Source Code Review Sheet

Phase 1: The Light Pass

General

- Adding comments: Be polite and constructive
- Correct use of code formatter
- Line indentation char and line breaks
- Only required files are changed
- Verify comments in source
- Check static code analysis results
- Disagreement between code and specification
- Max usage of static compiler checking
- Optimistic and undefensive programming

Unclear Or Messy

- Verify correct and meaningful naming
- Magic numbers and values
- Variables used for more than one purpose
- Design principles: DRY, SOLID, GRASP, YAGNI
- Variable, method and class scope

- Method signature
- Packing too much into one line
- Long Method
- Cyclomatic complexity

Error Handling

- Avoid empty catch blogs
- Error handler over-catches exceptions and aborts current flow or application
- Error handler is not implemented e.g. contains TODO, FIXME

Java

- Review class imports
- Missuse of == and equals()
- Misuse of Arrays, List or Set
- Object contract errors (equals and hashCode)
- Exposure of immutable data types

Git Commit Message

- All details are included
- Describe what and why it has changed

Phase 2: The Contextual Pass

Code Structure

- Understandability of written changes
- Logical errors
- Wrong usage of implementation patterns
- Alternative implementations that increase simplicity, readability or maintainability
- Check edge cases in functions
- Better approach to use a framework, library or class
- Look for omissions: Shouldn't this component also do X?

External Systems

- Reduce amount of calls / Optimize calls to external systems

Tests

- Existing tests still pass
- Test coverage of changed lines and critical path
- Enhancements to newly added tests

Code-Review Commenting

- Add positive comments for good code: Unusually ellegant solution, creative solution, great design,...

Finalizing

- Sign of the pull request

Additional Checklists

Internet Scale Services Checklist

References

Code Review

Code Briefing: What does it mean when code is “easy to reason about”?

Do Not Allow Bad Smells In Your Code

Writing Great Git Commit Messages; A Revision

Giving better code reviews

Coding like Shakespeare: practical function naming conventions

Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems