**Code Conventions**

1. **Introduction**
   1. Why is it needed?

* Code conventions are important to programmers due to varied reasons:
* Major portion of lifetime cost of the project goes into maintenance
* Code conventions improves the readability of the software, leading to an improved understandability
* If we ship the code as a product, it must be well packaged and clean as any other product created.
  1. Acknowledgments

This document mentions and highlights the major coding standards used in the Project development. Major contributions are from Pushpa, Jasmeet, Mahgol and Himani.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **File Names**

This section lists commonly used suffixes and names

* 1. Suffixes

Java uses the following file Suffixes:

File Type Suffix

Java source .java

Java bytecode .class

* 1. Names
* Commonly used file names are:
* AttackController.java
* TournamentController.java
* PlayerStrategy.java
* AggressivePlayerStrategy.java
* CheaterPlayerStrategy.java
* TournamentView.java
* Graph.java

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **File Organisation**

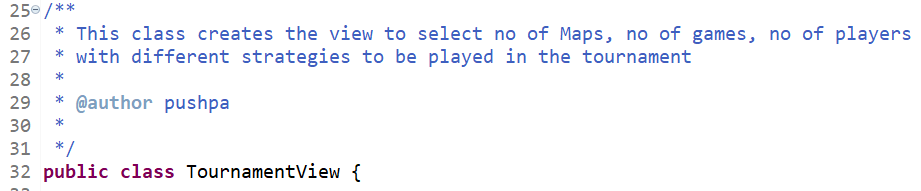
File is mainly composed of sections separated by blank lines or comments. Large size files should be avoided.

* 1. Java Source File

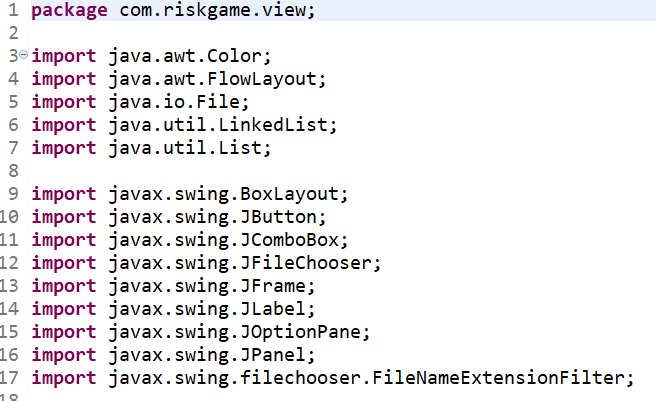
Each Java source File has a public class or interface. If a private class/interface is linked to a public class, they can be combined into the same source file as the public class. The public class should be the first class or interface in the file.

List of objects Java source file consists:

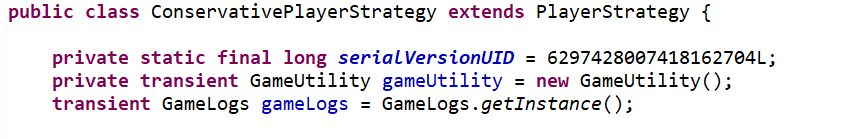
1. Beginning comments



1. Package and Import Statements



1. Class and Interface declarations



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Indentation**

Four spaces should be used as the unit of Indentation. Tabs can also used and set after exactly 8 spaces.



* 1. Line Length

We have avoided lines longer than 70 words so that thy can be handled well by terminals

* 1. Wrapping Lines

Breaking the expression when it does not fit into a single line, on the basis of comma, operator etc.

function(longExpression1, longExpression2, longExpression3,

longExpression4, longExpression5);

var = function1(longExpression1,

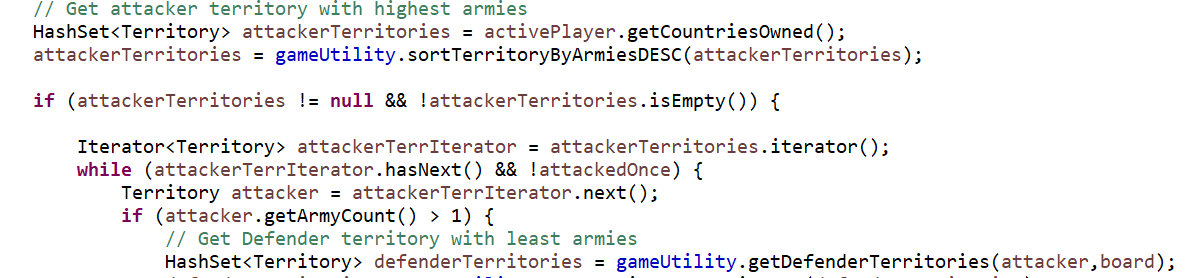
function2(longExpression2, longExpression3));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

