**File Names**

Each source file contains one public class . the file name has to be the name of that class. By convention, the source file uses a .java filename extension.

Example

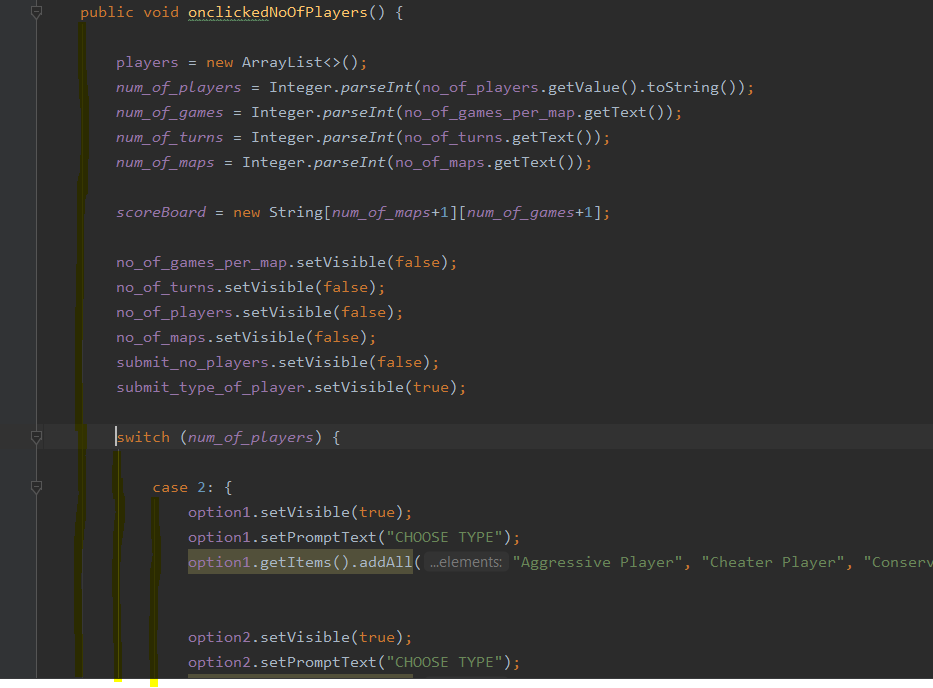
Public Player

The file stored should also be Player.java

It should be capital in both cases as java is case sensitive it will throw you an error if it finds the class name and file number are different.

**Indentation**

Tabs are used as the building blocks for indentation. They can be used and set after every 4 spaces but in some cases they can also be used for 8 spaces.



**Line Lengths**

The line lengths greater than 80 need to be avoided as they weren’t handled well by the Terminals.

**Wrapping Lines**

on based on the Comma and operation breaking the expression when it does not fit to the single line

longName1 = longName2 \* (longName3 + longName4 - longName5)

+ 4 \* longname6; // PREFER

longName1 = longName2 \* (longName3 + longName4

- longName5) + 4 \* longname6; // AVOID

**Comments**

The java comments are statements that are not executed by the compiler and interpreter. The comments in general provide us details of variables, methods, loops, etc. more over there are few comments which are used to hide the code for future purposes. Avoid the usage of comments in cases like where the comments are similar to code as it increases the redundancy and won’t be helpful

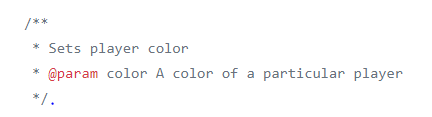
**Single line comments**

/\* …….\*/.

The single line comments start at the same line and end at the same line.

**Block Comments**

Typically this kind of comments are used in the beginning of the program, method etc. in some cases these are used inside the specific methods too the block comments should begin with /\* and end with \*/



**Trailing Comments**

The comments should begin in the new line and comments and the code are generally in different lines but there are few cases where the comments are on the same line as code as represented in the examples below.

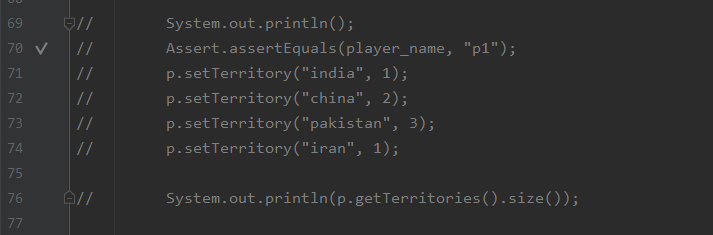
IF(){

A=B /\* COMMENT \*/

}

**End of Line Comments**

This Comment // is not used as often unlike the trailing comments and this is particularly used in the places to comment out the code for multiple sections.



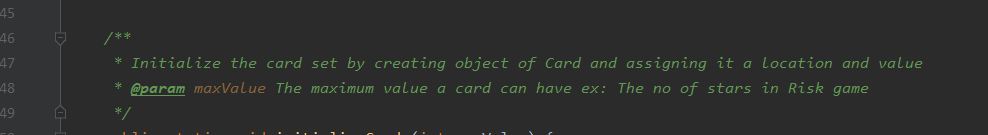
**Implementation comments**

These are used inside the methods for giving details regarding what a particular variable is doing .



