Source Code Review Sheet

Phase 1: The Light Pass

General

- Adding comments: Be polite and constructive
- Code formatter has been used
- Line indentation char and line breaks
- Only required files are changed
- Disagreement between code and specification
- Comments in source code are included
- Documentation of the implementation is created or updated
- Does not contain any unimplemented areas like //TODO or //FIXME
- Optimistic or undefensive programming

Unclear Or Messy

- Verify correct and meaningful naming
- Magic numbers and values
- Variables used for more than one purpose
- Minimized variable, method and class scopes
- 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
- Wrong use of == and equals()
- Wrong use of data types, like Arrays, List or Set
- Object contract errors (e.g. 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
- No logical errors
- Max usage of static compiler checking
- Does not violate architecture guidelines
- Does not violate design principles
- Does not violate implementation patterns
- Are there any alternative implementations that increase simplicity, readability or maintainability
- Check edge cases in functions
- Any better approach to use a framework, library or class exists?
- Look for omissions: Shouldn't this component also do X?

External Systems

- Reduce amount of calls / Optimize calls to external systems

Tests

- Unit-tests and End2End Tests are added
- Test coverage of changed lines and critical path
- Enhancements to newly added tests

Code-Review Commenting

- Add positive comments for good code e.g. unusually elegant solution, creative solution, great design

Finalizing

- Sign of the pull request

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

Software Architecture and Design

YouTube: Core Design Principles for Software Developers by Venkat Subramaniam