Nomenclatura

El idioma por defecto a la hora de dar sentido funcional al nombre de clases, variables, constantes, etc. será una mezcla entre la nomenclatura tradicional en inglés y la nomenclatura funcional adoptada.

Resumiendo, aquella codificación que por estandarización y/o aceptación se pueda escribir en inglés se mantendrá así por convenio, casos como *insert*, *update*, *delete*, *create*, *retrieve*, *list*, *set*, *get*, *newInstance*, *Delegate*.

Para la parte funcional se utilizará castellano, por lo tanto, la nomenclatura de los métodos será: *getListEmpresa* en sustitución de *getListCompany* o *insertBanco* en lugar de *insertarBanco*.

Paquetes

Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales. El paquete base queda definido como es.gobcantabria, en este paquete no se definirá ninguna clase.

Se tendrá, así mismo, otro nivel extra dentro del paquete definido como el nombre del proyecto o del módulo (Ej. es.gobcantabria.xxxx).

Si existiera una parte común a varios de estos módulos, el nombre de los paquetes comenzarán por: *es.gobcantabria.common*

La estructura en árbol de los paquetes siguientes a partir de este último se define como sigue: (presuponiendo que se trata de una aplicación web multicapa)

- bussines
  - dao
  - domain
  - service
  - helper
  - exception
- util
- web
  - controller
  - model
  - view

Nombres de Interfaces

Los nombres de interfaces utilizarán el sufijo **Interface** y estarán compuestos por palabras con la primera letra en mayúscula (CamelCase). Se debe evitar el uso de abreviaciones que dificulten la comprensión del código.

Ejemplo: *ConexionInterface*, *ComponenteTablaInterface*

Nombres de clases

Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas (CamelCase).

Debemos intentar mantener los nombres de clases simples y descriptivos.

Debemos usar palabras completas y evitar acrónimos y abreviaturas (se permiten DAO, DTO, URL, HTML, etc.).

Si la clase cumpliese algún patrón determinado o tuviese una funcionalidad específica es recomendable definirlo en el nombre.

Paquete	Funcionalidad	Nombre
bussines.dao	Data Access Object (Interface)	<nombre>DAO
bussines.dao.impl	Data Access Object (Implementation)	<nombre>DAOImpl
bussines.exception	Excepciones	<nombre>Exception
bussines.service	Service	<nombre>Service
bussines.helper	Helper	<nombre>Helper
bussines.dto	Data Transfer Objects	<nombre>DTO
util	Clases de constantes.	<scope>Keys <nombre>Keys
web.controller	Controller	<nombre>Controller
web.filter	Filter	<nombre>Filter
web.model	Model	<nombre>Model
web.listener	Listener	<nombre>Listener

Nombres específicos de gestiones

Cuando se trata de gestionar una entidad determinada (Ej. Usuario) se definen los nombres de clases, demás ficheros implicados con las siguientes reglas:

Clase: *<<FuncionalidadGenerica>><<Entidad>><<Especificación de Clase>>*

Ejemplo: Usuario
UsuarioAction, *FindUsuarioAction*

Métodos

Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (lowerCamelCase).

No se permiten caracteres especiales.

El nombre ha de ser lo suficientemente descriptivo, no importando a priori la longitud de este.

Variables

Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos, con la salvedad de que aquí sí importa más la relación entre la regla mnemónica y la longitud del nombre.

Ejemplo:

- Correctos: *diaCalculo*, *fechaIncoporacion*
- Incorrectos: *dC*, *DCal*, *fI*, *FI¿*

Se evitará en la medida de lo posible la utilización de caracteres especiales, así como nombre sin ningún tipo de significado funcional.

Las excepciones son las variables utilizadas en bucles for, para esos casos se permite utilizar i, j, k, l y siempre en ese orden de anidamiento.

El primer bucle siempre será el que tenga la variable i como iterador. (Esta variable se definirá para el bucle en cuestión).

Constantes

Los nombres de constantes de clases deberían escribirse todo en mayúsculas con las palabras separadas por subrayados ("_"). Todas serán declaradas como *public static final*

public static final String PROPERTY_URL_SERVICIO = "urlServicio";

Estilo de codificación

Comentarios

Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se tiene, por tanto, que evitar referencias al diseño funcional de la misma.

