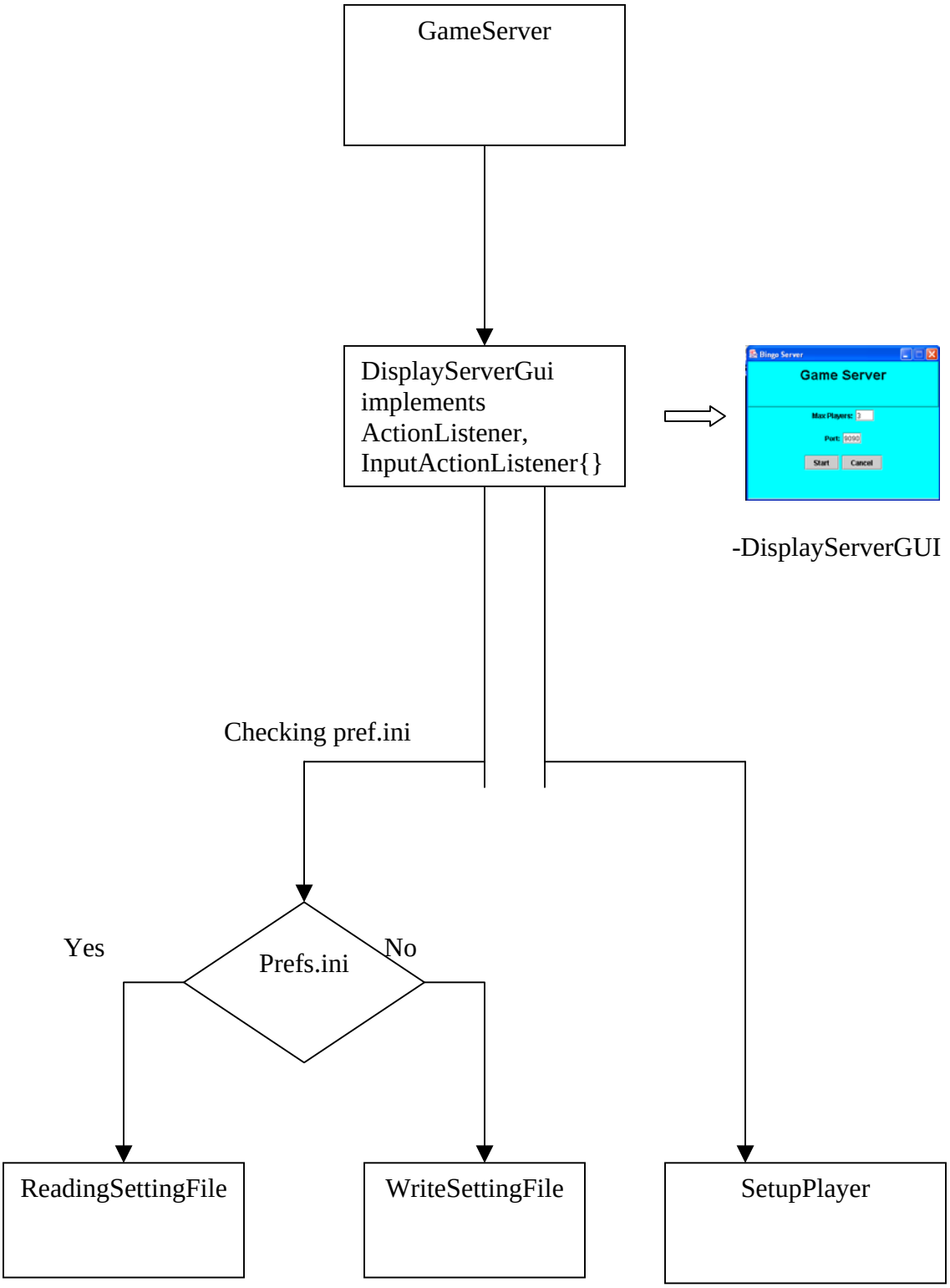
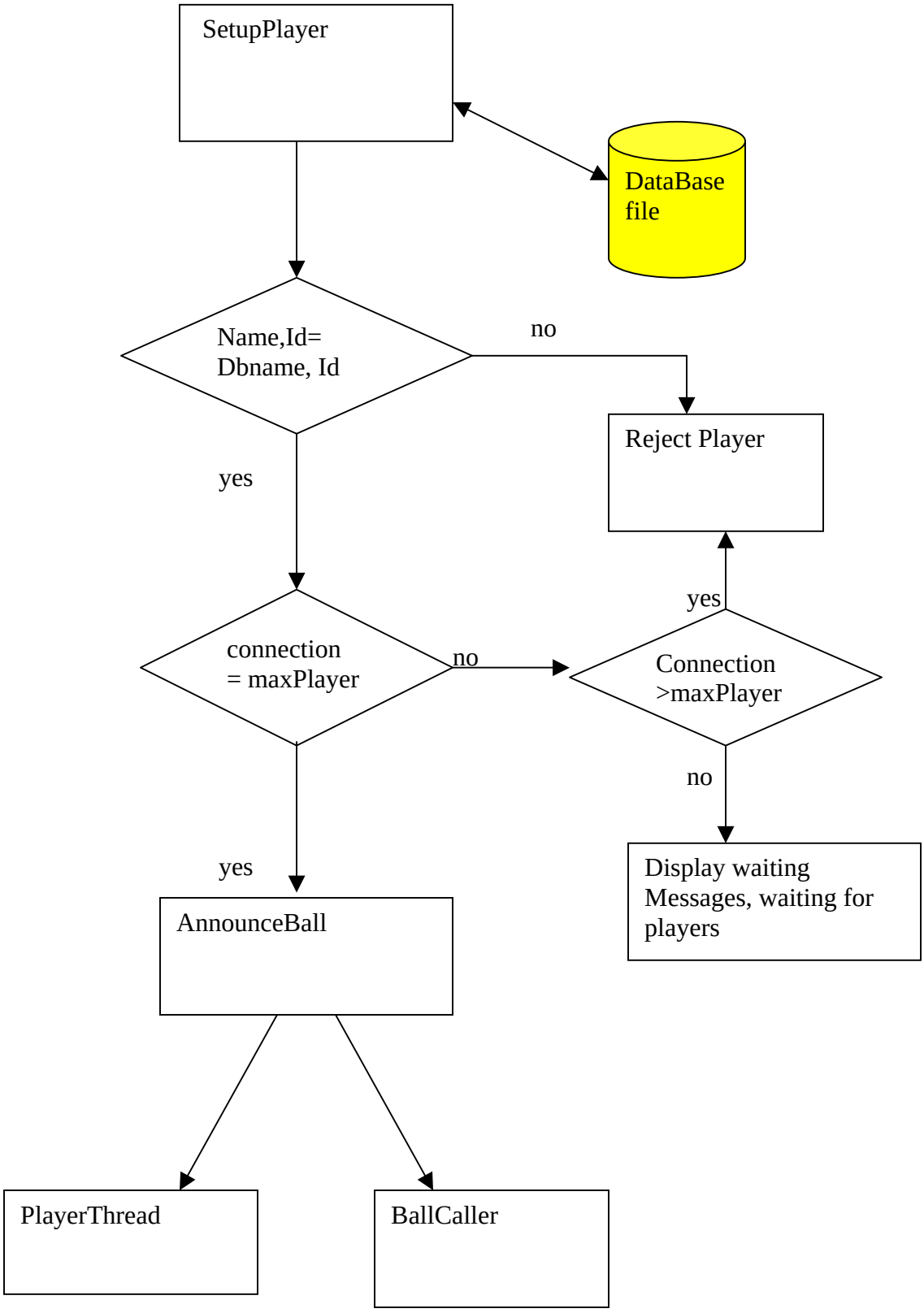
