Recomendaciones para programar en JAVA

Razones por la que los programadores deben mantener o llevar un orden en la estructura del programa son por los siguientes motivos:

1.- El 80% de los programas tiene algun tipo de mantenimiento. Quiere decir que en algun momento, vas a tener que darle algún tipo de mantenimiento a tu software (ya sea algun cambio, actualización, mejorar el software, etc.).

2.- Difícilmente el software es mantenido por el autor original. Que no siempre será el autor, quien escribió el programa, el que le de mantenimiento al software, algunas veces serán otras personas, y pues el código debe ser legible para cualquier persona.

3.- La nomenclatura ayuda a que los programadores comprendan más rápidamente el código del programa. Una buena estructura siempre hará el código mas legible.

En cuanto a gramática, las reglas con estas:

1. **Clases:** Los nombres de clases e interfaces siempre deben comenzar con la primera letra en mayúscula, deben ser simples y descriptivos. Ejem: Coche(), Vehiculo(), PruebaApplet().

2. **Metodos:** Deben comenzar con letra minúscula, y si está compuesta por 2 palabras, la primera letra de la segunda palabra debe comenzar con mayúscula. De preferencia que sean verbos. Ejem: arrancarCoche(), sumar().

3. **Variables:** las variables siguen la misma gramática que los métodos. Evite usar variables de una sola letra (a, b, c.). Las variables constantes o finales, las cuales no cambian su valor durante todo el programa se deben escribir en mayúsculas. Ejemplo: ANCHO, VACIO, IVA...

4. **Variables constantes:** las variables siguen la misma gramática que los métodos. Las variables constantes o finales, las cuales no cambian su valor durante todo el programa se deben escribir todo en mayúsculas. Ejemplo: ANCHO, VACIO, IVA...

Ahora profundicemos:

1.- SUFIJOS DE ARCHIVOS

Existen dos tipos de archivos los **.java** y los **.class**. El **.java** que es el código fuente del programa, el código que escribe el usuario. Y el **.class** el bytecode, el código que interpreta la Maquina Virtual de Java.

2.- ORGANIZACION DEL ARCHIVO

- Se deben evitar aquellos archivos que superen las **2000 lineas**, un lema de la Programacion Orientado a Objetos es *"Divide y Venceras"* , eso quiere decir que debemos dividir el programa en partes para evitar confusiones y llevar un control más claro del programa.

- Cada archivo fuente tiene una sola Clase o Interfaz Publica. Cuando una Clase Privada este asociada a la Clase Publica, este puede ser incluido dentro del mismo archivo fuente. La Clase Publica debe ser la primera declarada en el archivo.

El archivo/código fuente debe seguir el siguiente orden:

1. **Comentarios iniciales:** Aqui se especifica el nombre de la clase, la fecha, el nombre del autor, etc.

2. **Sentencias Package e Import:** package Hola, import java.awt.*;

3. **Declaración de Clases e Interfaces:** El cuerpo queda de la siguiente forma:
   a. **Variables/atributos de Clase**, primero se declaran las variables **publicas**, luego las **protegidas**, las de **nivel de paquete** (sin modificador de acceso), y por ultimo las **privadas**.
   b. Constructores
   c. **Métodos,** los cuales deben ser agrupados por funcionalidad y no por el orden de uso.

3.- SANGRIAS

- El tamaño de cada línea no debe superar los **80 caracteres**. (linea roja vertical que aparece en NetBeans)
- Cuando una expresión no cabe en una sola línea, hay que proseguir abajo:
  - Después de una coma ” , “.

  - Despues de un operador (+, -, *, etc).

- Se debe alinear la nueva línea al mismo nivel de la linea anterior donde comenzó la expresión.

4.- COMENTARIOS

Comentarios de Implementacion estan delimitados por

/*…*/ para varias líneas de código

// para una sola línea de código

Cuando se utilizan demasiados comentarios, se puede reflejar una mala calidad en el codigo. Antes de agregar un comentario, considera re-escribir el codigo para una mayor claridad.

5.- DECLARACIONES

Se recomienda declarar una variable por cada línea si se incluyen aclaraciones:

```
int longitud;  //longitud del mueble expresada en metros

boolean matriculado;  //true si la persona sí está matriculada, false si no lo está
```

Trate de inicializar variables donde se declaren, la única razón para no inicializar una variable donde se declara es solo si el valor inicial de esa variable depende de alguna operación posterior

Coloque las declaraciones al principio de cada bloque (delimitados por { }) y no espere a declarar la variable justo en el momento en el que se utiliza.

```
void metodo() {
      int producto;
}
```

Cuando se declaren las **Clases** y las **Interfaces**, hay que seguir las siguientes reglas:

- No dejar espacio entre el nombre del método y el paréntesis de apertura ” (“.
- La llave de apertura ” { ” aparece en la misma linea de la declaracion de la Clase/Interfaz

```
void ClaseEjemplo() {
      int e;
}
```

- La llave de cerradura ” } ” debe estar alineada con la linea de la decalaracion de la Clase/Interfaz.
- Si el método esta vacio las llaves van en la misma línea de declaración:

```
int getCalificaciones() { }
```