs2 == gs3)) {
 140
                                             System.out.println("Horizontal win: " + gs1 + gs2 + gs3);
                                      }
 141
 142
 143
                                      j++;
 144
                               }
 145
                               i += 9;
                       }
 146
 147
 148
                       //check side direction wins (Z-Axis wins)
 149
                       System.out.println("check Z-Axis wins");
 150
 151
                       while (i < 9) {
 152
                              gs1 = gamestate[i];
 153
                               gs2 = gamestate[i+9];
 154
                              gs3 = gamestate[i+18];
 155
 156
                              System.out.println("Game State: " + i + " " + gs1 + gs2 + gs3);
 157
 158
                               if ((gs1 == gs2) && (gs2 == gs3)) {
 159
                                      System.out.println("Horizontal win: " + gs1 + gs2 + gs3);
 160
                               }
 161
                               i++;
                       }
 162
 163
 164
                       //Check diagonal wins
 165
                       //Check diagonal wins from front
 166
 167
                       System.out.println("\nCheck diagonal wins from front");
 168
                       i=0;
                       while (i < 27) {
 169
 170
                              gs1 = gamestate[i];
 171
                               gs2 = gamestate[i+4];
 172
                              gs3 = gamestate[i+8];
 173
                              System.out.println("Game State \\: " + i + " " + gs1 + gs2 + gs3);
 174
 175
                               //check other way
 176
 177
                              gs1 = gamestate[i+2];
 178
                               //gs2 = gamestate[i+4]; // center no need to do this twice
 179
                              gs3 = gamestate[i+6];
 180
                              System.out.println("Game State /: " + i + " " + gs1 + gs2 + gs3);
 181
 182
                               i += 9;
 183
 184
                       }
 185
 186
                       //Check diagonal wins from top
                       System.out.println("\nCheck diagonal wins from top");
 187
 188
                       i=0;
                       while (i < 6) {
 189
                              gs1 = gamestate[i];
 190
                               gs2 = gamestate[i+10];
 191
 192
                              gs3 = gamestate[i+20];
 193
 194
                              System.out.println("Game State /: " + i + " " + gs1 + gs2 + gs3);
195
 196
                               //check other way
 197
                              gs1 = gamestate[i+2];
 198
                               //gs2 = gamestate[i+4]; // center no need to do this twice
```

```
199
               gs3 = gamestate[i+18];
200
201
               System.out.println("Game State \\: " + i + " " + gs1 + gs2 + gs3);
202
203
                i += 3;
204
           }
205
            //Check diagonal wins from side
206
           System.out.println("\nCheck diagonal wins from side");
207
208
209
           while (i < 3) {
210
               gs1 = gamestate[i];
211
               gs2 = gamestate[i+12];
212
               gs3 = gamestate[i+24];
213
214
               System.out.println("Game State /: " + i + " " + gs1 + gs2 + gs3);
215
216
                //check other way
217
               gs1 = gamestate[i+6];
218
                //gs2 = gamestate[i+12]; // center no need to do this twice
219
               gs3 = gamestate[i+18];
220
221
               222
223
                i ++;
           }
224
225
226
           //TODO check though center 3d wins
227
228
           printGameStateLayer(1, gamestate);
229
230
        }
231
232
233
        public void printGameStateLayer(int layer, int[] gamestate1d) {
234
           String s = "";
235
236
237
           for (int i = (layer * 9); i < 9; i++) {
238
                s += gamestate1d[i] + " ";
239
           }
240
           System.out.println("Game layer:" + layer + " " + s);
241
242
        }
243
244
245
246
247
        @Test
248
        public void perfTest() {
249
           initRandGameState();
250
            //byte[][][] anotherGameState = this.gamestate.clone();
251
252
           long startTime = 0;
253
           long endTime = 0;
254
           long diff = 0;
255
            long i = 1;
256
           long avg = 0;
257
258
           for (int j = 0; j < i; j++) {
259
                startTime = System.nanoTime();
260
261
               checkWinnerAlgoV1(this.gameState1D);
262
                endTime = System.nanoTime();
263
264
               diff = endTime - startTime;
```

File - PerformanceTest.java

File-ServerTortureTest.java

1 public class ServerTortureTest {
2
3 }
4

```
File - FunctionPrototypes.java
 1 import org.junit.jupiter.api.Test;
 3 public class FunctionPrototypes {
 5
        //used for prototpying methods and functions before implimenting into production
 6
 7
 8
 9
        //Transforms game from 3D to 1D
10
        @Test
        public void transform3Dto1D(int[][][] gameState3D) {
11
12
            int[] gamestate2D = new int[27];
13
14
            for (int z = 0; z < 3; z++) {
15
                for (int x = 0; x < 3; x++) {
                    for (int y = 0; y < 3; y++) {
16
                        gamestate2D[x+y+z] = gameState3D[x][y][z];
17
18
19
                }
20
            }
21 //
              return gamestate2D;
        }
22
23
24 }
25
```

```
File - SettingsManagerTest.java
 1 import edu.fau.eng.cop4331.ttt3d.util.SettingsManager;
 2 import org.junit.jupiter.api.Test;
 4 import java.util.Objects;
 5 import java.util.Random;
 7 public class SettingsManagerTest {
 8
 9
10
       /**
        * Test of settings load/save successfully.
11
12
        */
13
       @Test
       void settingsManagerTest() {
14
15
            SettingsManager sm = SettingsManager.getInstance();
            sm.loadSettings();
16
17
18
            String customPort = sm.getSettings().getString("userDefinedPort");
19
            System.out.println("before: " + customPort);
20
            sm.setValue("userDefinedPort", "0");
21
22
23
            sm.loadSettings();
            String newCustomPort = sm.getSettings().getString("userDefinedPort");
24
25
            System.out.println("after: " + newCustomPort);
26
27
            //i guarantee that this works in runtime.
28 //
              assert (!customPort.equals(newCustomPort)) : "Settings did not change";
       }
29
30
31
32 }
33
```