**ava – Naming Conventions**

**Java naming conventions** are sort of guidelines that application programmers are expected to follow to produce a consistent and readable code throughout the application. If teams do not follow these conventions, they may collectively write an application code that is hard to read and difficult to understand.

Java heavily uses [**Camel Case**](https://en.wikipedia.org/wiki/Camel_case) notations for naming the methods, variables etc. and [**TitleCase**](https://en.wikipedia.org/wiki/Letter_case) notations for [classes](https://howtodoinjava.com/java/basics/java-classes-objects/) and interfaces.

Let’s understand these naming conventions in detail with examples.

**1. Package naming convention**

Package names must be a group of words starting with all lowercase domain names (e.g. com, org, net, etc). Subsequent parts of the package name may be different according to an organization’s own internal naming conventions.

|  |
| --- |
| package com.howtodoinjava.webapp.controller;    package com.company.myapplication.web.controller;    package com.google.search.common; |

**2. Class naming convention**

In Java, class names generally should be **nouns**, in title-case with the first letter of each separate word capitalized. e.g.

|  |
| --- |
| public class ArrayList {}    public class Employee {}    public class Record {}    public class Identity {} |

**3. Interface naming convention**

In Java, interfaces names, generally, should be **adjectives**. Interfaces should be in titlecase with the first letter of each separate word capitalized. In same cases, interfaces can be **nouns** as well when they present a family of classes e.g. List and Map.

|  |
| --- |
| public interface Serializable {}    public interface Clonable {}    public interface Iterable {}    public interface List {} |

**4. Method naming convention**

Methods always should be **verbs**. They represent action and the method name should clearly state the action they perform. The method name can be single or 2-3 words as needed to clearly represent the action. Words should be in camel case notation.

|  |
| --- |
| public Long getId() {}    public void remove(Object o) {}    public Object update(Object o) {}    public Report getReportById(Long id) {}    public Report getReportByName(String name) {} |

**5. Variable naming convention**

All instance, static and method parameter variable names should be in camel case notation. They should be short and enough to describe their purpose. Temporary variables can be a single character e.g. the counter in the loops.

|  |
| --- |
| public Long id;    public EmployeeDao employeeDao;    private Properties properties;    for (int i = 0; i < list.size(); i++) {    } |

**6. Constant naming convention**

Java constants should be all **UPPERCASE** where words are separated by **underscore** character (“\_”). Make sure to use final modifier with constant variables.

|  |
| --- |
| public final String SECURITY\_TOKEN = "...";    public final int INITIAL\_SIZE = 16;    public final Integer MAX\_SIZE = Integer.MAX; |

**7. Generic types naming convention**

Generic type parameter names should be uppercase single letters. The letter 'T' for type is typically recommended. In JDK classes, E is used for collection elements, S is used for service loaders, and K and V are used for map keys and values.

|  |
| --- |
| public interface Map <K,V> {}    public interface List<E> extends Collection<E> {}    Iterator<E> iterator() {} |

**8. Enumeration or enum naming convention**

Similar to class constants, enumeration names should be all uppercase letters.

|  |
| --- |
| enum Direction {NORTH, EAST, SOUTH, WEST} |

**9. Annotation naming convention**

Annotation names follow title case notation. They can be adjectives, verbs, or nouns based on the requirements.

|  |
| --- |
| public @interface FunctionalInterface {}    public @interface Deprecated {}    public @interface Documented {}    public @Async Documented {    public @Test Documented { |

In this post, we discussed the **naming conventions in Java** to be followed for consistent writing of code which makes the code more readable and maintainable.

Naming conventions are probably the first [best practice](https://howtodoinjava.com/java-best-practices/) to follow while writing clean code in any programming language.