Code Review

# Instructions

1. 2181-swen-261-10-e-Echo10
2. https://github.com/RIT-SWEN-261-10/team-project-2181-swen-261-10-e-echo10/pull/9

# Code Review

|  |  |  |  |
| --- | --- | --- | --- |
| **Class name** | **Line #** | **Severity** | **Brief description** |
| Board | 28, 93, 130,... | H | Consistent lack of code documentation or explanation of complex flows of logic |
| Board | 153 | M | Replicated code that could be consolidated |
| Board | 163 | L | Helper methods could be made static |
| Board | 253 | L | Nesting of functions is difficult to read |
| BoardTest | 190 | M | Many of the tests duplicate the logic behind the test |
| BoardTest | 296 | M | Unit tests seem to be testing for specific scenarios and edge cases, but miss the general operation of the UUT |
| Game | 28, 38, 56,... | H | Consistent lack of code documentation or explanation of complex flows of logic |

# Review Checklist

## 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