NORMAS BÁSICAS DE CODIFICACIÓN EN JAVA

➔ **Todos los ficheros fuente en Java tienen la siguiente estructura**:

1.- Comentario inicial (*nombre de clase, versión, fecha y copyright*)
2.- Sentencias Package e Import
3.- Declaraciones de clase o interface

➔ **Las partes de una declaración de clase o interface deben seguir el siguiente orden**:

1. Comentario de documentación de la clase o interface `(/** ... */)`
2. Sentencia `class` o `interface`
3. Comentario de implementación de la clase o interface `(/* ... */)` si es necesario.
4. Variables de clase, estáticas (*públicas, protegidas* y luego *privadas*)
5. Variables de instancia (*públicas, protegidas* y luego *privadas*)
6. Constructores
7. Métodos (*públicos, protegidos* y luego *privados*)

➔ **Los programas pueden tener dos tipos de comentarios**:

1.- *Comentarios de documentación*. Para generarlos se utilizará la herramienta JavaDoc.

2.- *Comentarios de implementación*. Pueden ser de tres tipos:

Comentario de bloque	*Comentario de línea*	*Comentario corto*

`/*`
` * Un comentario de bloque`
` */`
	`/* comentario de línea */`	`// Comentario corto`

➔ **Declaraciones**

✓ Se realizará una única declaración por línea e inicializar las variables locales donde se declaran. También se deben poner las declaraciones al comienzo de los bloques. Por ejemplo:

```
void nombreMetodo() {
    int valor = 0; // Comienzo del bloque del método
    if (condicion) {
       int suma = 0; // Comienzo del bloque "if"
       ...
    }
}
```

✓ Se utilizará el siguiente formato para la declaración de clases e interfaces:

```
class Ejemplo extends Object {
   private int suma;
   private int contador;

   Ejemplo(int i, int j) {
     suma = i;
     contador = j;
   }

   int metodoVacio() {}
   ...
}
```

- No dejando un espacio en blanco entre el nombre del método y el paréntesis "(" y separando todos los métodos por una línea en blanco.

- Colocando las llaves de apertura "{" en la misma línea y las de cierre "}" en línea aparte y en la misma columna que inicio el bloque.

✓ Se utilizará el siguiente formato para las sentencias:

1. Sentencia `return`

```
return;
return Matrices.longitud();
return ((a >b) ? a : b);
```

2. Sentencia `if, if-else, if else-if else`

```
if ( condicion) {
   sentencias;
}
```

```
if ( condicion) {
     sentencias;
} else {
     sentencias;
}
```

```
if ( condicion) {
     sentencias;
} else if ( condition) {
     sentencias;
} else {
     sentencias;
}
```

3. Sentencia `for`

```
for(initializacion; condicion; actualizar) {
     sentencias;
}
```

4. Sentencia `while`

```
while(condicion) {
     sentencias;
}
```

5. Sentencia `do-while`

```
do {
    sentencias;
} while (condicion);
```

6. Sentencia `switch`

```
switch ( condicion) {
       case ABC:
           sentencias;
           /* sigue la ejecución  */

       case DEF:
           sentencias;
           break;

       case XYZ:
           sentencias;
           break;

       default:
           sentencias;
           break;
}
```

7. Sentencia `try-catch`

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

```
try {
    sentencias;
} catch (ExceptionClass e) {
    sentencias;
} finally {
    sentencias;
}
```

➔ **Convenios de nombrado**

1. Paquetes

- Siempre escrito en letras minúsculas
- Peden reflejar algún tipo de organización jerárquica con susu niveles separados por puntos.

- Ejemplos:

```
com.sun.eng
com.apple.quicktime.v2
edu.cmu.cs.bovik.cheese
```

2. Clases e Interfaces

   - Se deben usar sustantivos para las clases.
   - Se debe tratar de usar nombres sencillos y descriptivos.
   - Se deben capitalizar todas las iniciales en el caso de usar nombres formados por varias palabras, incluida la primera.
   - Ejemplos:

```
public class Ventana
public class BolsaCaracteres
```

3. Métodos

   - Se deben usar verbos en infinitivo.
   - Se debe tratar de usar nombres sencillos y descriptivos.
   - Se deben capitalizar todas las iniciales en el caso de usar nombres formados por varias palabras, pero no la primera.
   - Ejemplos:

```
public void dibujarEnLienzo()
public Nodo buscarAnterior(Nodo n)
```

4. Variables

   - Los nombres deberán ser sencillos y descriptivos.
   - Se deben capitalizar todas las iniciales en el caso de usar nombres formados por varias palabras, pero no la primera.
   - En el caso de variables que son usadas dentro de bloques cortos, temporales, se pueden nombres de una sola letra. Para enteros `i, j, k, l`, etc. Para letras `c, d, e, f`, etc.
   - Ejemplos:

```
int  puntoInicial;
String nombreFichero;
```

5. Constantes

   - Los nombres deberán descriptivos.
   - Se escriben en mayúsculas con las palabras separadas por "_".
   - Ejemplos:

```
public static int TAMAÑO_MAXIMO = 50;
publicc static String TITULO = "Apartado de inicio"
```

➔ **Indentación**

1. Como norma general se usarán 4 espacios como salto de indentación.
2. Se deben evitar las líneas más largas de 80 caracteres. Probablemente no serán representadas de forma adecuada en todos las resoluciones de pantalla, terminales e impresoras si son más largos.