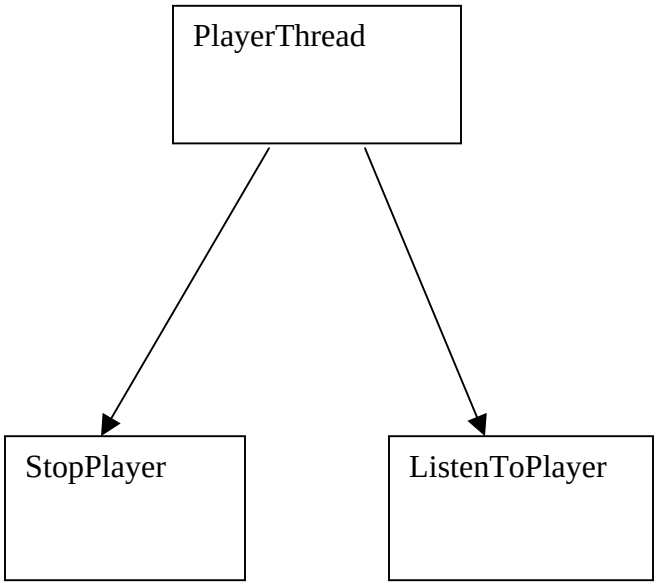
## GAME SERVER FLOWCHART




