UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

Coordinación General de Tecnologías de Información

Sistemas de Información

Estándares de Codificación para Plataforma

Tradicional en JAVA

COPIA
NO CONTROLADA

CONTENIDO

1 - Organización de los archivos 1
1.1 Archivos fuente Java 1
1.1.1 Comentarios de inicio 1
1.1.2 Estructura de clases e interfaces 2
2 - Identación 2
2.1 Formato para líneas extensas 2
3 - Comentarios 2
4 - Declaraciones 3
4.1 Cantidad por línea 3
4.2 Inicialización 3
4.3 Colocación 3
4.4 Declaraciones de clases e interfaces 3
5 - Sentencias 4
5.1 Estructura de las sentencias de control 4
5.1.1 Sentencias if, if-else, if else-if else 4
5.1.2 Sentencias for 4
5.1.3 Sentencias while 5
5.1.4 Sentencias do-while 5
5.1.5 Sentencias switch 5
5.1.6 Sentencias try-catch 5
6 - Convenciones de nombres 6
6.1 Tipos de identificadores 6
6.2 Nombramiento de componentes 7
6.3 Nombramiento de controles de menú 8
7 - Objetos 8
8 Conexión a base de datos 8

1 - Organización de los archivos

■ Un proyecto debe de estar organizado en los siguientes paquetes en letra minúscula;

nombreproyecto
- modelo
- presentacion
  - formas
  - reportes
  - recursos
    - imagenes
- basedatos

1.1 Archivos fuente Java

■ Cada archivo fuente Java debe tener una clase o interface pública. Cuando algunas clases o interfaces privadas están asociadas a una clase pública, deben ponerse anidadas a la clase pública.

■ Los archivos fuentes Java deben tener la siguiente ordenación:

- Comentarios de inicio
- Sentencias package e import
- Declaraciones de clases e interfaces

1.1.1 Comentarios de inicio

■ Todos los archivos fuente deben comenzar con un comentario en el que se lista el nombre del autor, fecha de creación y la historia de cambios mediante la utilización del siguiente formato:

```
/*
* Autor:    Juan Carlos Rodríguez Cota
* Fecha:    dd-MMM-aaaa
* Historia de cambios
```

* Versión	Solicitud Servicio	Modificador	Fecha	Descripción del cambio	Versión del estándar
* #.#.#	#Solicitud	Nombre	dd/mmm/yyyy	Descripción del cambio	Versión del documento que se empleó para la codificación.

```
*
*/
```

Donde:

Versión:	Versión de submódulo liberado.
Solicitud Servicio:	Solicitud registrada en Sistema de Gestión para la Atención a Usuarios (SGAU's). Cuando no aplique debe de especificarse N/A.
Modificador:	Persona que realizó el cambio.
Fecha:	Fecha en que se realizó el cambio.
Descripción del cambio:	Cambios / adecuaciones realizadas al proceso.
Versión del estándar:	Versión del documento de codificación de Java DeskTop empleado.

1.1.2 Estructura de clases e interfaces

Partes de la estructura de una clase o interface	Notas
Comentario de documentación de la clase o interface (`/**...*/`)	Los comentarios de documentación deben describir la especificación del código, y se deben aplicar a todas las clases y métodos públicos.
Sentencia `class` o `interface`	Nombrar clases e interfaces de acuerdo a este estándar.
Atributos de clase	Se deben agrupar por funcionalidad más que por visión o accesibilidad.
Métodos	Los métodos se deben agrupar por funcionalidad más que por visión o accesibilidad.

2 - Identación

■ Se deben emplear tabulaciones para identar el código y facilitar su lectura.

2.1 Formato para líneas extensas

■ Cuando una línea de código sea demasiado extensa, se debe dividir de acuerdo con estos principios:

- Después de una coma.
- Antes de un operador.
- División alto nivel (más a la derecha que el "padre") que de bajo nivel (más a la izquierda que el "padre").

Ejemplos de como dividir la llamada a un método:

```
unMetodo(expresionLarga1, expresionLarga2, expresionLarga3,
         expresionLarga4, expresionLarga5);

variable = unMetodo1(expresionLarga1,
                unMetodo2(expresionLarga2,
                          expresionLarga3));
```

3 - Comentarios

■ Para datos públicos (*clases, interfaces, constructores, métodos);* utilizar el siguiente formato con el objetivo de orientarlo a otros programadores:

```
/**
* La descripción de la clase, interface, constructores, métodos
* se escribe aquí.
* @versión (número y fecha en formato dd/mmm/aaaa)
* @author Nombre(s) y Apellido(s)
*/
```

■ Para el resto del código utilizar // o /* */ según juzgue el programador.

