CodeStyle

General Code Conventions

- The Cassandra project follows Sun's Java coding conventions (http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html) with an important exception: { and } are always placed on a new line

Exception handling

- Never ever write `catch (...) {}` or {{catch (...) { logger.error() }}} merely to satisfy Java's compile-time exception checking. Always propagate the exception up or throw RuntimeException (or, if it "can't happen," AssertionError). This makes the exceptions visible to automated tests.
- Avoid propagating up checked exceptions that no caller handles. Rethrow as RuntimeException (or IOError, if that is more applicable)
- Similarly, `logger.warn()` is often a cop-out: is this an error or not? If it is don't hide it behind a warn; if it isn't, no need for the warning.
- If you genuinely know an exception indicates an expected condition, it's okay to ignore it BUT this must be explicitly explained in a comment.

Boilerplate

- Avoid redundant @Override annotations when implementing abstract or interface methods
- Do not implement equals or hashcode methods unless they are actually needed.
- Prefer public final fields to private fields with getters. (But prefer encapsulating behavior in "real" methods to either.)
- Prefer requiring initialization in the constructor to setters.
- avoid redundant "this" references to member fields or methods
- Do not extract interfaces (or abstract classes) unless you actually need multiple implementations of it
- Always include braces for nested levels of conditionals and loops. Only avoid braces for single level.

Multiline statements

- Try to keep lines under 120 characters, but use good judgement – it's better to exceed 120 by a little, than split a line that has no natural splitting points.
- When splitting inside a method call, use one line per parameter and align them, like this:

```
SSTableWriter writer = new SSTableWriter(cfs.getTempSSTablePath(),
                                         columnFamilies.size(),
                                         StorageService.getPartitioner());
```

- When splitting a ternary, use one line per clause, carry the operator, and align like this:

```
var = bar == null
    ? doFoo()
    : doBar();
```

Whitespace

- Please make sure to use 4 spaces instead of the tab character for all your indentation
- Many lines in many files have a bunch of trailing whitespace... Please either clean these up in a separate patch, or leave them alone, so that reviewers now and anyone reading code history later don't have to pay attention to whitespace diffs.

imports

Please observe the following order for your imports:

- java

- [blank line]

- com.google.common
- org.apache.commons
- org.junit
- org.slf4j

- [blank line]

- everything else alphabetically

format files for IDEs

- IntelliJ: intellij-codestyle.jar
- IntelliJ 13: gist for IntelliJ 13 (this is a work in progress, still working on javadoc, ternary style, line continuations, etc)
- Eclipse (https://github.com/tjake/cassandra-style-eclipse)

`https://c.statcounter.com/9397521/0/fe557aad/1/|stats`