SetupPlayer
DataBase file
Name,Id= Dbname, Id
no
Reject Player
yes
yes
connection = maxPlayer
no
Connection >maxPlayer
no
yes
Display waiting Messages, waiting for players
AnnounceBall
PlayerThread
BallCaller



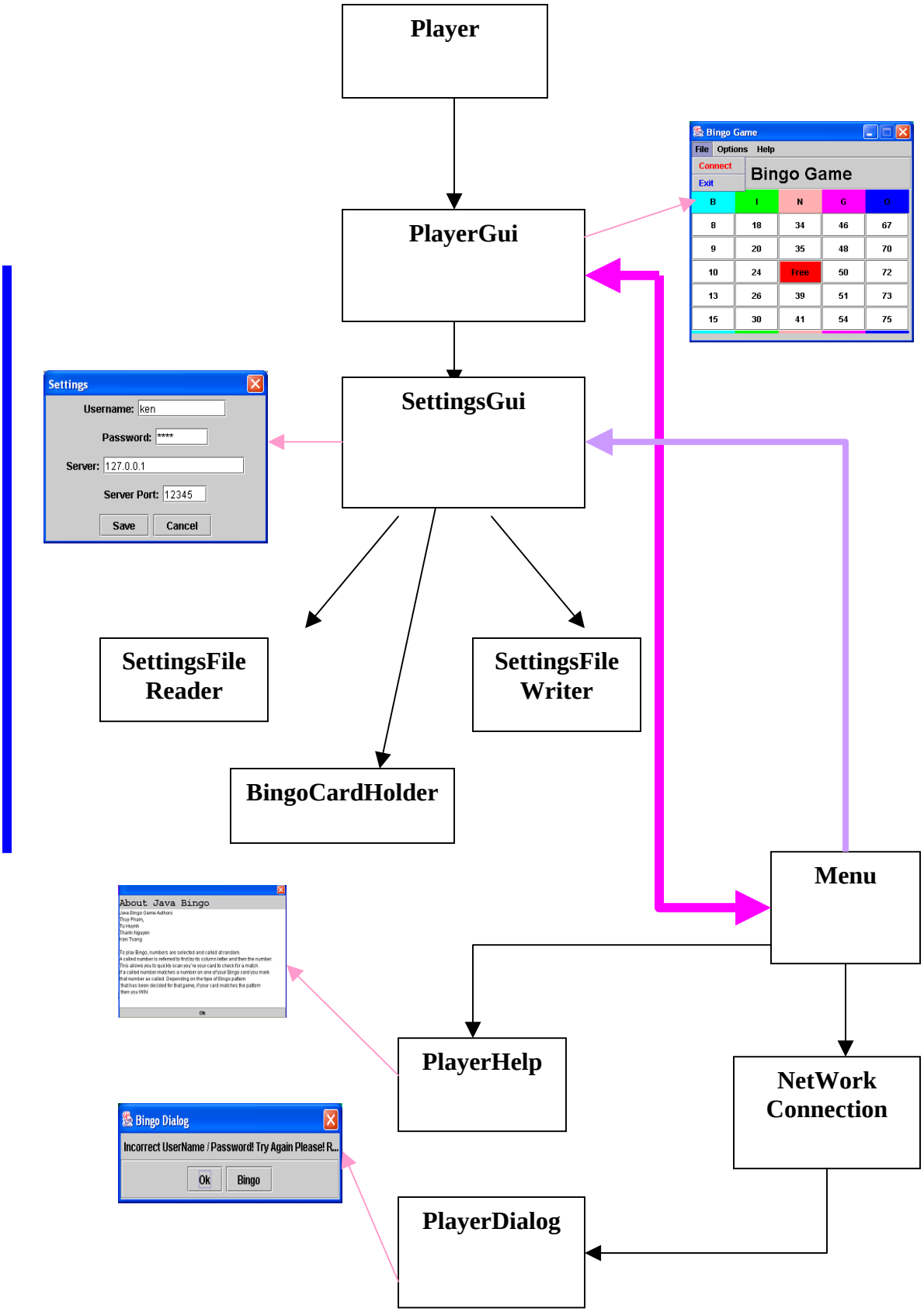

The flow chart leaves out some details but basically the flow of the server is:

- The game server must start up first and waiting for players.
- To join a game, a user must
  1- must have a account
  2- sign in with correct username and password
  3- click connect to make a connection.