■ Para permitir una fácil determinación de quien ha alterado piezas de código, y deducir cuando los cambios fueron realizados, el siguiente formato debe ser adoptado para todos los comentarios que describan cambios al código:

```
/*
* Versión:    (ligada a comentario inicial de la clase) 1.0.0
* Cambio:     Cambio que se realizó
*/
```

4 - Declaraciones

4.1 Cantidad por línea

■ Se debe tener una declaración por línea. En otras palabras, se prefiere:

```
int         intNivel    = 0;        // Nivel de identación
String      strPuesto   = "";       // Nombre del puesto
```

antes que

```
int intNivel, intCantidad;
```

4.2 Inicialización

■ Inicializar las variables donde se declaran.

4.3 Colocación

■ Poner las declaraciones solo al principio de los bloques (un bloque es cualquier código encerrado por llaves "{" y "}"). No esperar al primer uso para declararlas.

```
void myMethod() {
      int intContadorUno = 0;             // Bloque del método
      if (condicion) {
            int intContadorDos = 0;       // Bloque del "if"
            ...
      }
}
```

■ La excepción de la regla son los índices de bucles `for`, que en Java se pueden declarar en la sentencia `for`:

```
for (int intRenglon = 0; intRenglon < intMaximoVueltas; intRenglon++){
      ...
}
```

4.4 Declaraciones de clases e interfaces

■ Al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:

- Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.

- La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.
- La llave de cierre "}" empieza una nueva línea identada para ajustarse a su sentencia de apertura correspondiente.
- Los métodos se separan con una línea en blanco.

```
class clsEjemplo extends Object {
      int intVariableUno = 0;
      int intVariableDos = 0;

      Ejemplo(int intParamteroUno, int intParametroDos) {
            intVariableUno = intParamteroUno;
            intVariableDos = intParametroDos;
      }

      ...
}
```

5 - Sentencias

■ Cada línea debe contener una sola sentencia. Ejemplo:

```
intVariable++;                    // Correcto
intContador--;                    // Correcto
intVariable ++; intContador --;   // Incorrecto
```

5.1 Estructura de las sentencias de control

5.1.1 Sentencias if, if-else, if else-if else

■ La clase de sentencias `if-else` debe tener la siguiente forma:

```
if (condicion) {
      sentencias;
}

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

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

■ Las sentencias `if` usan siempre llaves {}.

5.1.2 Sentencias for

■ Una sentencia `for` debe tener la siguiente forma:

```
for (inicialización; condición; actualización) {
      sentencias;
}
```

■ La sentencia for debe tener solo una variable de inicialización y actualización.

5.1.3 Sentencias while

■ Una sentencia `while` debe tener la siguiente forma:

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

5.1.4 Sentencias do-while

■ Una sentencia `do-while` debe tener la siguiente forma:

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

5.1.5 Sentencias switch

■ Una sentencia `switch` debe tener la siguiente forma:

```
switch (condicion) {
case ABC:
      sentencias;

case DEF:
      sentencias;
      break;

case XYZ:
      sentencias;
      break;

