# Project Guidelines

## Variable Names

Use camel case in variable names as well e.g. price, quantity, totalAmount etc.

Use all caps for constants e.g. MAX\_QUANTITY, MAX\_PRICE etc.  
  
  
**Avoid Pointless Names:**

No: abc, temp, data

**Avoid Similar Names:**

No: employee and employees

**Avoid Clutters:**

**No:** \_, m\_, o\_, simply \_

Avoid Hungarian notation:

No: bExit for boolean variable, iMax for integer variables

Avoid using non ASCII characters and words from local language:

If you have to represent a collection or plural, prefer something like listOfEmployees, bunchOfEmployees over employees.

Follow Classical Programming Convention:  
  
i and j as loop counter in for loop

E.g.: for(int i=0; i<10; i++){ // your code }

## Method Names

Start name of method from small character and follow camel case e.g. getEmployee(), getPayDate() etc.

Instead of having three different methods destroy(), kill(), or finish() at different modules, prefer one of them e.g. destroy().

Method names should start with verb e.g. get, set, do, invoke etc.

Give Meaningful Names:

Yes: getPayDate()

No: getPD()

Prefer shorter name over longer one, if it reveal intent clearly:

getPayDate() is better than retreivePaymentDate()

Prefer shorter name if and only if it reveal intent completely, otherwise choose longer and descriptive name:

getLiquidityIndicator() is better than getLInd() 

If variable name is payDate then getter method name must be getPayDate()and setter method must be setPayDate().

## Class Names

Start name of class as capital letter e.g. Employee, Student or Thread.  
  
Class name should be noun and should tell what does this class represent e.g. Employee, Thread, String etc.

## Interface Names

Interface name should describe ability or CAN DO part e.g.

[Runnable - can run](http://javarevisited.blogspot.sg/2012/01/difference-thread-vs-runnable-interface.html), Callable - can be called etc. 

## Package Name

Package name should follow standard company structure e.g. com.company.project.module.

# Client Application

Structure

Name of Files

# Explanation

Documentation allows you to transfer the why behind code. Much in the same way code comments explain the why, and not the how, documentation serves the same purpose.

If people don’t know why your project exists,

they won’t use it.

If people can’t figure out how to install your code,

they won’t use it.

If people can’t figure out how to use your code,

they won’t use it.

You only get contributions after you have put in a lot of work.

You only get contributions after you have users.

You only get contributions after you have documentation.

Technical writing is an art that doesn’t come naturally. Writing documentation will start you down the road to being a better technical writer, which is a useful skill to have as a programmer.

Documentation of Project:

What problem your project solves

Installation instructions