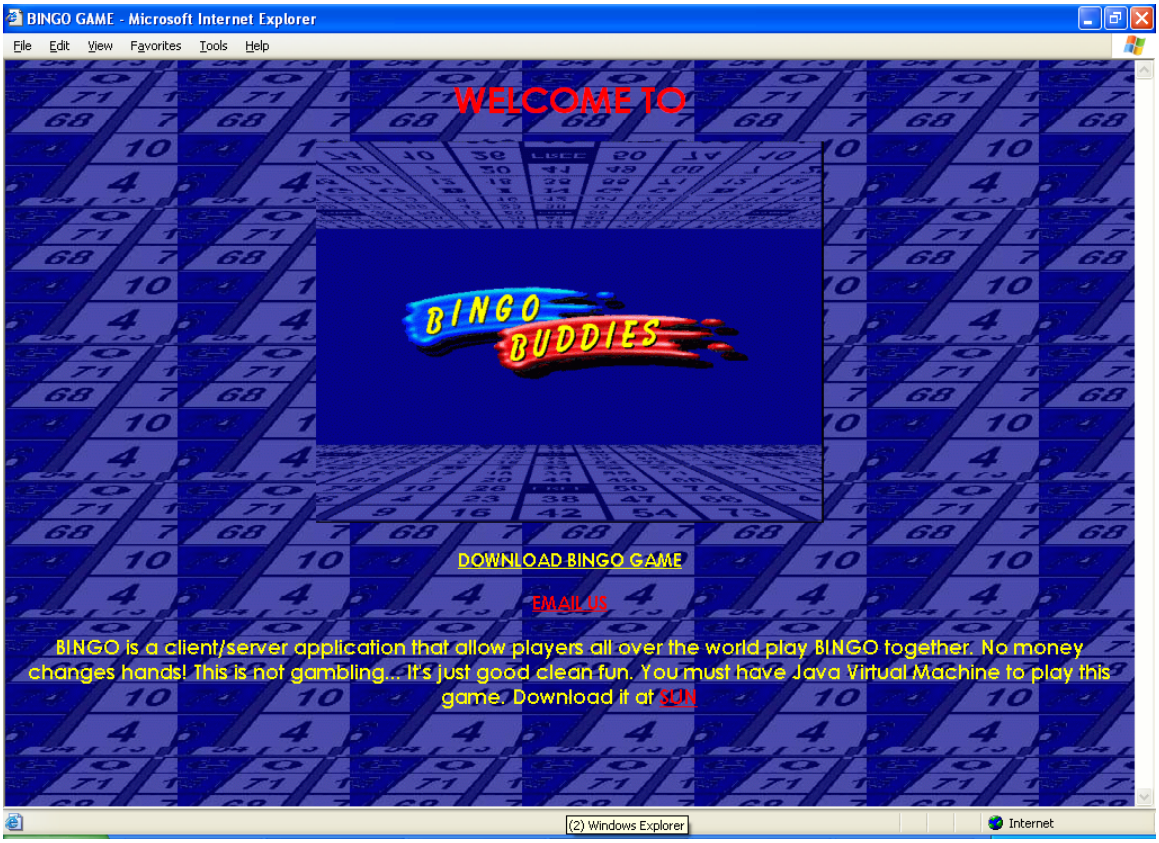
-After the first Player has joined a game, the Game Server begins to count coming connection to see if there are enough players to start the game.

-After the game is over, when there is a winner, the Game Sever restarts the game and wait for the players to join the next game.

