ESTÁNDAR DE CODIFICACIÓN

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:

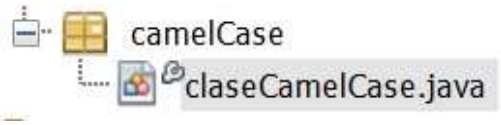
insert, *update*, *delete*, *create*, *retrieve*, *list*, *set*, *get*, *newInstance*, *Delegate*.

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

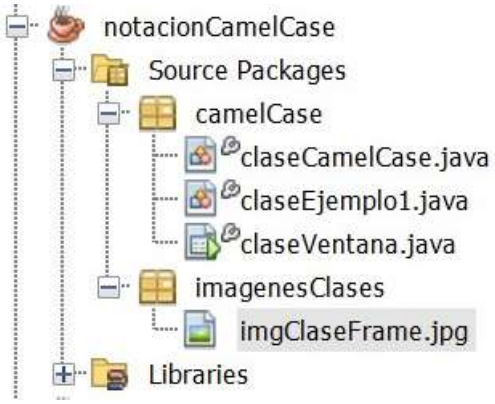
Por defecto todos los paquetes se escribirán en notación CamelCase y sin utilizar caracteres especiales. El paquete base queda definido como, en este paquete no se definirá ninguna clase.

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



Si existiera una parte común a varios de estos módulos, el nombre de los paquetes comenzará por:

La estructura en árbol de los paquetes siguientes a partir de este último se define como sigue:



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.

```
public interface ejemploInterface {
    //Atributos
    //Métodos
}
```

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.

```
public class ejemploClase {
    //Atributos
    //Métodos
}
```

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 con la primera letra de cada palabra interna en mayúsculas (CamelCase). No se permiten caracteres especiales. El nombre del método suele ser un verbo o una frase verbal.

```
public void solicitarNumeros(){
    //Instrucciones;
}
```

Indentación

El sangrado (también conocido como indentación) deberá aplicarse a toda estructura que esté lógicamente contenida dentro de otra. El sangrado será de un tabulador. Es suficiente entre 2 y 4 espacios. Para alguien que empieza a programar suele ser preferible unos 4 espacios, ya que se ve todo más claro.

Las líneas no tendrán en ningún caso demasiados caracteres que impidan que se pueda leer en una pantalla. Un número máximo recomendable suele estar entre unos 70 y 90 caracteres, incluyendo los espacios de sangrado. Si una línea debe ocupar más caracteres, tiene que dividirse en dos o más líneas. Para dividir una línea en varias, utiliza los siguientes principios:

- Tras una coma.
- Antes de un operador, que pasará a la línea siguiente.
- Una construcción de alto nivel (por ejemplo, una expresión con paréntesis).
- La nueva línea deberá alinearse con un sangrado lógico, respecto al punto de ruptura.

Variables

Los nombres de las variables tanto de instancia como estáticas recibirán la notación de camel case, a continuación, se muestran algunos ejemplos.

Por ejemplo:

```
int entero;
double decimal;
char caracter;
```

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, como se puede ver en el ejemplo).

Por ejemplo:

```
for (int i = 0; i < 10; i++) {
    //implementación del codigo
}
```

Constantes

Los nombres de constantes de clases deberán escribirse en la notación en mayúsculas. Todas serán declaradas como *public static final* como se muestra en el ejemplo.

```
public static final int EJEMPLOCONSTANTE = 10;
```

Comentarios

Todos los ficheros fuente deben comenzar con un comentario en el que se lista el nombre de la clase, descripción, fecha de creación y fecha de actualización. Como se muestra en el ejemplo:

```
/**********************************************************************************
 *
 *Nombre de la clase: ApliEjemplo
 *
 *Descripción:Ejemplo de los comentarios
 *
 *Fecha de creación:28 de octubre a las 8:30 pm
 *
 *Fecha de actualización:28 de octubre a las 10:00P pm
 **********************************************************************************/
```

Comentarios de implementación:

Delimitados por /*…*/, son para comentar nuestro código o para comentarios acerca de una implementación particular.
Los comentarios deben contener sólo información que es relevante para la lectura y entendimiento del programa. Evitar cualquier comentario que pueda quedar desfasado a medida que el código evoluciona.

Se utilizarán los comentarios de una línea, que llevara los siguientes requisitos:

- Un comentario de una sola línea debe ir precedido de una línea en blanco.
- Pueden aparecer comentarios cortos de una única línea al nivel del código que siguen.
- Siempre el comentario se colocará en la primera línea.

Por ejemplo:

```
for (int i = 0; i < 10; i++) { /*Esta funcion realiza...*/
    ...
}
```

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 vean.
La codificación correcta sería:

```
public void ejemplo()
 {   //variable local ejemplo
     int ejemplo = 0;
     ejemplo = ejemplo + 5;
     System.out.println("ejemplo de declaracion local: " + ejemplo);
 }
```

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

```
if (expresion ) {
// Bloque 1
} else {
// Bloque 2
}
```

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

```
for (int i = 0; i < 10; 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.

Propiedades

El acceso de las propiedades de una clase (no constantes) siempre mediante métodos de acceso get/set.
La asignación de variables / propiedades no podrá ser consecutivas.

```
Variable1 = variable2 = "hola mundo";//No aplicar de esta manera
```

Utilizar el operador de asignación en sitios donde se pueda confundir con el operador de igualdad, siempre y cuando hacer comentarios de la asignación para poder identificarla.