Guía Rápida de las Convenciones de Código de Java

http://java.sun.com/docs/codeconv/

Sangrado

1. Utiliza cuatro espacios como unidad de sangría.
2. Utiliza tabulaciones o espacios para sangrar, no una mezcla de ambos. (Excepción: Puede utilizarse una mezcla en el caso de ruptura de líneas).
3. No sangres las clases e interfaces de primer nivel.
4. Sangra un nivel las variables, los métodos y las clases interiores.
5. Sangra un nivel el cuerpo de un método.

Llaves para métodos, clases e interfaces

1. Coloca la llave de apertura en la misma línea que la declaración.
2. Coloca la llave de cierre en una nueva línea y sángrala al nivel de la declaración correspondiente.

```
class Example {

    private void doTask() {

        statements;
    }
}
```

Varios

1. Evita las líneas mayores a 80 caracteres.
2. Una instrucción por línea.
3. Una declaración por línea.
4. Inicializa las variables cuando son declaradas, excepto cuando el valor inicial es desconocido.
5. Si una estructura de control—como una cláusula if o un bucle for—contiene una sola instrucción, dicha instrucción debe estar delimitada por llaves.
6. Utiliza el nombre de la clase, no una referencia, para acceder métodos y variables estáticos.
7. Utiliza paréntesis para clarificar el orden de la evaluación en expresiones complejas.
8. Evita codificar directamente constantes literales. Mejor utiliza una constante simbólica con nombre. (Excepción: 0, 1 y –1 son aceptables.)

Comentarios de implementación

1. *No añadas comentarios para enunciar lo obvio*.
2. Una línea en blanco debe preceder a un comentario.
3. Minimiza la necesidad de comentarios haciendo tu código auto-documentado con nombres apropiados y una estructura lógica explícita.
4. Los comentarios deben proveer información adicional que no es aparente en el código en sí. Los comentarios que presentan un vistazo general de un bloque de código pueden ser útiles.

```
// single-line comment
/* single-line comment */
/*
 * block comment
 */
statement;    // trailing comment
```

Comentarios Javadoc:

1. Utilízalos para documentar clases, interfaces, métodos y variables (con ámbito de clase).
2. Deben describir la entidad que está siendo documentada desde una perspectiva independiente de la implementación.

```
/**
 * Javadoc comment
 */
/** Javadoc comment */
```

Ruptura de Líneas

Cuando una instrucción no cabe en una sola línea:

1. Haz una ruptura después de una coma
2. Haz una ruptura antes de un operador binario
3. Elige las rupturas de alto nivel en vez de las de bajo nivel
4. Alinea una nueva línea con el inicio de la expresión (o lista de argumentos) de la línea anterior:

```
a = b * (c + d - e)
    + (f / g);
x = getValue(a + b + c,
             d + e + f);
```

5. Si estas reglas hacen confuso el código o hacen que el código quede sobre el margen derecho, mejor haz una sangría de 8 espacios (2 tabulaciones).

Convenciones de nombres

1. Los nombres deben ser palabras o frases de palabras. Los nombres deben ser cortos pero descriptivos. Evita las abreviaciones.
2. Clases e interfaces. Utiliza sustantivos, cuando sean varios escribe con mayúscula la primera letra de cada palabra. Ejemplos: TextField y MouseListener
3. Métodos: Utiliza verbos, cuando sean varios escribe la primera letra en minúscula y la primera letra de cada palabra interna con mayúscula. Ejemplo: setBackground
4. Variables: Utiliza sustantivos, cuando sean varios escribe la primera letra en minúscula y la primera letra de cada palabra interna con mayúscula. Ejemplo: fontSize
5. Constantes: Todas con mayúscula con palabras separadas por guiones bajos. Ejemplo: EXIT_ON_CLOSE

Líneas en blanco

Utiliza una línea en blanco:

1. Antes de un comentario
2. Entre métodos
3. Después del encabezado de un método
4. Después de un bloque de declaraciones de variables locales
5. Entre secciones lógicas de código de modo que las instrucciones lógicamente relacionadas estén agrupadas

Espacios

Utiliza un espacio:

1. Entre una palabra clave y un paréntesis izquierdo
2. Después de las comas en las listas de argumentos y parámetros
3. Para separar un operador binario de sus operandos (ver la excepción más adelante)
4. Para separar un operador ternario de sus operandos
5. Entre las partes de inicialización, expresión y actualización de un bucle for
6. Después de una conversión de tipos

No utilices un espacio:

1. Entre el operador punto (.) y sus operandos
2. Entre un operador unario y su operando
3. Entre el nombre de un método y un paréntesis izquierdo

Estatutos return

1. No encierres entre paréntesis el valor de retorno a menos de que de alguna manera, obvien el valor de retorno.
2. Haz que la estructura de tu código coincida con su propósito:

Reemplaza este estatuto if-else:

```
if (booleanExpression) {
    return true;
} else {
    return false;
}
```

con un estatuto return:

```
return booleanExpression;
```

Reemplaza este fragmento de código:

```
if (condition) {
    return x;
}
return y;
```

con un estatuto return:

```
return (condition ? x : y);
```

Estatutos ternarios

Los siguientes formatos son aceptables:

```
a = condition ? b : c;
a = condition ? b
                : c;
a = condition
    ? b
    : c;
```

1. Los paréntesis alrededor de la condición son opcionales.
2. Utiliza paréntesis cuando la condición es una expresión binaria:

```
absoluteValue = (x >= 0) ? x : -x;
```

3. Evita instrucciones ternarias anidadas.
4. Utiliza el operador condicional, no el estatuto if-else cuando asignes un valor a una variable:

```
a = condition ? b : c;
```

Estatutos while

Utiliza el siguiente formato:

```
while (condition) {
    statements;
}
```

Estatutos for

Utiliza el siguiente formato:

```
for (initialization; condition; update) {
    statements;
}
```

Declara la variable de control del bucle dentro del bucle for:

```
for (int i = 0; i < size; ++i) {
    statements;
}
```

Estatutos do-while

Utiliza el siguiente formato:

```
do {
    statements;
} while (condition);
```

Estatutos if-else

Utiliza los siguientes formatos:

```
if (condition) {
    statements;
}
if (condition) {
    statements;
} else {
    statements;
}
if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

Estatutos switch

Utiliza el siguiente formato:

```
switch (condition) {
case ABC:
   statements;
   /* falls through */

case DEF:
   statements;
   break;

default:
   statements;
   break;
}
```

1. Siempre incluye un caso default.
2. Utiliza la línea de comentario */* falls through */* cuando la etiqueta case no tenga ninguna instrucción break.

Bloques try-catch

Utiliza el siguiente formato:

```
try {
    statements;
} catch (ExceptionClass e) {
    statements;
}
```

Recorte de líneas para estatutos if

Utiliza la regla de 8 espacios (2 tabulaciones) cuando envuelvas una cláusula if para que el cuerpo sea más fácil de ver:

```
if ((a && b)
        || (c && d)
        || (e && f)) {
    statements;
}
```