# Board Game Client/Server Package

Eric Crawford (eric.crawford@mail.mcgill.ca)

March 4, 2015

## 1 Overview

This software consists of two packages, both implemented in Java:

1. boardgame: a generic board game client/server package
2. omweso: an implementation of the omweso game

The boardgame package is intended to allow the implementation of different board games and AI players with minimal effort. It provides networking support, log file management, and GUI support. No familiarity with this code is required to build an AI player, and only limited knowledge is required to implement an entire game.

To play, two clients (the players) connect to the server via TCP sockets. This allows the clients to be run on separate computers. A GUI which can be used to display an ongoing game or to examine existing log files is provided, but is not required to run a game.

The source files are arranged in the following manner:

```
src
|-- boardgame
| |-- Board.java                Abstract game board logic class
| |-- Move.java                 Abstract game move class
| |-- Player.java               Abstract player class
| |-- Client.java               Generic client, implements networking
| |-- Server.java               Generic server, implements networking
| |-- HumanPlayer.java          Generic remote GUI client
| |-- ServerGUI.java            Main GUI class
| \-- BoardPanel.java           Displays and gets input for a board in GUI
|
\-- omweso
  |-- CCBoardState.java         Implementation of omweso. All the game logic can
  |                             be found here, and also provides the methods that
  |                             agents will can to interact with the game.
  |
  |-- CCBoard.java              Maintains an instance of CCBoardState, and is used
  |                             by the server and GUI to interact with the board state.
  |
  |-- CCMove.java               An omweso move.
  |
  |-- CCBoardPanel.java         Displays an omweso game in the GUI.
  |
  \-- CCRandomPlayer.java       An example omweso player that plays randomly.
```

You will not need to edit or compile any of these files; they are packaged together for you in "jar/omweso.jar".

## 2 Launching the server

To start the server, type:

```
java -cp ./path/to/omweso.jar boardgame.Server [-p port] [-ng] [-q] [-t n] [-ft n] [-b class] [-k]
```

where:

- -p port sets the TCP port to listen on. (default=8123)
- -ng suppresses display of the GUI.
- -q indicates not to dump log to console.
- -t n sets the timeout to n milliseconds. (default=2000)
- -ft n sets the first move timeout to n milliseconds. (default=60000)
- -b class determines the game to run. (default=omweso.CCBoard)
- -k launch a new server every time as game is started (used to run multiple games without the GUI)

For example, assuming the current directory is the top level director of the project package, the command:

```
java -cp jar/omweso.jar boardgame.Server -p 8123 -t 300000 -b omweso.CCBoard
```

launches a server on the default TCP port, with the GUI, a timeout of 300 seconds, and running the omweso game.

The server waits for two clients to connect and does not accept any more connections unless the -k flag is passed as an argument. Closing the GUI window will not terminate the server. The server exits once the game is finished. Log files, which record all of the moves in each game, are automatically written to the logs subdirectory. The log files contain a list of all moves, as well as information about the games class and other parameters. The server also maintains a file, **outcomes.txt**, which keeps a summary of game results. At present this consists of the integer game sequence number, the name of each player, the color and name of the winning player, the number of moves, and the name of the log file.

## 3 Launching a client

The server waits for two clients to connect before starting the game.

```
java -cp ./path/to/omweso.jar boardgame.Client [playerClass [serverName [serverPort]]]
```

where:

- playerClass is the player to be run (default=omweso.CCHumanPlayer)
- serverName is the server address (default=localhost)
- serverPort is the port number (default=8123)

For example:

```
java -cp ./path/to/omweso.jar boardgame.Client omweso.CCRandomPlayer localhost 8123
```

launches the random omweso player client, connecting to a server on the local machine and default TCP port. The game starts immediately once two clients are connected. If using the server GUI, it is also possible to launch clients to be run on the same machine as the server from the "Launch" menu. This starts a regular client running in a background thread.

# 4 Launching the log-file viewer

It is possible to use the server GUI examine log files from previously played games. Moves are loaded from the file as though the game was being played on a server.

```
java -cp ./path/to/omweso.jar boardgame.ServerGUI [filename]
```

where **filename** is an optional log file to open. As an example:

```
java -cp ./path/to/omweso.jar boardgame.ServerGUI logs/game00001.log
```

allows you to review the complete sequence of moves in your first game.

# 5 Human Players

## 5.1 On the Server GUI

To play against a software player or another person, launch the server with a GUI. The status bar indicates that the server is waiting for a client connection. Select "Launch human player" from the "Launch" menu to start a user client which will obtain moves through the GUI. Launch another client from the command line or from the "Launch" menu.

## 5.2 Controls

During the seed placement phase, click on a pit to add a seed and right click on a pit (containing a non-zero number of seeds) to remove a seed. Your seed placement turn will finish as soon as you have 0 seeds left to place. During normal play, when it is your turn, left click on a pit containing more than 1 seed to begin sowing from there.

# 6 Implementing a Player

To implement a new player, extend the boardgame.Player class and, in particular, override the chooseMove method. It may also be useful to override the gameOver method to have your agent take actions upon winning or losing. To run your new player, pass the new class as the argument to the client software (as in Section 3). See the file Player.java for further instructions, and "example_submission" folder for an example. Explicit instructions for compiling and running your player are given in "example_submission/README.txt". **Please read the project specifications carefully so that you structure your code properly!**

## 6.1 Classpath

You'll also need to make your player visible to the provided code by including it in the classpath. You can do this replacing every occurrence of "-cp ./path/to/omweso.jar" with:

```
-cp ./path/to/omweso.jar:/path/to/your/class/files/sXXXXXXXXX
```

### 6.1.1 Windows Users

Please note that the classpath syntax is slightly different on the Windows command line. Instead of the separator :, use ;. Also make sure the path syntax itself is in Windows format.

### 6.2 Eclipse Users

You can link to the jar using

`Build Path --> add External Archives`

Please make sure your submission format is correct before submitting.