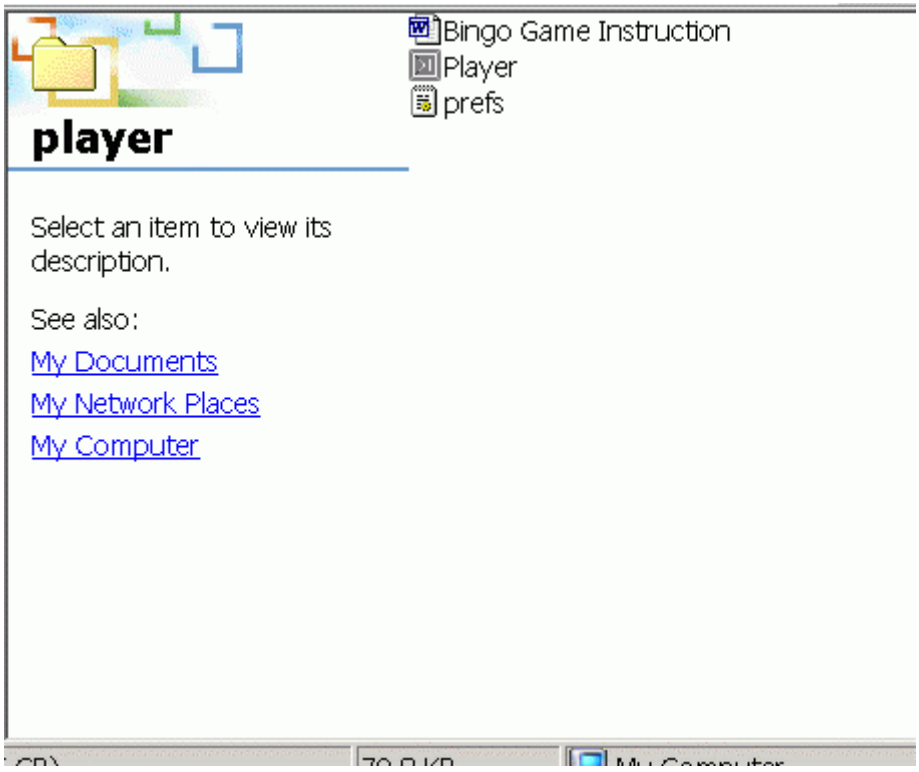
# Client Side Flow Chart



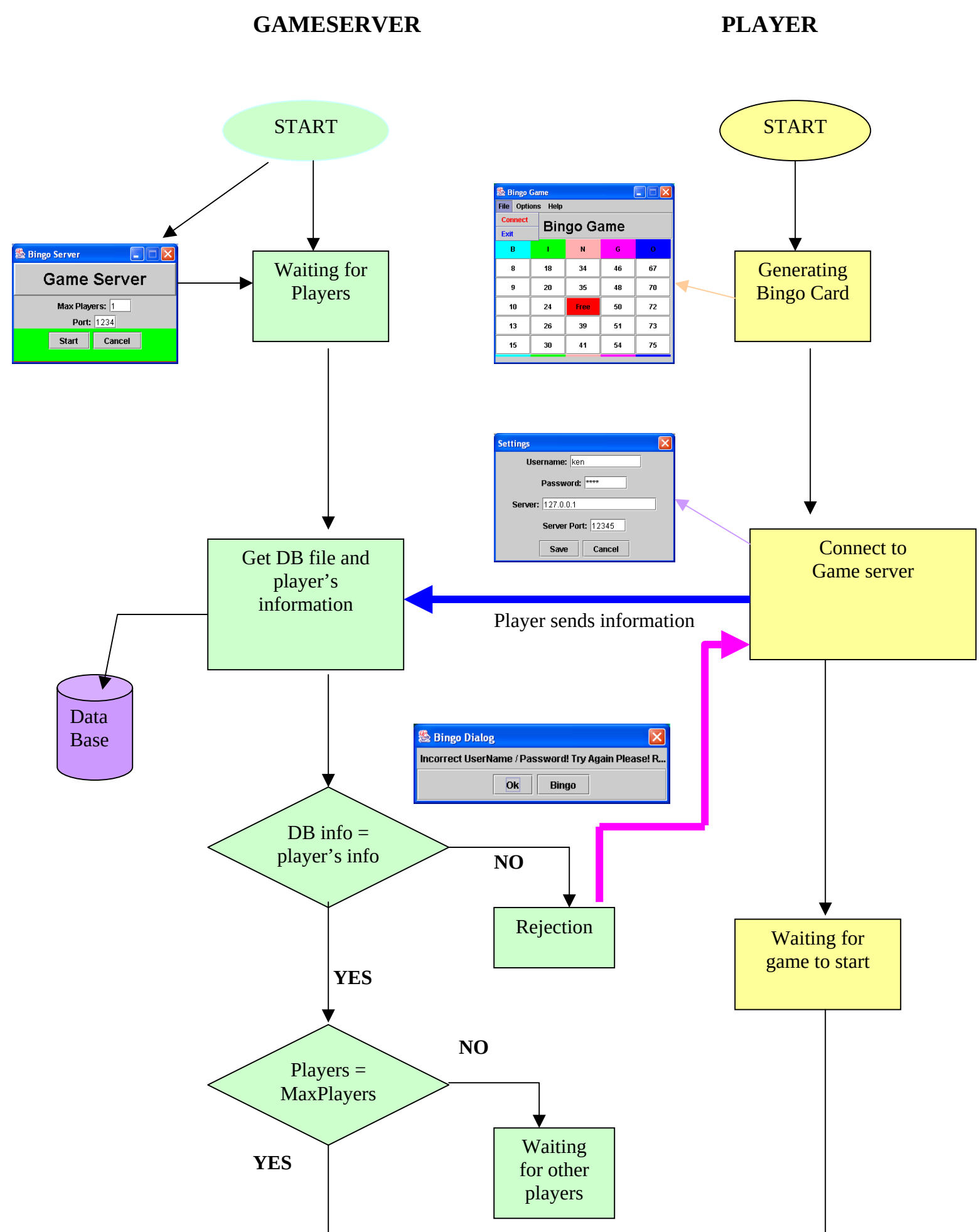
Here is our website.

