Code Review

# Instructions

1. Enter your team letter and name: \_\_2C\_\_\_
2. Pick a feature branch and create a pull request (PR).
3. Enter the PR URL:   
   https://github.com/RIT-SWEN-261-04/team-project-s02c/pull/15
4. As a team, perform a review of the pull request using the GitHub UI.
5. Using the checklist below, review the code and document suggested changes in this document.
6. Upload the final Word file to the *Code Review* dropbox in the **Exercises** category.

# Review Checklist

This section provides a checklist of things to check during the review.

## Coding Practices

* Code to abstractions
* Declare type parameters on generics
* Use loops appropriately
* Declare local variables (and parameters) **final** where appropriate
* Keep methods small
* Write small, cohesive classes
* Use logging appropriately
* Identify violations in the [Don't Repeat Yourself principle](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself)
* Use lambdas and streams appropriately

## Code Communication

* Use a consistent coding style
* Use meaningful names
* Provide code documentation (javadocs, for example)

## Design Practices

* Adherence to architectural tiers
  + Component has behavior aligned with the responsibilities for the tier it is in; eg, no business logic in Controller code
  + Adherence to use of application wiring principles:
    - Use shared service objects and pass-around using dependency injections
    - Keep application wiring code in a few, isolated, cohesive configuration components
* Adherence to core OO principles
  + Encapsulation
  + Information hiding
  + Appropriate use of inheritance
  + Appropriate use of polymorphism
* Adherence to OO design principles
  + SOLID
  + GRASP
  + Law of Demeter

## Testing Practices

* Test scenarios are clear
* Test methods are independent
* Appropriate use of assertions
* Appropriate use of mock objects
* Test code follows professional coding and design principles

# Code Review

Use the following table to document your review findings. Severity records how serious the infraction is: high (H), medium (M) or low (L).

|  |  |  |  |
| --- | --- | --- | --- |
| **Class name** | **Line #** | **Severity** | **Brief description** |
| Board | 35 | L | In order to make presentations seamless, the configurations of the BoardView should be set up in special spaces depending on the usernames that we decide to assign to test or demonstrate a specific board view. (Setup for multiple jumps, kinging pieces, etc). |
| Board | 42 - 64 | L | In order to appeal to the code design guidelines, the code should be refactored. |
| BoardView | 120 | L | The comments should indicate special cases for the addPieces() method on the Board constructor. |
| BoardView | 123 - 234 | M | The methods in the area are mostly repeating code. This can be solved by creating a switch statement to decrease repetition of code. Otherwise this would violate the “Don’t repeat yourself Principle.” |
| PostSubmitTurn Route | 107 | H | With the current implementation, the King A Piece story is not taking into account the implementation used for the multiple jump story. This can lead to a violation of the intended game logic of ending the turn once the piece has turned into a king piece. |
| PostValidateMoveRoute | 71 | L | A print statement that was used for debugging purposes was kept in. |