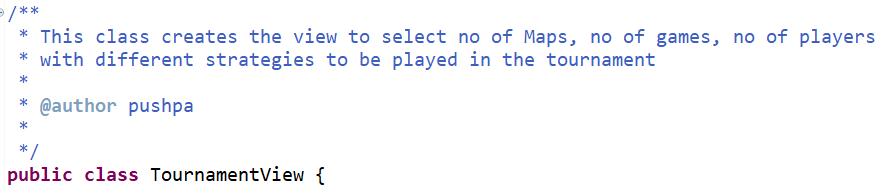
1. **Comments**

We have used both Implementation comments as well as Documentation comments (Java Doc Comments for Documentation)

1. Implementation Comments:



1. Documentation Comments



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Declarations**
   1. Number per Line

One declaration per line is recommended since it encourages commenting. In other words,

int level; // indentation level

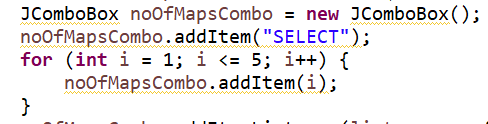
int size; // size of table

is preferred over

int level, size;

* 1. Initialization

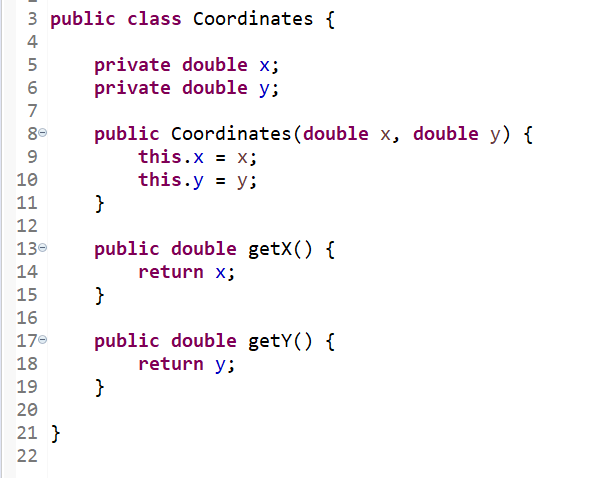
We have tried to initialize the local variables where they are declared.



* 1. Class and Interface Declarations

Below mentioned points are kept in mind for Class or interface declarations:

* No space between method name and parentheses
* Open brace in the same line as the declaration statement
* Closing brace is indented in the same place as the opening brace, except when it is a null statement where we close immediately after the “{”



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

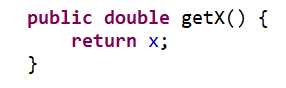
1. **Statement**
   1. Simple Statements

Each line contains at most one statement:

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

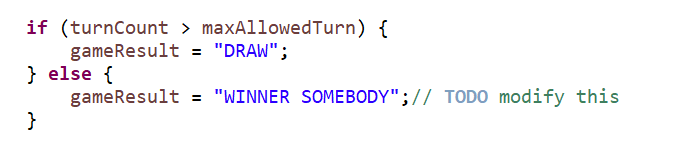
* 1. return statements

return statement with a value should not have parentheses unless obvious



* 1. if, if-else, if-else-if statements

if statements always use braces, we have avoided error-prone form.



* 1. Try-catch statements: This is used judiciously as per the standard format.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

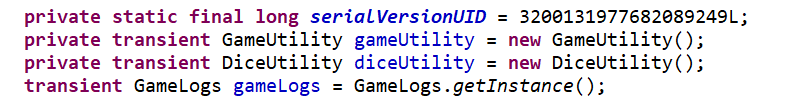
1. **Naming Conventions**

Naming conventions make program understandable as it becomes easy to read. It also helps us identify whether it is a constant, package, or class.

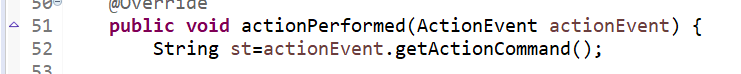
* For Constants, we have used upper case letters, especially for global constants.
* For Classes, Interfaces, User Defined types, we have the first letter in upper case.



* For variables, we have used camel case format.



* For file names, we have used lower characters only.
* For local variables, we have used lower case entirely.



We can summarize the naming conventions as below:

|  |  |  |
| --- | --- | --- |
| **Identifier Type** | **Rules for Naming** | **Examples** |
| 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, m, and n for integers; c, d, and e for characters.  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 TournamentView;  class AggressivePlayerStrategy;  createTournamentParentPanel();  runReinforcePhase(Player,Board);  int noOfMap;  Boolean isSetUpPhase; |