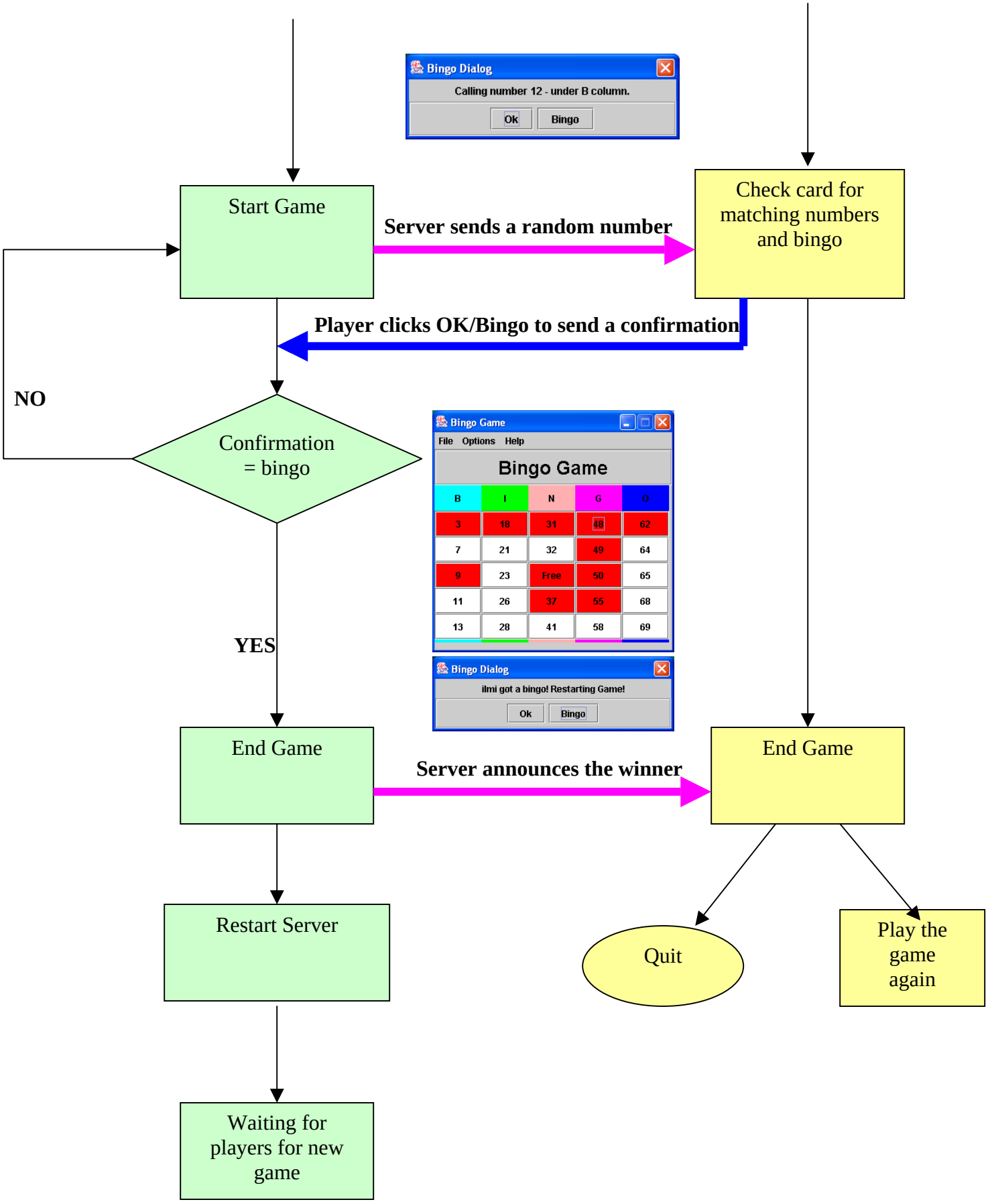


After downloading the Player package from our website and unzip it, here is what you should see in Player folder: one EXEC file to play the game, one prefs file, and one Bingo instruction file.  Please follow the instruction carefully.



# The Flow of Bingo Game




Bingo Dialog
Calling number 12 - under B column.
Ok
Bingo
Start Game
Server sends a random number
Check card for matching numbers and bingo
Player clicks OK/Bingo to send a confirmation
NO
Confirmation = bingo
Bingo Game
File Options Help
Bingo Game
B I N G O
3 18 31 48 62
7 21 32 49 64
9 23 Free 50 65
11 26 37 55 68
13 28 41 58 69
Bingo Dialog
ilmi got a bingo! Restarting Game!
Ok
Bingo
YES
End Game
Server announces the winner
End Game
Restart Server
Quit
Play the game again
Waiting for players for new game

TERM PROJECT

# BINGO GAME

**by Thuy Pham, Tu Huynh,**
**Thanh Nguyen, and Ken Tsang**

**Instructor: Ilmi Yoon**

**Class: CSC-667**

**Spring 2003**

# Description of the BINGO Programs

Our implementation of the **BINGO** game in Java is a client/server application, and as such, is comprised of two Java programs that run in separate Java Virtual Machines. The server is called the Game Server, and the client is called the Player. We will run this project on one machine but use two directories, one for the client and one for the server, to simulate the use of a remote site.

To play BINGO, a Game Server application is running somewhere on our network and Player must run the Player application. At the first time, the Player must go to our website (http://userww.sfsu.edu/~thuyp) to download the Player application that includes all Client's needed classes and instruction how to create a setting file. The server port is already set up, the Player only needs to follow the instruction carefully to create the setting file by choosing their name and password, and send it to the server administrator by email. After being added to our database, the Player is ready to play the game. The Player won't be allowed to play if his or her name and ID are not in our database or the server port number is incorrect.

