## Coding Standards

### General

**Use vertically aligned curly brackets**

**if** **(** x **)**

**{**

//Code here

**}**

**The else statement starts on the line after the closing brackets of the previous statement**

**if** **(** x **)**

**{**

//Code here

**}**

**else**

**{**

**}**

**Pad parenthesized expressions AND operator usage with spaces**

public void Add **(** int x**,** int y **);**

int x **=** 0**;**

int x **=** 5 **+** 5**;**

**Instead of**

public void Add **(**int x**,**int y**);**

int x**=**0**;**

int x**=**5**+**5**;**

**Use precision specification for floating point values unless there is  
an explicit need for a double.**

float f **=** 0.5f**;**

**Instead of**

float f **=** 0.5**;**

**And**

float f = 1.0f;

float f **=** 1.0f**;**

**Instead of**

float f **=** 1.f**;**

**FOR JAVA**

**Use camelCase for function names and variable names (camelCase means first letter of each word is lower case. Subsequent words start with upper case** *e.g. additionNotSubtraction()* **)**

1. **Function Names**

void foo **(** void **);**

**instead of**

void Foo **(** void **);**

1. **Variable Names**

float maxDistanceFromPlane**;**

**instead of**

float maxDistanceFromPlane;

**Use PascalCase for class names (PascalCase means the first letter is uppercase. Subsequent words also start with an uppercase)**

1. **Class Names**

class BaseObject

**instead of**

class baseObject

1. **Constants**

**Constants should be written in uppercase characters separated by underscores. You may also include digits when appropriate but NOT AS THE FIRST CHARACTER**

**Constants should**

static final int NUMBER\_OF\_NOOBS;

**instead of**

static final int numberOfNoobs;

**Use ‘**final**’ as much as possible on variables that shouldn’t be modified.**

**Place** final **keywords before the variable type.**

final int foo;

**instead of**

int final foo;

**Align code using real tabs which are equal to 4 spaces**

**Examples:**

int x**;**

int foo**;**

class idVec3

**{**

float x**;**

float y**;**

float z**;**

**}**

**Ordering of class variables and methods should be as follows:**

1. public variables
2. public methods
3. protected variables
4. protected methods
5. private variables
6. private methods

**This allows the public interface to be easily found at the beginning**

**of the class****.**

### 

### Comments

Write comment blocks formatted with Javadoc tags that gives a meaningful description of the class.

**The following example is a Class which adds two numbers together**

/\*\*

\* <h1>Add Two Numbers!</h1>

\* The AddNum program implements an application that

\* simply adds two given integer numbers and Prints

\* the output on the screen.

\* <p>

\* <b>Note:</b> Giving proper comments in your program makes it more

\* user friendly and it is assumed as a high quality code.

\*

\* @author Zara Ali

\* @version 1.0

\* @since 2014-03-31

\*/

/\*\*

\* This method is used to add two integers. This is

\* a the simplest form of a class method, just to

\* show the usage of various javadoc Tags.

\* @param numA This is the first paramter to addNum method

\* @param numB This is the second parameter to addNum method

\* @return int This returns sum of numA and numB.

\*/

public int addNum**(**int numA**,** int numB**)**

**{**

**return** numA **+** numB**;**

**}**

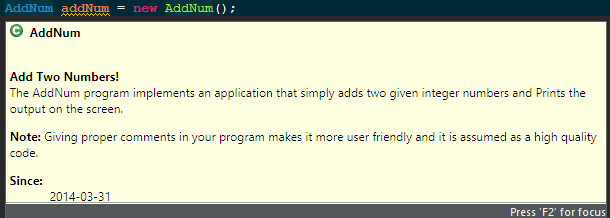
**Refer to this link for more information on Javadocs**

<https://www.tutorialspoint.com/java/java_documentation.htm>

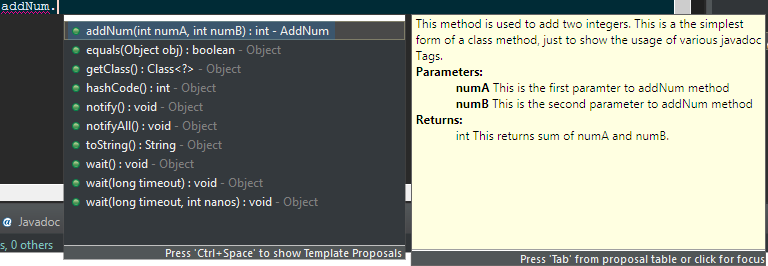
**EXAMPLE OF JAVADOCS**

When you write properly formatted Javadoc comments, this is what shows up in the IDE.

**Class Javadoc**



**Function Javadoc**



**AVOID obvious comments :**

**if** **(** a **==** 5 **)** // if a equals 5 ( DON’T DO THIS, I’LL DESTROY YOU)

**Align comments in consecutive lines**

final int maxItems **=** 10**;** // maximum number of items

final int maxHP **=** 100**;** // maximum HP of character