El uso abusivo de los comentarios es desaconsejable, principalmente por el trabajo extra necesario para su correcto mantenimiento. Es preferible rediseñar el código para una mejor compresión de este.

Estándar de Codificación JEE 13ESTÁNDAR DE CODIFICACIÓN JEE

Se tienen que evitar el uso de caracteres especiales dentro de los comentarios, así como el uso de cajas u otro tipo de gráfico creado mediante códigos ASCII.

La estructura de los diferentes tipos de comentarios y su uso general se presenta en la clase base de codificación.

Comentarios de Documentación (JavaDoc)

Como norma es obligatorio proporcionar un comentario de documentación por cada clase / interface, método, propiedad o constante creado.

Son obligatorios los siguientes comentarios de documentación:

Comentario de la clase / interface:

- Prescripción genérica de la clase y su responsabilidad.
- Autor.

Todas las variables tipo *private* o *protected* han de ser obligatoriamente comentadas.

Reglas generales a la hora de escribir comentarios de documentación.

- Siempre se escribe en tercera persona.
- Los caracteres especiales tales como tildes y eñes se han de codificar con su código HTML correspondiente.
- Las descripciones siempre deberían empezar por un verbo.

En cuanto a los tag's utilizados el orden de estos es el siguiente:

- @***param*** seguido del nombre del parámetro e indentada la descripción del mismo. Usualmente esta descripción será una frase corta que comienza definiendo el tipo del parámetro.

@param alturaCaja Entero que define la altura de la caja en píxeles

@param longitudCaja Flotante que define la longitud de la caja en píxeles

- @***return*** este tag no aparece para aquellos métodos que retornan *void*. Por lo demás se comporta como el tag anterior
- @***throws*** Descripción breve de la posible causa de la excepción.
- @***see*** Su uso queda restringido en cuanto a cantidad de los mismos como con el atributo @***link***.

Los demás tags permitidos (@*since*, @*serial*, etc..) el uso es menos común y por lo tanto no se define una manera de utilizarlos.

Usar el atributo <*code*> para las palabras reservadas de java, nombres de clases, métodos, interfaces, propiedades, argumentos y ejemplos de código.

El uso del atributo *@link* tendrá que ser mínimo, para evitar llenar el documento de enlaces. Como norma general solo se incluirán links cuando la referencia sea necesaria y sólo en la primera aparición de esta.

Para distinguir entre dos métodos con el mismo nombre, pero diferentes parámetros de entrada se utilizarán el nombre del método seguido por los argumentos de este entre paréntesis.

Declaraciones

Para la declaración de las variables se utiliza una declaración de cada vez y no se permiten dejar variables locales sin inicializar salvo en el caso de que sean propiedades de un objeto bean.

La codificación correcta sería:

public static Integer entero = new Integer(0);

La declaración de las variables locales a una clase, método o bloque de código se realizan al principio de este y no justo antes de necesitarse la utilización de la variable.

La única excepción a esta regla son las variables que gestionan los bucles *for*.

Las variables de avance de bucles *for* no podrán ser modificadas de ninguna manera fuera de la propia sentencia del bucle.
La duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase se tiene que evitar.

Sentencias

Normas básicas son:

- Una sentencia por la línea de código.
- Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un *if*.

La declaración de los bucles for serán usualmente de la forma:

for (int i = 0; i < condicion; i++)

Son obligatorias las tres condiciones del bucle *for*: inicialización, condición de finalización y actualización del valor de la variable de avance.

La variable de avance del bucle nunca podrá ser modificada dentro del propio bucle.

Buenas prácticas

Constantes

Como norma general todas las constantes numéricas no deberían codificarse directamente, salvo la excepción de -1, 0 y 1.

Propiedades

El acceso/modificación de las propiedades de una clase (no constantes) siempre mediante métodos de acceso get/set.

La asignación de variables / propiedades no podrá ser consecutiva.

Variable1 = variable2 = "hola mundo" No válido

No utilizar el operador asignación en sitios donde se pueda confundir con el operador igualdad. Ni dentro de expresiones complejas.

Métodos

Como norma general no se debe acceder a un método estático desde una instancia de una clase, debemos utilizar la clase en sí misma.