Universidad Tecnológica del Centro de Veracruz

Tecnologías de la información y la comunicación

Área Sistemas Informáticos

7°A DZ

ESTANDARES

Presenta:

González Blanco María de los Ángeles

Ramírez García Alan Hernán

Sosa Rincón Daniel Isaac

H. Cuitláhuac, Ver a 9 de octubre del 2015

Estándares de codificación

Nombre de clases

- Iniciar siempre con letra mayúscula y el resto en minúsculas.
- Los nombres de clases deben ser escritos con la primer letra de cada palabra interna en mayúsculas (Clase, ClaseDos).
- No utilizar caracteres especiales ni contener espacios.
- Los nombres de las clases deben ser simples y descriptivos.
- Usar palabras completas y evitar acrónimos y abreviaturas
- Si la clase hace referencia a algún patrón determinado se definirá en el nombre (DAO, VO, DTO, etc.).
- Debe ser un sustantivo

Nombre de métodos

- Deberán ser verbos en infinitivo.
- Primera letra del nombre en minúscula.
- De ser formado por dos palabras, la segunda palabra iniciara con mayúscula (nombreMetodo).
- El nombre debe ser descriptivo.

Nombre de variables

- Debe ser descriptivo (mostrar el propósito de su uso).
- Deben ser nombres cortos.
- No utilizar caracteres especiales.
- No utilizar nombre sin ningún significado funcional.
- Los nombres de un solo carácter deben ser evitados, excepto para usarlos como variables temporales.

Nombre de constantes

- Los nombre de variables declaradas como constantes deben ser todas en mayúsculas con palabras separadas por guion bajo (“_”).

- (String PROPERTY_URL_SERVICIO = "urlServicio";)

Controles

- Utilizando las iniciales de cada uno de los controles para su identificación más rápida se implementaran los prefijos de los nombre (Check box: cb, Label (lbl): lb, Button(btn): btn).

Directrices de codificación

Apertura vertical entre contextos

En él, el flujo y la secuencia de las operaciones, va de arriba hacia abajo. Es una lista ordenada de las operaciones de un proceso con toda la información que se considere necesaria, según su propósito.

Ejemplo:

```
public void crearReporte(){

        private int a;

        private int b;

        private String val1;

        private string val2;


        public void imprimirPantalla(){

                System.out.println(“numeros: “ + a + “,” + b;

                System.out.println(“cadenas de caracter: “ + val1 + “,” + val2;

        }

}
```

Densidad vertical

Si la apertura separa los conceptos, la densidad vertical implica asociaciones. Por tanto, las líneas de código con una relación directa deben aparecer verticalmente densas. Es mucho más fácil de leer cuando la estructura asocia los campos comunes de código.

Funciones dependientes

Si una función invoca a otra, deben estar verticalmente próximas, y la función de invocación debe estar por encima de la invocada siempre que sea posible, de este modo el programa fluye con normalidad. Si la convención se sigue de forma viable, los lectores sabrán que las definiciones de la función aparecen después de su uso.

La función superior invoca las situadas por debajo que, a su vez, invocan a las siguientes. Esto facilita la detección de las funciones invocadas y mejora considerablemente la legibilidad del módulo completo.

Ejemplo:

```
public void aceptar(String saludo){
        actualizar(saludo);
}
public void actualizar(String saludo){
        System.out.println(saludo);
```

Formato horizontal

¿Qué ancho debe tener una línea de código? Las líneas de código deberían ser con el antiguo limite Hollerith de 80 caracteres, sin embargo, es un límite en algunos casos pequeño así que se establece actualmente un límite de 120 caracteres.

Sangrado

Un archivo de código es una jerarquía más que un entorno. Incluye información que pertenece a la totalidad del archivo, a sus clases individuales, a los métodos de las clases, a los bloques de los métodos y a los bloques de los bloques. Cada nivel de esta jerarquía es un ámbito en el que se pueden declarar nombre y en el que se interpretan declaraciones e instrucciones ejecutables.

Las instrucciones al nivel del archivo, como las declaraciones de clases, no se sangran.

Romper el sangrado

En ocasiones tenemos la tentación de romper la regla del sangrado, sin embargo, esta regla es la que da formato al código, en especial a instrucciones *if* y bucles *while* de contenido simple.

Ejemplo:

Incorrecto

```
public void comprobar(int a, int b){if(a>b){System.out.println("A es mayor que b")}}
```

correcto

```
public void comprobar(int a, int b){
```

```
    if(a>b){
        System.out.println("A es mayor que b")
    }
}
```

Tamaño de sangrado

Establezca una convención para el tamaño del sangrado que prefiera y entonces aplique de manera uniforme dicha convención. Puede utilizar la tecla *tab* para crear el sangrado, aunque tab podría variar entre editores. Le recomendamos el uso de tabuladores de ¼ de pulgadas o tres espacios para formar un nivel de sangrado.

