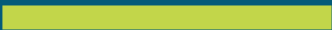


# LEAN CODING STANDARDS

ONE-PAGE TEMPLATE

**“You do not rise to the level of your goals. You fall to the level of your systems.”**

**– James Clear**



**No one has time for coding standards that are 100 pages long. Use this guide as a starting point to create a one-page(ish) set of rules to help you actually streamline coding across a team.**

## ONE-PAGE CODING STANDARDS

---

### Getting Started

In order to form a habit out of something, it needs to be easy enough to repeat and obvious enough that you can remember. How can you achieve that? Use the below lists as a jumping off point for rules to include. What are some ground rules?

- Start small and make it attainable. You can always add more as you go.
- Shoot for your standards to fill one 8.5" x 11" page or less.
- Decide on what to include as a team for maximum buy-in and use.
- Be concise. Only include the amount of information needed to ensure clarity.

### Sample Standards

We typically break our document down into two main sections: **coding standards** and **unit test standards**. Start with practices like those included within our Level 100 recommendations. Once those become a habit, start adding new practices.

#### LEVEL 100 PRACTICES

##### Coding Standards

- No duplication
- Include ticket number in commit message
- No commented out code
- Methods have 10 or fewer lines - including test methods.
- Methods/constructors have 3 or fewer parameters.
- Classes have 7 or fewer public members (single responsibility).
- No method calls or logic in constructors.
- No checked exceptions (Java).
- Classes are named using a noun or noun phrase.
- Methods are named using a verb or verb phrase.
- Variable and field names are pronounceable
- Only 1 level of inheritance (Abstract classes only)

##### Unit Test Standards

- No `assertNotNull()`.
- No `@Ignore` tests.
- 3 or fewer assertions per test method.
- No mocked static methods.

## ONE-PAGE CODING STANDARDS

---

### Sample Standards (continued)

#### LEVEL 200 PRACTICES

##### **Coding Standards**

- No name decorations (FooImpl.java)
- No getter/setters on interfaces
- No constant interfaces
- No public constants
- Query methods don't throw exceptions
- Query methods don't change state
- Command methods change state
- Command methods throw exceptions when state is unable to be changed
- Factory methods over constructors

##### **Unit Test Standards**

- Arrange, Act, Assert
- No uncovered lines.
- Prefer state based testing over interaction based testing (prefer stubs over mocks).

##### **What To Do Next**

- Once you have your coding standards drafted, post them up in a physical or virtual location where everyone can see them daily.
- Choose automation over documentation. If you can automate it, you don't have to remember to do it.
- Review coding standards quarterly as a team. Remove any that have become a habit and add in new rules (such as the level 200 practices noted above).

## **A LITTLE ABOUT LT**

**Lean TECHniques works alongside leaders and teams who are sick of the status quo and are ready to do things differently within their organization. Together, we'll create a company where technology becomes your competitive advantage and customers rave about the experience you offer. Status quo? Hell no.**