D1 Coding Style

Robot Game coding Style

**Source File Characters:**

**File name**

Source file names will consist of case-sensitive name with a .java extension

**File Encoding:**  Source files are encoded in standard UTF-8

**Special Characters**

White space

- White space characters will only be found in comments

- Tab characters are for indentation

Non-ASCII characters

- No non-ASCII characters

**Source File Structure:**

**Author**

Identification of the author of the code will be first read on the source file as a comment. The groups name and Description of the code will be included as well.

**Package Statements**

Package Statements are not line wrapped.

**Import Statements**

There will be no wildcard imports, and all imports will be found in one section of the source file. All static imports will be sorted in a single block and all non-static imports in another single block. Only one blank line will separate the state and non-static import statements. There are no other blank lines between import statements.

**Class Declarations**

Every top level class will only have one class declarations in it’s own source file. Each member of the class will have a logical order decided by its user. When a class has multiple constructors or methods with similar names they will appear sequentially with no intervening members.

**Formatting**

**Braces**

Braces are used in if, else, for, do and while statements. even if the body is empty or contains a single statement. There are no line breaks before the opening brace and there will be line breaks after the opening brace, before the closing brace, and every brace that terminates a statement. Empty blocks can be closed immediately after it is opened.

**Block indentation:**

Each time a new black or block-like structure is opened, the indent is increased by a tab character. When the block ends the indent returns to the previous indent level. The indent level applies to both code and comments throughouth the block.

**One statement per line**

Each statement is followed by a line break.

**Column limit**

Source files will follow the standard Java limit of 100

**Whitespace**

A single blank line appears between consecutive members of class (fields, methods, constructors, nested classes), between statements, as required by the document and optionally before the first member or after the last member of each class. Multiple consecutive blank lines are permitted but never encouraged.

Horizontal whitespace are required by the language, other style rules literals, comments and Javadoc. A single ASCII space appears only in separating any reserved word from parentheses, separating reserved words from curly braces, on both sides of any binary or ternary operator, after ,;: on a closing parentheses of a cast, on both sides of a double slash, between the type and variable of declaration and optionally just inside both braces of an array initializer. Horizontal Alignment in never required

**Grouping parentheses**

Optional grouping parentheses are omitted only and reviewer agree that there is no reasonable chance the code will be misinterpreted without them, nor would they have made the code easier to read. I is not reasonable to assume that every reader has the entire Java operator table memorized

**Specific constructs**

enum

Within enum constants line breaks are optional and additional blank lines are also allowed. Since enum classes are classes, they will follow similar format of classes

Variable declarations

There will only be one variable per declaration and will only be declared when needed. The variable declarations do not need to be all in one block of the source code.

Arrays

There will be no C style declarations and array initializers and their block like structure will be decided by what the author is comfortable with.

Switch statements

As with any other block the switch block will be indented with a tab character. After a switch label, a newline appears and the level is increased by one tab character. Within a switch block each statement abruptly terminates or is marked with a comment to indicate is execution. Each switch statement includes a default statement group even if there is no code.

Annotations

Annotations applying to a class, method or constructor appear immediately after the documentation black, and each annotation is listed on a line of its own. These line breaks do not constitute line-wrapping. Annotations applying to a field false appear immediately after the documentation black, but in this care, multiple annotations may be listed on the same line. There are no specific rules for formatting annotations on parameters, local variables, or types.

Comments

Block comments are indented at the same level as the surrounding code.

**Naming**

**General**

Identifiers will only use ASCII letters and digits and we will be using underscores to separate words

**Package Names**

package names are all lowercase, with consecutive words simply concatenated together and with no underscores.

**Class Names**

Class Names are written in UpperCamel|Case. Class names are typically nouns or noun phrases. Interface Names may also by nouns or noun phrases.

**Method Names**

Method names are written in lowerCamel|Case. Method names are typically verb or verb phrases. Underscores may appear in JUnit test method names.

**Constant Names**

Constants are written in all uppercase letters with words separated by underscores. Constants are static final field that are typically nouns or noun phrases.

**Non-constant field names**

Non-constant field names are written in lowerCamel|Case.

**Parameter names**

Parameter Names are written in lowerCamel|Case.

**Local variable Names**

Local variables are written in lowerCamel|Case.

**Type Variable Names**

Type variable names will be the variable name followed by a single capitally letter is optionally followed by a single numeral

**Programming Practices**

**Override**

A method is marked @Override annotation whenever it is legal. This includes a class method overriding a superclass method, a class method implementing an interface method, and an interface method specifying a super interface method.

**Excepetions**

Every exception will be handles and rarely will it be caught doing nothing about it. When its appropriate to do no action a comment will justify why.

**Static members**

**Finalizers**

Finalizers will not be used

**Javadoc**

**General form**

Paragraphs

At-cluases

**Summary**

The Java doc for each class and member begins with a brief summary fragment and it is constructed with complete sentences

**Contents**

Javadoc is present for every public class and for obious methods it is optional. The Javadoc will not be present for Overrides.