**Estándar de codificación Java**

|  |  |
| --- | --- |
| **Propósito** | **Guiar la codificación en el lenguaje Java** |
| **Descripción del programa.** | Todo programa debe tener una descripción que indique su propósito:   * Debe contener el propósito del programa, nombre de los autores que lo iniciaron y lo modificaron por última vez y fecha en la que se creó y se realizó la última modificación del programa. |
| **Formato de descripción del programa.** | /\*\*  \* Proposito: Propósito del programa.  \* Autor(s): Nombres de las personas que inician el desarrollo.  \* Fecha creacion: Fecha en la que se crea el programa.  \* Modificado por: Nombre de la última persona que modifica el programa.  \* Ultima modificacion: Fecha en la que se modifica el programa.  \*/ |
| **Descripción de métodos, clases, propiedades** | Sobre toda clase, método o propiedad debe tener su respectivo summary.   * Clases: Debe contener una descripción y el nombre del autor. * Métodos: Debe contener una descripción, si tiene parámetros, retorno o excepciones deben ser documentadas también. * Propiedades: Debe contener una descripción. |
| **Formato de descripción para método.** | /\*\*  \* Descripción funcional del método.  \* @params Nombre parámetros: descripción del parámetro.  \* @return tipo de retorno: Si el método tiene retorno Se debe indicar el tipo de retorno.  \* @throws tipo exception : Si el método tiene excepciones, se deben exponer.  \*/ |
| **Formato de descripción para clases.** | /\*\*  \* Descripción de la clase.  \* @author Nombre del autor.  \*  \*/ |
| **Formato de descripción para propiedades** | /\*\*  \* Descripción de la propiedad.  \*/ |
| **Escritura de paquetes** | * Los paquetes deben tener estructura lowerCamelCase. |
| **Ejemplo de paquete** | uniandes.ecos.conceptosAvanzados.modelo |
| **Clases** | * Debe ser sustantivo que represente parte del negocio. * Deben seguir la estructura CamelCase. * Deben tener su modificador de acceso. * Toda clase debe estar en su propio archivo .java. El archivo será llamado tal como se llama la clase. |
| **Ejemplo de clase** | public class ArchivoEstadistico |
| **Identificadores** | Se deben usar nombres descriptivos para todo nombre de variable, contantes, funciones y demás identificadores.  Seguir este estándar:   * Todo identificador debe comenzar en minúscula. * Se debe usar la estructura lowerCamelCase. * Deben tener su modificador de acceso. |
| **Ejemplo de identificadores** | private Dato datoSiguiente; |
| **Comentarios** | Los comentarios se deben hacer:  - En partes del código donde se identifique la necesidad de clarificar lógica. – Se deben redactar de tal forma que indiquen el propósito a nivel dinámico y el comportamiento dentro del código.  - Los comentarios permitidos son doble slash (//) y (/\* \*/) |
| **Líneas vacías** | Use líneas vacías para separar métodos y propiedades. |
| **Identación** | * Los inicios de corchetes deben estar sobre la misma línea en la cual se está declarando. * Los cierres de corchetes deben comenzar en una nueva línea, los cuales se idéntan con el código que están cerrando. |
| **Ejemplo de Identación** | if(args.length < 1) {  throw new Exception("No tiene argumentos");  } |
| **Sentencias lógicas separadas.** | El código debe ser escrito de tal forma que toda sentencia sea escrita en nueva línea.  Igualmente no se debe poder agregar código al cierre de corchetes (}), al cierre de corchetes solo es válido los comentarios. |
| **Ejemplo sentencias lógicas separadas.** | Ejemplos incorrectos:   * public void calcular() { int suma = 1+1;   }  En el caso anterior la operación de suma debe estar en una nueva linea.   * int suma = 3 + operando; division = suma / divisor;   En el caso anterior la operación de división debe estar en una nueva linea.  Ejemplos correctos:   * public void calcular() {   int suma = 1+1;  }   * int suma = 3 + operando;   division = suma / divisor; |
| **Manejo de corchetes.** | Todo condicional debe tener los corchetes de “{}” para indicar cuando se inician y terminan sus operaciones lógicas. Además de toda sentencia vacía. |
| **Ejemplo para el manejo de corchetes.** | |  |  | | --- | --- | | Ejemplo incorrecto: | Ejemplo correcto: | | * if (limite > 5)   limite += 1; | * if (limite > 5) {   limite += 1;  } | | * public void method(){} | * public void method(){   } | |
| **Enumeradores** | Se deben escribir los enumeradores con coma separada y salto de línea.  Ejemplo:  public enum TipoEstructura {  Clase,  Interfaz,  Enum  } |
| **Condicionales encadenados** | Todo condicional encadenado debe respetar la línea del cierre de corchete del condicional inmediatamente anterior. (Es válido que en la misma línea este else if, o estén en líneas separadas)   |  |  | | --- | --- | | Ejemplo incorrecto: | Ejemplo correcto: | | * if (limite > 5) {   limite += 1;  }else if {  Limit = 1;  } | * if (limite > 5) {   limite += 1;  }  else if {  } | |