ninarow-fx - ex2

Name: Dor Azouri
ID: 204034300

Instructions
============

1. Double click the given batch file.
 The batch file executes the main module's jar: 'ninarow-fx.jar'.
2. There are 3 modules in this project (including the main ninarow-fx), each with a corresponding JAR artifact:
 ninarow-fx -   main, depends on the other 3
      handles the interaction with the user through the UI controls.
      Depends on both GameEngine and XmlLoader
 GameEngine -   the core game logic and classes needed
 XmlLoader -   handles the loading of the config XML into java variables
3. Running one of the non-main JARs will only print a stub text of the module name

Classes
============

ninarow-fx.Controller:
 The sole controller for the UI fxml.
 It renders the board as well as defines the buttons behavior.
 The initial rendering dynamically renders the bottom buttons used for the Popout game type.
 Responsible for the game flow as wells, through calls to the exported functions of the Game class.
ninarow-fx.CustomEvent:
 An abstract Event class used for the user click events.
ninarow-fx.UserTurnClickEvent:
 The concrete class representing a user click event, that defines the turn the human player has made.
ninarow-fx.UserTurnClickEventHandler:
 An abstract Event handler class used in the controller to anonymously implement the behaviour of user clicks.

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.PlayerFX:
 Concrete Player implementation for a human player through the FX UI. The turns logic is just a stub that's called from events handled in the Controller.
GameEngine.PlayerComputer:
 Concrete Player implementation for a Computer player. Decides on turns in a naive way.

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