```
ninarow-war - ex3
------------
Name: Dor Azouri
ID: 204034300
------------

Instructions
============
1.    Deploy to tomcat server (should work on any version, I used version
9.0.12)
2.    Browse with Chrome to URL:
      http://localhost:8080/index.html
3.    There are 3 modules in this project (including the main ninarow-
war), each with a corresponding JAR/WAR artifact:
      ninarow-war -                main, depends on the other 2
                                         handles the interaction with the
user through the UI controls, using servlets.
                                         Defines the web appearance.
                                         Depends on both GameEngine and
XmlLoader, and the external gson jar.
      GameEngine -                 the core game logic and classes needed
      XmlLoader -                  handles the loading of the config XML
into java variables
4.    Spectator bonus is not implemented, but an option in rooms page is
added to see a game's board as it plays ("View Board" button on top).

Classes
============
servlets.LoginServlet:
      Serves the login requests at "/login"
servlets.RoomsServlet:
      Serves the rooms page's requests at "/rooms":
            Displays users list
            Displays romms list
            Handles Enter Room
            Handles Logout
servlets.GameServlet:
      Serves the game's requests at "/game":
            Displays game details including users list
            Displays board
            Handles client turns
            Handles Leave Room
            Handles Reset Game when game ends
            Handles messages from server
-----------
web.script.boardPage.js:
      The client logic, for both the human and computer players. Handles
all events of the game using requests to server.
      Pulls information at interval.
web.script.rooms.js:
      The logic for the rooms page. Handles all possible actions of that
page, using requests to corresponding servlet.
      Pulls information at interval.
web.script.common.js:
      Common shared functions
index html and jsp:
      The login form, requests go to the LoginServlet
web.css.*:
      Style files, including external bootstrap, mdb, login-style...
-----------
```

GameEngine.Game:
The main logic of the game. Holds a board object as well as the players list, and delegates requests from the UI to the logic and calculations that are mainly done in the Board class.

GameEngine.Board:
Describes the game board, holds its content and performs the in-game calculations for the turns made and for deciding a winner.

GameEngine.Player:
An interface that defines a player in the game. It is maintained to be non-UI dependant: it was introduced in Exercise 1 and re-used here. The interface is needed because we expected different kind of players that share same functionality from the main game perspective.

GameEngine.PlayerCommon:
An abstract partial implementation of the Player interface. Defines common implementation used by all other deriving Player classes.

GameEngine.PlayerWeb:
Concrete Player implementation for a human player through the WEB UI. Used a stub class, real turn logic is implemented in client-side js files, and servlets.

GameEngine.PlayerComputer:
Concrete Player implementation for a Computer player. Decides on turns in a naive way.

GameEngine.*Info classes:
This group of classes are defining simple structs that describe the different classes, and are used for the client-server JSON communication.

GameEngine.RoomsManager:
A class for maintaining the rooms of the game. Corresponds to the rooms page.

XmlLoader.XmlLoader:
Defines one important function - that loads an XML file into the concrete parameters needed for the game related classes (Game, Board, Players...)

Assumptions

1. All assumptions described in the exercise
2. Computer cannot be the first to enter a room. A computer can only enter after at least one human is in the room (mainly to avoid a game with computer players only, that should not be supported)