**Game Server**

The Game Server manages bingo games. The sever administrator has the choice to make the server run automatically or generate a window that allows him or her to make some changes to the program setting. To start the game, the administrator of the Game Server pushes the **START** button to make the server on and waiting for players to connect. It verifies player's name and ID, starts and stops games, announces the balls as they are randomly chosen from a bag of balls, verifies player's confirmation, prevents players from being a nuisance, and provides status to game players.

Where there are enough players, the Game Server will start the game. It will announce a random number and send it to all players and wait for their confirmation before calling the next number. After waiting for 60 seconds with no confirmation, the Game Server will automatically kick out all players and restarts a new game. Also, after the winner is announced, the Game Server automatically restarts a new game after sixty seconds. The Game Server is on until the Game Server Administrator shut if off. After being closed, the Player won't be able to contact the Game Server anymore.

**Player**

The Player runs the Player application that has been downloaded from our website to generate a Bingo card. This card provides the interface for Players to interact with the Game Server. To play, a player pushes the **CONNECTION** button to make the connection to the Game Server. If the Player's name or ID is incorrect for some reasons, he or she will be rejected and asked to make the connection again. In this case, the Player should go to his or her **setting** on the Bingo card and re-input the correct ID and name and connects to the serve again. The Player is allowed to play if there are not enough players and the game has not been started yet. Otherwise, they will receive a waiting message from the server. When the game is started, the player marks the card as balls are announced and click the button **OK** on the Bingo Dialog to confirm to the Game Server that he or she officially gets a number. When the card has a BINGO (5 in a row, column, or diagonal) the user clicks the button **BINGO** on the Bingo Dialog that notifies the Game Server of the win. The Player can quit the game anytime by pushing the **EXIT** button on the Bingo card to exit the game.