default:
      sentencias;
      break;
}
```

5.1.6 Sentencias try-catch

■ Una sentencia `try-catch` debe tener la siguiente forma:

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

ó.

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

6 - Convenciones de nombres

6.1 Tipos de identificadores

<table>
<tr><th>Tipos de identificadores</th><th>Reglas</th><th>Ejemplos</th></tr>
<tr><td>Clases</td><td>■ Inician con el prefijo “cls”.
■ Los nombres de las clases se escriben con letra capitalizada.
■ Mantener los nombres de las clases simples y descriptivas.
■ Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).</td><td>clsCliente;
clsImagenAnimada;</td></tr>
<tr><td>Interfaces</td><td>■ Los nombres de las interfaces siguen la misma regla que las clases.</td><td>interface clsObjetoPersistente;
interface clsAlmacen;</td></tr>
<tr><td>Métodos</td><td>■ El nombre del método debe reflejar la acción que realiza, cuando son compuestos tendrán la primera letra de la primer palabra en minúscula, y la primera letra de la o las siguientes palabras que lo forman en mayúscula.</td><td>realizarCalculoNomina();
dibujarCuadro();
mostrarFondo();</td></tr>
<tr><td>Variables</td><td>■ Inician con el prefijo (en minúsculas) que codifica el tipo de dato para la variable.
■ Los nombres de las variables se escriben con letra capitalizada.
■ La elección del nombre de una variable debe ser un mnemónico.</td><td>
<table>
<tr><th colspan="2">Prefijo</th></tr>
<tr><td colspan="2">Primitivos</td></tr>
<tr><td>byt</td><td>Byte</td></tr>
<tr><td>sht</td><td>Short</td></tr>
<tr><td>int</td><td>Int</td></tr>
<tr><td>lng</td><td>Long</td></tr>
<tr><td>flt</td><td>Float</td></tr>
<tr><td>dbl</td><td>Double</td></tr>
<tr><td>chr</td><td>Char</td></tr>
<tr><td>bln</td><td>Boolean</td></tr>
<tr><td colspan="2">Estructurados</td></tr>
<tr><td>str</td><td>String</td></tr>
<tr><td>arl</td><td>Array List</td></tr>
<tr><td>dte</td><td>Date</td></tr>
<tr><td>mdl</td><td>Modelos</td></tr>
<tr><td>(tipo dato)</td><td>Arreglos</td></tr>
<tr><td>obj</td><td>Objetos instanciados de clases de usuario</td></tr>
</table>
</td></tr>
<tr><td>Constantes</td><td>■ Los nombres de las constantes deben ir totalmente en mayúsculas separando las</td><td>static final int ANCHURA_MINIMA = 4;</td></tr>
</table>

<table>
<tr><td></td><td>palabras con un subguión ("_").</td><td>static final int ANCHURA_MAXIMA = 999;
static final int ORIENTACION_HOJA = 1;</td></tr>
<tr><td>Objetos de base de datos</td><td>■ Para la declaración de objetos de base de datos, se debe de poner un prefijo (en minúsculas) seguido del nombre del mismo, con letra capitalizada.</td><td><table><tr><th>Objeto</th><th>Prefijo</th></tr><tr><td>Connection</td><td>sqlc</td></tr><tr><td>Statement</td><td>stmt</td></tr><tr><td>PrepareStatment</td><td>pstmt</td></tr><tr><td>CallableStatement</td><td>cstmt</td></tr><tr><td>ResultSet</td><td>rs</td></tr><tr><td>…</td><td></td></tr></table></td></tr>
<tr><td>Reportes</td><td>■ Inician con el prefijo “rpt”.
■ Los nombres de los reportes se escriben con letra capitalizada.
■ Mantener los nombres de los reportes simples y descriptivos.
■ Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).</td><td>rptClientes

rptHorarioAlumnos</td></tr>
</table>

6.2 Nombramiento de componentes

■ Los controles deben ser nombrados con un prefijo que facilite la identificación del tipo de objeto, seguido por el nombre asignado con la primera letra en capital de cada palabra. Estos prefijos se enlistan a continuación:

Control	Prefijo
Text Field	txt
Text Area	txta
Button	but
List	lst
Check Box	chk
Frame	frm
Combo Box	cbo
Label	lbl
Radio Button	rbut
Button Group	butg
Scroll Bar	scb
Progress Bar	prb
Slider	sld
Formatted Field	fof
Spinner	spn
Password Field	pwf
Separator	sep
Text Pane	txtp

Control	Prefijo
Table	tbl
Menu Bar	mnub
Menu	mnu
Menu Item	mnu
Menu Item / CheckBok	mnu
Menu Item / RadioButton	mnu
Popup Menu	mnup
Panel	pnl
Tabbed Panel	tab
Internal Frame	frm
Layered Panel	lay
Split Panel	spl
Dialog	dlg
Color Chooser	col
File Chooser	fil

6.3 Nombramiento de controles de menú

■ Los prefijos para los controles de menú deben extenderse más allá de la etiqueta inicial "mnu" mediante la asignación de un prefijo adicional por cada nivel de anidación. La siguiente tabla enlista algunos ejemplos:

Secuencia de leyendas en el menú	Nombre asignado al menú
Archivo Abrir	mnuArchivoAbrir
Archivo Enviar Email	mnuArchivoEnviarEmail
Archivo Enviar Fax	mnuArchivoEnviarFax
Formato Carácter	mnuFormatoCaracter
Ayuda Contenido	mnuAyudaContenido

7 - Objetos

■ Cada objeto debe ser destruido cuando deje de utilizarse para ayudar a la administración de la memoria. Utilizar objNombreObjeto = null;.

8 Conexión a base de datos

■ En la configuración de los parámetros de conexión a bases de datos, no deberá utilizarse el nombre de usuario "sa" en ninguno de los sistemas para su operación.