Naming conventions

Naming conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier-for example, whether it's a constant, package, or class-which can be helpful in understanding the code. **Java naming conventions** are sort of guidelines which application programmers are expected to follow to produce a consistent and readable code throughout the application. If teams do not not follow these conventions, they may collectively write an application code which is hard to read and difficult to understand.Java heavily uses **Camel Case** notations for naming the methods, variables etc. and **TitleCase** notations for classes and interfaces.

1. Packages

The prefix of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166, 1981.Subsequent components of the package name vary according to an organization's own internal naming conventions. Such conventions might specify that certain directory name components be division, department, project, machine, or login names.

Examples:
com.sun.eng
com.apple.quicktime.v2
edu.cmu.cs.bovik.cheese
package com.howtodoinjava.webapp.controller;
Package com.company.myapplication.web.controller;
package com.google.search.common;

2. Classes

   Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).

   Examples
   class Raster;
   class ImageSprite;

3. Interface

   Interface names should be capitalized like class names.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.

   Examples

```
interface RasterDelegate;
interface Storing;
```

4. Method

   Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.Methods should always be **verbs**. They represent an action and the method name should clearly state the action they perform. The method name can be a single or 2-3 words as needed to clearly represent the action. Words should be in camel case notation.

   Examples
   ```
   run();
   runFast();
   getBackground();
   ```

5. Variables

   Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed.

   Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is,designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.

   Examples
   ```
   int        i;
   char        c;
   float       myWidth;
   ```

6. Constants

   The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_"). (ANSI constants should be avoided, for ease of debugging.)

   Examples
   ```
   static final int MIN_WIDTH = 4;
   static final int MAX_WIDTH = 999;
   static final int GET_THE_CPU = 1;
   ```

7. Enumeration Names

   Enumeration names should follow the conventions of class names. The enumeration set of objects (choices) should be all uppercase letters:

```
enum Battery {CRITICAL, LOW, CHARGED, FULL}
```