Lab 1: 15-111 – Game of Tic Tac Toe

Due Date: Saturday May 24th, 2008 at 11:59 pm

Assignment
Write a program to simulate a game of Tic Tac Toe.

Concepts: 2D Arrays, interactive I/O, String Parsing

Objectives:

1. Understand multi-class programs
2. Understand random number generators
3. Use of 2D arrays
4. interactive I/O
5. String Processing

Description: Many computer games, such as strategy games, simulation games etc uses game boards that can be implemented as two dimensional arrays. In Java (and in many other languages) 2D arrays can be defined as an array of 1D arrays. For example a 2D matrix of characters (with dimensions 8x10) can be defined as follows.

```
char[][]  Board = new char[8][10]
```

We can think of Board as a table with 8 rows (indexed from 0..7) and 10 columns(indexed from 0..9)

Tic Tac Toe is played on a 3x3 board with two players alternating between X's and O's, alternatively placing their respective marks on the board. First player succeeding in getting three of his marks in a row, column or diagonally wins the game.

How do we design the classes to implement a tictactoe game? Let us think of the objects in a tictactoe game environment. We can think of four types of objects, tictactoe board, human player, dumb (or random) player and smart player. In this program we will only implement human player and dumb player. As an extra credit assignment you are encouraged to write a smart player that can analyze the board and make the best move. However that is not necessary. Let us begin with the following classes and methods.

Class	Methods
TicTacToe	clearBoard( ) isWin(int), winner(), printBoard( ), putMark(int, int)
Driver	Main( )
HumanPlayer	makeMove(TicTacToe B)
ComputerPlayer (extra credit only)	makeMove(TicTacToe B), findMove(TicTacToe B)
DumbPlayer	makeMove(TicTacToe B)

The description of each method can be found in the source code. Your assignment is to complete the incomplete methods from TicTacToe, HumanPlayer and DumbPlayer classes.

Testing your program

You may use any kind of IDE such as eclipse to test your program. However, you need to make sure all programs run under Andrew linux. You should also check your program by creating different scenarios.

Human vs Human

First define, two human players and play them against each other. You can do this by initializing two HumanPlayers as follows.

HumanPlayer Player1 = new HumanPlayer( );
HumanPlayer Player2 = new HumanPlayer( );

The program should prompt each player to enter position as a pair of "valid" integers. For example, two players taking turns can be shown as follows:

> Player X : 0 1
> Player O : 1 1

Game should continue until one player wins or game is a draw. The program should correctly display the end of the game.

For Example:

- **The Player X wins**
- **Game is a draw**

Human vs Dumb

Initialize a human player and a dumb player as follows.

HumanPlayer Player1 = new HumanPlayer( );
DumbPlayer Player2 = new DumbPlayer( );

DumbPlayer will always make a *random* move. Therefore, human player should be able to usually beat the dumb player.

Dumb vs Dumb

First define, two dumb players and play them against each other. You can do this by initializing two DumbPlayers as follows.

DumbPlayer Player1 = new DumbPlayer( );
DumbPlayer Player2 = new DumbPlayer( );

It would be interesting to see which dumb player wins and to see if starting dumb player has any advantage over the other one.

*Human vs Computer (**extra credit only**)*
For 10 extra points, you can complete the computerPlayer class. A computerPlayer in most cases must be able to defeat the dumbplayer. ComputerPlayer must be smart, analyzing the entire board to make a move. We suggest that at this point you use heuristics to find the best move. Later when you take courses such as 15-211, you will be able to learn techniques to analyze the game tree and make a good move. To test the ComputerPlayer against a dumbPlayer you can do
DumbPlayer Player1 = new DumbPlayer( );
ComputerPlayer Player2 = new ComputerPlayer( );

Getting Started

Download the lab1.zip file. Save the zip file to your desktop and unzip the files. Open the directory lab1. You should see the following files:

- **`Driver.java`** (the driver file)
- **`TicTacToe.java`**
- **`DumbPlayer.java`**
- **`HumanPlayer.java`**
- **`ComputerPlayer.java`** (extra credit only)
- `smartPlayer.class` // provided for testing your computerPlayer
- a copy of the lab description

Most classes already have method headers. Your job is to fill them in with the appropriate code. DO NOT introduce any new **public** methods to given classes. You are allowed to define any new **private** methods to any of the classes provided. The driver program is **complete**. Do not change this unless when you are testing for various player combinations (eg: human vs human, human vs computer etc..)

Compiling and Running your code

You must be able to compile and run your program from command line under Andrew linux. If you running on windows on command line, then you need to set the classPath so that javac and java commands are recognized from your working folder. See Blackboard/External links to find out how to set your classPath in windows. You can also use Eclipse to develop, debug and run your program. Your TA will show you how to install eclipse on your computer. However, the final zip folder **must run** under Andrew linux.

The following commands are useful

> javac *.java // compiles all source files

> java Driver // runs the program

Handing in your Solution

Your solutions should be in the form of .zip files. When we grade your solution we will unzip the folder and test the code.

Your teaching assistants will show you how to zip up the files for submission.

All labs are submitted to afs. The folder to submit your program is

/afs/Andrew.cmu.edu/course/15/111-M08/handin/lab1/yourid

Grading Criteria

The following grading criterion is strictly enforced.

1. A program that does not compile - 0 points
2. A program that is 24 hours later – max of 50% of the grade
3. A program that is more than 48 hours late – 0 points

Grading Programming Components

1. TicTacToe class – 40 points
2. HumanPlayer class – 20 points
3. DumbPlayer class – 20 points
4. Driver class – 10 points
5. Style points – 10 points (style points are based on indentation, proper use of variable names, structure of your program etc. Your TA can provide more guidance on this. Please ask)