Guía de estilo de programación en Java

1. Nombres

1.1 –Utiliza nombres descriptivos.

> Utiliza nombres descriptivos para todos los identificadores (nombres de clases, variables y métodos). Evita la ambigüedad. Evita las abreviaciones.
>
> Los métodos mutadores simples deberían llamarse setAlgo(...).
>
> Los accesores simples deberían llamarse getAlgo(....). Los métodos accesores que devuelven un valor boolean se nombran a menudo como esAlgo(...), estaAlgo(...), por ejemplo, esPrimo(), estaVacia().

1.2 Los nombres de clase empiezan siempre con letra mayúscula y son sustantivos en singular. Se escriben en formato **UpperCamelCase**, esto es, cuando se trata de un nombre compuesto cada una de las palabras que forman el nombre se capitaliza: BicicletaMontaña, SocioHabitual.

1.3 Los nombres de métodos empiezan en letra minúscula. Habitualmente son verbos o frases verbales. Se escriben en formato **lowerCamelCase**, es decir cuando se trata de un verbo compuesto cada una de las palabras que siguen a la primera se capitaliza: parar(), enviarMensaje()

1.4 Las variables (atributos, variables locales y parámetros) son sustantivos y empiezan en letra minúscula. Se escriben también en formato **lowerCamelCase:** numeroIntentos, totalRecaudadoDia

1.5 Las constantes se escriben en mayúsculas. Las constantes utilizan ocasionalmente el símbolo subrayado (_) para identificadores compuestos: MAXIMO_INTENTOS, CABECERA_FACTURA.

1.6 Los nombres de paquetes se escriben en minúsculas: com.daw.programacion.ejemplo

2. Aspecto

2.1 Un nivel de indentación son cuatro espacios.

2.2 Todas las sentencias dentro de un bloque se indentan un nivel.

2.3 Las llaves en las clases y métodos están en una línea o al final de línea (elegir una opción y mantenerla)

```
public int getEdad()
{
        sentencias
}
```

```
public int getEdad() {
        sentencias
}
```

2.4 Para el resto de bloques (if, while, for) las llaves también se sitúan en una línea separada o al final de línea (elegir una opción y mantenerla)

```
if (condition)
{
        sentencias
}
else
{
        sentencias
}
while (condition)
{
        sentencias
}
```

```
if (condition) {
        sentencias
}
else {
        sentencias
}
while (condition) {
        sentencias
}
```

2.5 Utiliza siempre llaves en las estructuras de control incluso cuando incluyan únicamente una sentencia.

2.6 Utiliza un espacio alrededor de los operadores.

2.7 Utiliza una línea en blanco alrededor de métodos y constructores.

2.8 Utiliza líneas en blanco para separar bloques lógicos de código. Hay que hacerlo entre métodos, pero también entre partes lógicas dentro de un método.

3. Documentación

3.1 Cada clase tiene un comentario de clase al principio.

El comentario de clase contiene al menos:

- una descripción general de la clase
- el nombre (s) del autor (es)
- el número de versión

3.2 Cada método incluye un comentario de método.

3.3 Los comentarios son comentarios *javadoc*.

Los comentarios de clases y métodos deben ser reconocidos por *javadoc*, por tanto, deben comenzar con el símbolo de comentario /**.

3.4 Incluye comentarios en el código solo cuando sea necesario.

Los comentarios en el código hay que incluirlos únicamente cuando el código no sea claro o sea difícil de entender. No hay que comentar las sentencias obvias.

4. Restricciones en el uso del lenguaje

4.1 Orden de declaraciones: **atributos, constructores, métodos**.

> En la definición de una clase sus elementos aparecen en el siguiente orden: sentencia package, sentencias de importación de paquetes, comentarios de clase, cabecera de la clase, definición de atributos, constructores, métodos

4.2 Los atributos no deben ser públicos.

4.3 Utiliza siempre modificadores de acceso (especifica la visibilidad).

> Especifica para todos los atributos y métodos si son *private*, *public* o *protected*. No utilices nunca el acceso por defecto (*package*)

4.4 Importa las clases separadamente.

> Las sentencias *import* son más claras si se indica la clase concreta que se importa en lugar de incluir el paquete completo.
>
> ```
> import java.util.ArrayList;
> import java.util.HashSet;
> ```
>
> es mejor que,
>
> ```
> import java.util.*;
> ```

4.5 Incluye siempre un constructor (incluso aunque el cuerpo del constructor esté vacío).

4.6 Incluye siempre la llamada al constructor de la superclase.

> En los constructores de las subclases incluye una llamada explícita *super(...)* al constructor de la superclase.

4.7 Inicializa todos los atributos en el constructor.

4.8 Utiliza iteradores con las colecciones, no índices.

5. Problemas habituales a evitar

5.1 No asignes valores a los parámetros de un método dentro del método

5.2 Simplifica las expresiones booleanas tal como `(if primo == true)` ...
En su lugar, `if (primo)` .....

5.3 Evita sentencias if innecesarias al devolver un valor booleano.

```
public boolean esPar() {

        if (valor % 2 == 0) {
                return true;
    }
    return false;
}
```

```
public boolean esPar(int valor) {

        return valor % 2 == 0;
}
```

5.4 No modifiques la variable de control dentro de un bucle for.

5.5 Evita asignaciones en subexpresiones: `String str = Integer.toString(i = 2);`