Ejemplo:

```
public void iniciar(){
    if(a>b){
        System.out.println("mensaje");
    }
]
```

Donde añadir llaves

El uso de las llaves se aplica directamente después del paréntesis de cierre, esto aplica en una función y en cláusulas de condición, *if else, for, while, switch;* en el caso de los arrays las llaves se colocan después del corchete de cierre.

Entre las llaves se coloca el contenido, una vez completo el contenido se coloca la llave de cierre al mismo nivel de inicio de la condición en la parte superior.

Ejemplo:

```
If(condicion){
    Contenido…
}
Int [] numbers = new int[] {1,2,3,4,5};
```

Estándares de la Base de Datos

Los nombres de las tablas y campos deben especificarse bajo el estándar camelCase. Este estándar especifica escribir las palabras compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra. En este ámbito se utilizara la variante lowerCamelCase (la primer letra del nombre, en minúscula).

Únicamente se utilizaran caracteres alfabéticos, salvo que por la naturaleza del nombre se necesiten dígitos numéricos. Se prohíbe el uso de caracteres de puntuación o símbolos. Ejemplo listaAspirantes2015

Las letras acentuadas se reemplazarán con las equivalentes no acentuadas, y en lugar de letra eñe(ñ) se utilizara (ni). Ejemplo: anioCierre

El nombre elegido debe ser lo más descriptivo posible, evitando términos ambiguos o que se presten a distintas interpretaciones. Ejemplo : tiposMaterial => categoriasMateriales

El nombre no debe abreviarse, salvo que por necesidad específica deban especificarse más de una palabra en el mismo. Ejemplo: freg=>fechaRegistro

Agregar comentarios a las bases de datos y los campos, sobre todo a los booleanos.

Tablas

Los nombres deben especificarse en plural, y de acuerdo a las reglas generales. Ejemplo: departamentos, facturas, monedas.

En el caso de tablas que se relacionan específicamente con otra tabla (ejemplo: tablas tipo, nomencladores, entidades débiles), esta relación debe quedar expresada en el nombre.

Ejemplo: domicilioPersona, categoríaMateriales.

Las tablas de relación (objetos asociativos, representan relaciones de N a M) deben nombrarse utilizando los nombres de las tablas intervinientes, siguiendo un orden lógico de frase.

Ejemplo: localidadesMunicipios, facturasNotas

Campos Clave (identificadores de tabla)

Toda tabla debe poseer uno o más campos clave.

Toda relación entre tablas debe implementarse mediante constrains (claves foráneas) con integridad referencial, de acuerdo al motor de base de datos utilizado.

La integridad referencial deberá actualizar en cascada en todos los casos, y restringir el barrado salvo para las entidades débiles.

Ejemplo: No se podrá eliminar un registro de la tabla Materiales que tenga ocurrencias en otras tablas; para que este caso deberá implementarse el borrado lógico. Por el contrario, si podrá habilitarse el borrado en cascada si la relación fuera entre las tablas Notas y renglonesNotas.

Los campos clave deben ubicarse al inicio de la definición de la tabla (deben ser los primeros).

El nombre del campo clave debe estar compuesto por “id” + nombre de la tabla en singular (para claves no compuestas).Dependiendo de la naturaleza de la entidad, el nombre de la tabla a usar es el de la misma tabla, o el de la relacionada.

Ejemplos: tabla materiales =>idMaterial.

Las claves compuestas solo deben utilizarse en casos específicos, por ejemplo, tablas de relación o entidades débiles. Si una tabla X con clave compuesta necesita ser referenciada desde otra tabla Y, deberá generarse un campo clave en X al inicio de la misma como “idX” y generar un índice único en los campos que la identificaban.

Otros Campos

Todo campo que represente un nombre o descripción, se colocara inmediatamente después de los campos clave, y se nombrara como a la tabla a la que pertenece, en singular. Ejemplo : tabla materiales => idMaterial , material.

Algunos campos que representan datos, de acuerdo a su representación conceptual en el ámbito del negocio, deberá prefijarse de la siguiente manera:

Números: num (ejemplo: Número de nota =>numNota)

Fechas: fecha (ejemplo: Fecha de compra => fechaCompra)

Códigos: código (ejemplo: Código de producto => codigoProducto)

Los campos booleanos deberán nombrarse de acuerdo al estado correspondiente al valor 1/Verdadero/True de los mismo.

Ejemplo: autorizado, oculto, vigente.

Los campos de relación (foreing keys, claves foráneas) deben nombrarse de la misma manera que los campos clave (usando el nombre de la tabla a la que hacen referencia).

Ejemplos: tabla personas => idTipoDocumento, idEstadoCivil

Procedimientos Almacenados

El estándar utilizado es el siguiente:

USP_+<Nombre identificado del procedimiento>, todo en letras mayúsculas.

Funciones

El estándar utilizado es el siguiente:

UFN_+ <Nombre identificado del procedimiento>, todo en mayúsculas.

Desencadenadores (triggers)

El estándar utilizado es el siguiente:

TB_+<Alias Tabla>_+ <nombre del trigger>, todo en letras mayúsculas.

Donde Alias Tabla son por lo general las 3 o 4 primeras letras del nombre de la tabla a la cual pertenece el trigger.