

# Check in criterias

A developer is not allowed to check in any code that does not fulfill all the below given criterias:

## Unit Tests

- All new features added to the system must have a unit test suite.
- The developer has to ensure that the code written has tests for everything that could possibly break (well, as long as it is practical).

## JavaDoc

- JavaDoc has to be written for **all** methods, fields and classes.
- The JavaDoc has to be thought through and should describe the purpose and responsibility of the method, field or class.
- All method parameters should be described
- All classes must have a copyright notice at the top and author tags on the format `@author <a href='mailto:dduck@codehaus.org'>Donald Duck</a>`

## Documentation

- All new features must be well documented in the *Maven XDoc* format and put in the CVS.

## Code convention

- indentation must be four spaces (no tabs)
- never use **this** (`this.<field>`) when referencing member fields, instead use prefixes (`s_` and `m_`)
- Member variables should always be prefixed with `m_`
- Static variables should always be prefixed with `s_`
- Constants should always be in uppercase and use underscores: `THIS_IS_A_CONSTANT`
- no lines should be longer than 120 characters
- wildcards are NOT allowed in package imports and the imports should be logically grouped
- invocations to static methods should always be prefixed with their class, e.g. `MyClass.myStaticMethod`
  
- grouping of fields and methods in a class should be in the following order (top down):
  - constants
  - static fields
  - member fields
  - public static methods
  - public member methods
  - protected static methods
  - protected member methods
  - etc.
  - inner classes
  
- try write *self-documenting code* by giving the classes, methods and fields descriptive names (even if it makes them long), when this is not sufficient use comments
  
- use curly braces for single statements (except for assertions):

```
// correct
if (true) {
    return true;
}
// wrong
if (true) return true;
```

- opening curly braces should **not** be on a newline:

```
if (...) {
    ...
}
```

- opening curly braces are always followed by a newline, e.g. f.e. don't write:

```
public String getName() { return name; }
```

- else, catch and finally statements should **not** start on a new line:

```
try (...) {
    ...
} catch {
    ...
} finally {
    ...
}
```

- spaces should be used as in the code sample given:

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        i += 5;
    }
}
```