Code Review Template

This template is to be completed and submitted by the *Reviewer*.

Names of the Reviewer: Parker Link (30045201)

Name of the developer being reviewed: William Ledingham (https://github.com/William-Ledingham/ENSF409_Lab4/tree/master/Ex3_TicTacToe/src)

Category	Comments/questions about of the reviewing group about the design documents	Responses by the developer (if any)
Spelling Mistakes	• No observable spelling mistakes were present.	Nice.
Naming Issues	• The method name testForBlocking of BlockingPlayer.java is not inherently clear what it is. • The methods addHyphens/addSpaces of Board.java should be called print___. Inconsistent naming between displayColumnHeaders and addHyphens. • The create_player() method in Game.java uses an underscore instead of camelCase.	I agree with those inconsistencies and changing them would make things clearer.
SOLID Principle Violations	• Single Responsibility: All classes effectively follow a single responsibility. Better implementation of this could be employed by following the refactoring notes below, though. • Open/Close Principle: Not directly applicable here, but the general guidelines are followed by allowing configurability with the constants defined in Constants.java. • Interface Segregation: The one interface used (Player.java) was implemented effectively, as it allows the developer to only implement the methods they	Good comments.

	need. By following the refactoring notes, further minor improvements can be made. • Dependency Inversion Principle: The higher-level player does not depend on lower-level modules, but rather depends on its children.	
Lack of documentation	• The checkWinner method of Board.java contains no inline comments. • Throughout all the code, very few inline comments (//) are used, making the intention of various sections quite difficult.	I agree Inline comments should be added for clarity in a few sections
Consideration of "Overly-Complicated" code	• The checkWinner method of Board.java could be implemented in a more clear way, potentially breaking its functionality into subroutines/methods. • The create_player method of Game.java is quite long, and could be broken into separate methods, with more methods used for human output. • Having a SPACE_CHAR constant is highly unnecessary, considering it wasn't even used in locations such as the addSpaces() method of Board.java. • Excellent use of javadocs overall, especially for member variables/attributes/fields.	I agree those sections can be simplified and broken into different methods.
Refactoring Suggestions (Avoid Code Reuse)	• testForWinning() of SmartPlayer.java contains redundant code from checkWinner of Board.java and testForBlocking of BlockingPlayer.java. Better inline documentation would help making moving this code into some shared methods, possibly placed in the Player superclass. • RandomGenerator class should be developed as a static math library-style, instead of requiring object instantiation.	I agree there does appear to be redundant code. It should be refactored with better documentation. That is a good point, RandomGenerator would make more sense as static.

My reflection about Will's review of my code (not regarding what's in the table above).

- While my spelling and naming conventions were quite good, I need to work on overall documentation quality.
- While most javadocs were done well, more javadocs can be written, specifically for member variables/attributes/fields within classes.
- Overall, I can employ the SOLID principle of single-responsibility classes better, by ensuring that each class, and each method within each class, handles only a single responsibility.