## Features of the Bingo programs

The Game Server and the Player applications use many of the JDK features such as:

- The JFC ("Swing") User Interface classes
  - Using the JFC to implement the UI's. Displays the graphic interface (Bingo card) using JFS components to provide interface for players.
- Multi-threading and thread synchronization
  - There are many threads are created in the Game Server. Some of them run independently, and some of them must coordinate to other activities.
  - Coordinating Threads in the Game Server: for each game, SetupPlayer creates 2 threads, AnnounceBall, that announces the balls for the current game, and PlayerThread, to play the game. The activities of these threads must be coordinated with the activities of other threads in the program.
- Communication between the Game Server and the Player
  - The Player must initiate the communication.
  - The Game Server must be able to send a reply to the Player or announce a winner.
  - The Player must send a confirmation of receiving a number or detecting a Bingo to

the Game Server and also the name of the winner.

- Using JDBC to access Player's name and ID.
- Digital Signatures: check Player's name, ID, and verify the BINGO cards.
- Managing program settings in the Game Server and the Player
  - Allow the Game Server to change the number of maximum player and port number in preference file
  - Allow the Player to change the name, ID in setting file.

The BINGO game is implemented using approximately 24 classes contained in 2 packages:

- **`GameServer`**--contains the code that implements the Server application
- **`Player`**--contains the code that implements the Client application

**The Primary Classes**

This section provides a few classes that are the most important classes in the GameServer and Player applications and how they related.

**GameServer:**

***DisplayServerGUI:*** is called when the GameServer starts. It displays the graphic interface using JFS components. This class extends from JFrame class and implements from ActionListener and InputMethodListener. It displays the Game Server card that provides the interface for Game Server to start the connection, and the option to change the number of maximum of players or server port number. The Game

Server administrator pushes the button **START** to make the server on and waiting for connection from players.

***SetupPlayer***: is called when the GameServer starts. It opens a server socket and wait for the players to connect to the server. It gets the InputStream and OutputStream for the server. It verifies the port number to allow the connection. It also accesses the information in the database to verify Player's name and ID. When there are enough players, the game will start. It also creates a new thread of each player to the vector clients, and call AnnounceBall to start the game. It also removes all the players that store in the vector clients and restart server when the players stop their connection with the serve or if there is a winner.

***AnnounceBall***: Set all players in the vector clients to begin. It announces the ball and sent it to all players in the player's list. The BallCaller will be called in this class to pick a random ball from number 1 to number 75. It also checks the player's confirmation to see if there is a Bingo and reset the game.

***PlayerThread***: handles each server thread to each player. It gets the InputStream and OutputStream for a player to read message from the player and write response to the player. It checks to see if the message passed from players is a Bingo message or a receiving number message. If it's a Bingo message, then sends a Bingo message with the winner's name to all players.

***BallCaller***: randomly pick a ball with a number in the range from 1-75. When a ball is picked, it will be stored in an array. When it picks a new ball next time, it will go back to the array to see whether the new ball exists. It it's, it will pick another one to prevent a number is called twice.

***ListeningToPlayer***: this class is connected to PlayerThread class. It reads the message from the client's socket, and then sends it back to PlayerThread class to handle each different case.

**Player**

***PlayerGui:*** displays the graphic interface using JFS components. This class extends from JFrame class and implements from ActionListener and InputMethodListener. It displays the Bingo card that provides the interface for players to interact with the Game Server. To play, a player needs to push the **CONNECTION** button to make connection to the Game Server.

***SettingGui:*** displays the **setting** on the Bingo card that allows players to change their name, ID, and port number. This new information will override the setting in the preference file of the player.

***ClientDialog:*** let players make a confirmation to the Game Server by pushing the button BINGO or OK. Also it accepts the calling number and the winning message from the server.

***NetworkConnection:*** opens a player socket and send setting file to the Game Server to make the connection. It also reads in messages from the Game Server. On the player's setting file, the player must have the correct IP number of the Game Server. Otherwise, it won't be connected.

***BingoCardHolder:*** keeps track all the matching numbers on the Bingo card. If the number is matched with the announce number, that box will be marked red after player pushes the button **OK** on Bingo Dialog. When a player clicks on the **BINGO** button, it will check to see if there are five numbers on a road either vertically, horizontally, or diagonally. If it's not, it won't send a Bingo confirmation to the Game Server.

## Exercises

The following is a list of potential programming exercises that you can do to update, modify, or improve the Player or GameServer programs.

- Allow a Player to override MaxPlayers on the setting file so he or she doesn’t have to wait for having enough players.
- Allow Players to remember cards they like and reuse them.
- Set up a message center that lets Players send messages to each other through the Game.
- Use the new sound APIs to announce the balls with a voice.