Code Review

# Instructions

1. Enter your team letter and name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Pick a feature branch and create a pull request (PR).
3. Enter the PR URL:
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
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 (Java and TypeScript)

* Code to abstractions
* Use loops appropriately
* 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)

## Code Communication (Java and TypeScript)

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

## Design Practices (Java and TypeScript)

* Adherence to architectural tiers
	+ Component has behavior aligned with the responsibilities for the tier it is in
	+ 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
* Adherence to OO design principles
	+ SOLID
	+ GRASP
	+ Law of Demeter

## Testing Practices (Java only)

* Test scenarios are clear
* Test methods are independent
* Appropriate use of assertions
* Appropriate use of mock objects (where appropriate)
* 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).

The first line is a sample to show you want we are looking for.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class name** | **Line #** | **Severity** | **Brief description** |
| MyClass | 47 | H | Accessor methods expose an internal List. Remove these and create new methods to find, add and remove elements (as needed). |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |