Naming

Rules common to all identifiers

Identifiers use only ASCII letters and digits, and, in a small number of cases noted below, underscores. Thus each valid identifier name is matched by the regular expression \w+ .

In Google Style, special prefixes or suffixes are **not** used. For example, these names are not Google Style: name_, mName, s_name and kName.

Rules by identifier type

Package names

Package names are all lowercase, with consecutive words simply concatenated together (no underscores). For example, com.example.deepspace, not com.example.deepSpace or com.example.deep_space.

Class names

Class names are written in UpperCamelCase.

Class names are typically nouns or noun phrases. For example, Character or ImmutableList. Interface names may also be nouns or noun phrases (for example, List), but may sometimes be adjectives or adjective phrases instead (for example, Readable).

There are no specific rules or even well-established conventions for naming annotation types.

Test classes are named starting with the name of the class they are testing, and ending with Test. For example, HashTest or HashIntegrationTest.

Method names

Method names are written in lowerCamelCase.

Method names are typically verbs or verb phrases. For example, sendMessage or stop.

Underscores may appear in JUnit *test* method names to separate logical components of the name, with *each* component written in lowerCamelCase. One typical pattern is *<methodUnderTest>_<state>*, for example pop_emptyStack. There is no One Correct Way to name test methods.

Constant names

Constant names use CONSTANT_CASE: all uppercase letters, with each word separated from the next by a single underscore. But what *is* a constant, exactly?

Constants are static final fields whose contents are deeply immutable and whose methods have no detectable side effects. This includes primitives, Strings, immutable types, and immutable collections of immutable types. If any of the instance's observable state can change, it is not a constant. Merely *intending* to never mutate the object is not enough.

Non-constant field names

Non-constant field names (static or otherwise) are written in lowerCamelCase.

These names are typically nouns or noun phrases. For example, computedValues or index.

Parameter names

Parameter names are written in lowerCamelCase.

One-character parameter names in public methods should be avoided.

Local variable names

Local variable names are written in lowerCamelCase.

Even when final and immutable, local variables are not considered to be constants, and should not be styled as constants.

Type variable names

Each type variable is named in one of two styles:

- A single capital letter, optionally followed by a single numeral (such as E, T, X, T2)
- A name in the form used for classes (see Section 5.2.2, Class names), followed by the capital letter T (examples: RequestT, FooBarT).

Camel case: defined

Sometimes there is more than one reasonable way to convert an English phrase into camel case, such as when acronyms or unusual constructs like "IPv6" or "iOS" are present. To improve predictability, Google Style specifies the following (nearly) deterministic scheme.

Beginning with the prose form of the name:

1. Convert the phrase to plain ASCII and remove any apostrophes. For example, "Müller's algorithm" might become "Muellers algorithm".
2. Divide this result into words, splitting on spaces and any remaining punctuation (typically hyphens).
    - *Recommended:* if any word already has a conventional camel-case appearance in common usage, split this into its constituent parts (e.g., "AdWords" becomes "ad words"). Note that a word such as "iOS" is not really in camel case *per se*; it defies *any* convention, so this recommendation does not apply.
3. Now lowercase *everything* (including acronyms), then uppercase only the first character of:
    - ... each word, to yield *upper camel case*, or
    - ... each word except the first, to yield *lower camel case*
4. Finally, join all the words into a single identifier.