**Coding Standard Template**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DefinitionName: | Estandar Codificación | | Language: | Java |
| Author: | UMBRELLA | | Date: | 13/04/2015 |
|  |  | |  |  |
| Purpose | | | Guía para desarrollar programas en JAVA | | |
| ProgramHeaders | | | Comenzar cada clase con un encabezado | | |
| HeaderFormat | | | /\*  \* Description  \*@author: your name  \*@version:  \*/ | | |
| ListingContents | | | Provide a summary of the listing contents. | | |
| Contents  Example | | |  | | |
| ReuseInstructions | | | * Para describir cómo será usado un método, se empleara Javadoc con los tags que tiene la estructura que se ve en el ejemplo. | | |
| ReuseExample | | | /\*\*  \*Obtener la suma entre dos números enteros  \*@author Javier Virvescas  \*@param a: int  \*@param b: int  \*@return resultado: int  \*@throws e: Exception  \*\*/  publicintsumar(int a, int b)throw Exception{… | | |
| Identifiers | | | Utilizar nombres descriptivos, evitando abreviaturas e identificadores de una sola letra. | | |
| IdentifierExample | | | publicvoidcontarLineas(){… // Así está bien  publicvoidcontLin(){… // Así está mal | | |

|  |  |
| --- | --- |
| Comments | * Los comentarios deben proporcionar información clara y precisa de lo que está realizando en el código. * Utilizar “//” únicamente para comentarios de una sola línea. |
| GoodComment | //Variable que almacena el número de líneas del código |
| BadComment | //Variable que almacena  //un numero |
| MajorSections | * Proporcionar información más detallada sobre algún proceso que se está realizando. Debe ser igualmente clara y precisa. * Utilizar /\* … \*/ únicamente para comentarios de más de una línea. |
| Example | /\*Recorrer toda la clase java para conocer y determinar cuantas líneas de código se emplearon y cuantos métodos tiene\*/ // Así está bien  /\*Recorrer toda la clase\*/ // Así está mal |
| BlankSpaces | * Dejar un espacio en blanco entre cada uno de los bloques de paquete, imports, definición de clase, parámetros y métodos. * No dejar espacios en blanco dentro de la implementación de un método. |
| Indenting | * Cada línea de código deben estar alineada con su bloque de ejecución. * El primer corchete ({) debe ir en la misma línea de donde va a empezar el bloque de ejecución y el corchete de cierre debe sola en una línea. |
| Indenting  Example | for (inti = 0; i< 10; i++){  int j = i;  if (j%2 == 0){  System.out.println(“Es par”);  System.out.println(“Es par”);  }  else  System.out.println(“Esimpar”);  } |
| Capitalization | * Clases: La primera letra de cada palabra debe comenzar con mayúsculas seguido de minúsculas. * Variables y métodos: La primera palabra debe empezar con minúscula y si tiene más de una palabra su primera letra debe empezar con mayúscula seguido de minúsculas. * Constantes: Se deben definir en mayúscula y separando las palabras con raya al piso (\_) |
| CapitalizationExample | Ejemplo Clases:  publicclassProgramaPrincipal{… // Así está bien  publicclassprograma\_principal{… // Así está mal  Ejemplo Variables y Métodos:  intnumeroLineas; // Así está bien  publicvoidcontarLineas() // Así está bien  floatnumeroconteo; // Así está mal  Ejemplo Constantes:  publicstatic final int NUMERO\_PI = 3.1416 // Así está bien  publicstatic final stringconstante\_No = “NO” // Así está mal |