Coding Conventions

File Layout

1. Import Statements
2. Javadoc for class.
3. Class Declaration. – Within each section, public first, then protected, then package level, then private.
    a. Class Constants (static & final) variables.
    b. Class (static) variables.
    c. Instance Variables.
    d. Constructors.
    e. Methods

Each class **must** be stored in its own file.

Soft Rules – Can be broken if necessary but should generally be kept to.

Hard Rules – Cannot be broken, unless specified for extreme circumstances.

Modifiers

1. Public/protected/private.
2. Abstract
3. Static
4. Final

E.g. public static void main(String[] args) {
NOT: static public void main(String[] args){

Code Length

1. No lines longer than 80 characters (white space not counted). – Soft Rule
2. No method longer than 75 lines – Soft Rule

Methods

1. No more than 5 levels of indentation. – Soft Rule
2. No more than 5 parameters. – Soft Rule

Continue/Break

1. **Never** use Continue/Break Statements unless in a switch statement.

Identifiers and Case

1. Classes must be written in **Camel Case** and start with a **capital** (Exceptions: Acronyms may be all uppercase e.g. URLTarget). E.g. SalesOrder

2. Constants in **block captials**, with underscores between them. E.g. VAT_RATE

3. All other identifiers (e.g. local variables, methods and parameters) must be written in **camel case**, starting with a **lower-case letter**. E.g. totalCost

4. Class names must be **singular nouns**. E.g. Board

5. Variable names should be **meaningful**. E.g. totalCost – **NOT** – tc

6. Variable names should normally be **singular**, except collections such as arrays. E.g:
   Person neighbour;
   Person[] employees;

7. Methods have meaningful "**verb-like**" names. E.g. calculatePay()

Declarations

1. Only one declaration per line. – Soft Rule E.g:
   int height; //in cm
   int weight; //in grams
2. Always have declarations as close as possible to where they are used – Hard Rule (but can be broken if really needed)

Classes

1. **No blank space** between method name and "(".
   E.g. calculatePay()
   NOT: calculatePay ()

2. Open "{"appears at end of **same line as declaration**, with a space before it.
   E.g. calculatePay() {
   NOT: calculatePay()
   {

3. Closing brace "}" is **on a line of its own** (Exception: if-else, else can be on the closing brace line). E.g.

```
calculatePay() {
    …
    return pay;
}
```

4. Blank Space after keywords (if, else, for, while, do).
   E.g. if (condition) {
   NOT: if(condition){

5. Spaces after semi-colons in for loops.
   E.g. for (int i = 0; i < 10; i++) {
   NOT: for (int i =0;i<10;i++) {

6. All statements **must always use braces** even if they control only one statement.

7. Spaces after commas in argument lists.
   E.g. anotherVariable = methodCall(42, 100, 10);
   NOT: anotherVariable = methodCall(42,100,10);

Binary Operators

1. All binary operators (except ".") must be separated from their operands by a single space.
   E.g. a = c + d;
2. Blank spaces should not be used for unary operators (unary minus (-), ++, --).
   E.g. n = -9;
   E.g. d++;

Arrays and Casting

1. All array access should be followed immediately by a "[".
   E.g. x = myArray[0];
   NOT: x = myArray [0];

2. All casts should be written with a single space between.
   E.g. x = (int) method(42);
   NOT: x = (int)method(42);

Data Encapsulation

1. All class variables (instance and static) **must be private**. (Exception: Constants, they can be public)

2. All class attributes are accessed **using get and set** methods.

No Magic Numbers

1. Don't use magic numbers. A 6 in one place may not be the same as a 6 elsewhere. Instead **define a constant** and use that.

Indentation

1. **USE 4 SPACES FOR INDENTATION – SUPER HARD RULE FOR GROUP 4 DO NOT BREAK**

Javadoc

1. All classes require a class level Javadoc which describes the overall purpose of the class. Must have @author tag. For public entities it must not discuss implementation details.

2. All methods must have a Javadoc comment. Must have @param tag to describe the parameters. Must have @return tag to describe the return value.