**Documentation comments**

The documentation comment is used to create documentation API. To create documentation API, you need to use Javadoc tool. java doc comments are any multi-line comments which are placed before class ,field and method declaration. they began with /\*\* which includes special tags which describes characteristics like method parameters or return values.HTMl files generated describes fields and methods of class



**Declaration**

**Number per line**

One declaration per line is recommended since it encourages commenting.

In other words,

int big;

int small;

The above examples are preferred over this int big, small;

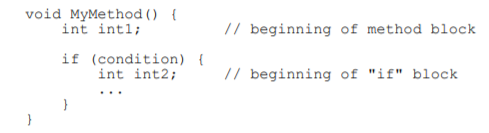
In absolutely no case should variables and functions be declared on the same line.

Example: long dbaddr, getDbaddr();

Moreover Don’t put different types on the same line.

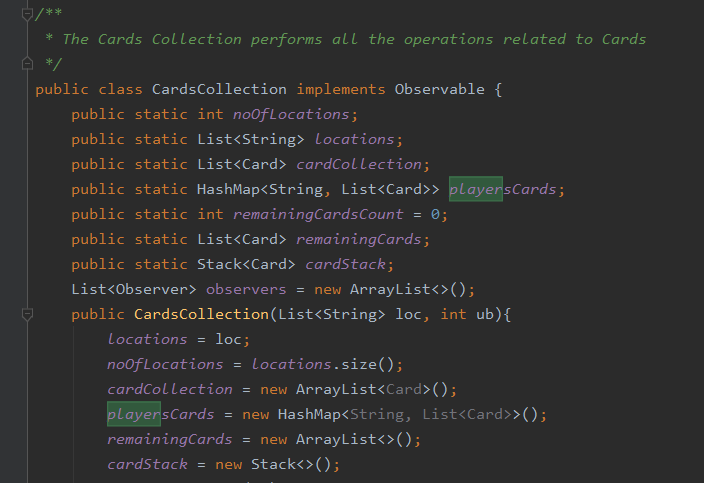
Int cat,dog[];

Put the declarations at the start of block and avoid declaring them at the



**class, interface and method declaration**

class names should be nouns, in mixed case with first letter of each internal word capitalized. try to keep your class names simple and descriptive. avoid acronyms and abbreviations.

Methods should be verbs, in mixed case with first letter lowercase, and first letter of second word capitalised.

**Statement**

**Simple statements**

They should have a single line

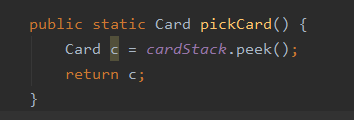
argv++; argc--; // AVOIDED!

**Compound Statements**

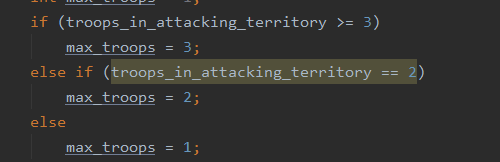
Compound statements are statements that contain lists of statements enclosed in braces . The enclosed statements should be indented one more level than the compound statement. • The opening brace should be at the end of the line that begins the compound statement; the closing brace should begin a line and be indented to the beginning of the compound statement.

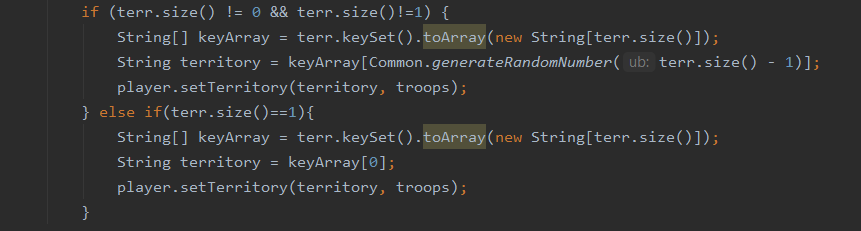
**Return Statements**

while writing a return statement the important point is to declare it with the return type value.



**If else statement**





**Exceptions**

These are done as per the standard format



**Naming Conventions**

All constants ,global variables go into Common.class available in java folder.Name them in all Capital letters ,words separated by \_..All functions/method names are in camel case.All variable names should be in small case letters separated by underscore

**Blank Lines**

Blank lines indicate the readability of the code they should be given under following circumstances a. between sections and source code.

b. between class and interface definition.

c. between methods.

d. between logical sections in the method.

**Blank Spaces**

Blank spaces should be only used under following conditions

a. the expressions in for statement should be separated by spaces.

b. the blank spaces can be given in print statements in order to improve the readability.

|  |  |  |
| --- | --- | --- |
| Identifier | Naming Conventions | Example |
| Classes  Interfaces  Methods  Variables  Constants | Classes names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep 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).  Interface names should be capitalized like Class names.  Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.  Except for variables, all instances, class, and class constants are in mixed case with a lower-case first letter. Internal words start with capital letters.  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 of its use. One-character variable names should be avoided except for temporary “throwaway” variables. Common names for temporary variables are i, j, k.  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.) | class CheaterPlayer;  class Tournament;  interface PlayerBehaviour  takeInputMap  String [][] scoreboard; |