**SDS**

Tabla de contenido

[1. Especificaciones de diseño de software 2](#_Toc415817940)

[1.1 Estándares de diseño 2](#_Toc415817941)

[1.2 Revisión de diseño 6](#_Toc415817942)

[1.3 Estándares de reusó 8](#_Toc415817943)

[1.3.1 Clase Buscador: 8](#_Toc415817944)

[1.3.2 Clase Contador: 8](#_Toc415817945)

[1.3.3 Clase – Clase 9](#_Toc415817946)

[1.3.4 Clase Método 9](#_Toc415817947)

# 1. Especificaciones de diseño de software

## 1.1 Estándares de diseño

|  |  |  |
| --- | --- | --- |
| **Making Solutions** | **Formato** | **CD01** |
| **Fecha de actualización** | **01/04/2015** |

|  |  |
| --- | --- |
| Propósito | Estandarizar el esquema de codificación en los programas escritos en JAVA |
| Encabezado de los programas | El encabezado de las clases principales debe contener:  Descripción de la clase  Nombre de autor  Fecha en la que se inicia la implementación  Última modificación  Último autor de modificación |
| Formato de encabezado | /\*\*  \* Descripción  \* @author: Nombre de autor  \* Fecha Fecha en la que se inicia la implementación  \* Última modificación: Fecha de la última modificación  \* Modificado por:  \* @param  \* @return  \*/ |
| Lista de contenido | Instrucciones de reusó  Ejemplo de reusó  Identificadores  Ejemplo de identificadores  Comentarios  Buen comentario  Mal comentario si lo viste?  Secciones principales  Ejemplo de sección principal  Uso de mayúsculas y Minúsculas  Ejemplo de uso de mayúsculas y minúsculas |
| Instrucciones de reutilización | - **Formato de declaración:**  - Las declaraciones deben hacerse por línea, no se aceptan varias declaraciones en una línea.  - La declaración de variables debe hacerse siempre al inicio del método.  - La declaración del método debe hacerse en una línea  - Luego de la palabra clave que describa un ciclo siempre debe existir un espacio.  - La llave que da inicio al método debe encontrarse en el mismo renglón de la declaración del método.  - La sentencia if siempre debe llevar la llave que inicie el método y la llave de cierre.  - Para separar los métodos se utilizará una línea en blanco  - Para separar variables locales y la primera sentencia de un método se utilizará una línea. |
| Ejemplos de Instrucciones de reutilización | - **Declaraciones líneas separadas:**  - String ejemplo;  - String aplicacion;  - String arreglo[];  - **Declaraciones inicio método:**  - public String getSize(){  - int size =0;  - }  - **Declaración de método:**  - public String getSize(){  - **Espacio en palabras clave:**  - for (  - **Ubicación llave inicio:**  - public String getSize(){  - **Sentencia if:**  - if (size > 0) {  - **Separación método:**  - public String getSize(){  - **}**  - public String getDimension(){  - **}**  -  - **Separación variable local:**  - public String getSize(){  - int size=0;  - System.Out.Println("size" + size);  - **}** |
| Identificadores | Los identificadores deben de cierta manera describir la propiedad del atributo. |
| Ejemplo identificador | String nombreIdentificador; /\* Declaración correcta  String nI ; /\* Declaración incorrecta |
| Comentarios | - Los comentarios deben ser claros y concisos  - Debe evitarse comentarios redundantes  - Los comentarios no deben contar con una cantidad excesiva de caracteres |
| Buen comentario | If(nombreIdentificador.equals("correcto")) /\* Es el nombre de identificador aplicado correctamente? \*/ |
| Mal comentario | If(nombreIdentificador.equals("correcto")) /\* valida en estado del identificador |
| Secciones principales | Las principales secciones del programa deben estar precedidas por la línea de descripción de la sección. |
| Ejemplo | /\*\*  \* Ejecuta el conteo de líneas de una clase de acuerdo a estándar definido  \*  \* @param archivo Archivo donde ejecutará el conteo de líneas  \* @param url Ubicación del archivo  \*/ |
| Uso de mayúsculas y Minúsculas | - Todas las constantes deben definirse con mayúsculas y si cuentan con varias palabras deben la primera Letra mayúscula.  - Los nombres de atributos siempre contendrán minúsculas y para varias palabras se usará una letra mayúscula. |
| Ejemplo de uso de mayúsculas y minúsculas | - private static final String PALABRA\_CLAVE\_CLASS = "class";  - private int cantidadDirectorios; |
| Formato de nombres | * Todos los nombres deben tener en cuenta el uso de Mayúsculas y minúsculas especificado en este documento * Los nombres de los componentes del programa deben ser auto descriptivos |
| Mensajes de Error y de sistema | * Todos los mensajes de errores y del sistema deben especificar el componente y el método que lo origina, sumado a una descripción resumida de la causa de origen. |
| Estándar de defecto | * Para la estandarización de defectos se tomará como referencia el estándar de tipificación de defectos de psp. |
| Conteo de LOCS | * Para el conteo de líneas se tomará como referencia el estándar de tamaño de PSP. |

## 1.2 Revisión de diseño

|  |  |  |
| --- | --- | --- |
| **Making Solutions** | **Formato** | **DS01** |
| **Fecha de actualización** | **01/04/2015** |

|  |  |
| --- | --- |
| Propósito | Orientar el proceso de una efectiva revisión de diseño |
| General | Revisar el programa completamente para cada categoría de la lista de chequeo. Es importante no intentar revisar más de una categoría al mismo tiempo.  Tan pronto finalice la revisión de cada paso, registre el item en la columna de la derecha  Complete la lista de chequeo para un programa o unidad del programa antes de revisar el siguiente. |

|  |  |  |  |
| --- | --- | --- | --- |
| Criterio | Descripción del criterio | Cumple | Componente |
| Complete | Verificar que el diseño cubra todos los requerimientos aplicables  -Todas las salidas específicas han sido producidas  -Todas las entradas específicas son suministradas  -Todos los requerimientos incluidos son claros |  |  |
| Lógica | Usar una tabla de Traza, demostraciones matemáticas, o métodos similares para verificar la lógica   * Verificar que la secuencia del programa es apropiada. * La recursión es terminada correctamente. * Verificar que todos los ciclos son apropiadamente inicializados, incrementados y terminados. * Examinar cada sentencia condicional y verificar todos los casos. |  |  |
| Casos Especiales | - Chequee todos los casos especiales  - Asegúrese que todas las variables funcionen con datos, vacíos, mínimos, máximos, negativos y valores de cero.  - Verifique que no exista la posibilidad de desbordamientos o iteraciones fuera de límite.  - Asegúrese que condiciones imposibles son absolutamente imposibles.  -Tenga en cuenta todas las posibles condiciones incorrectas o con error |  |  |
| Uso funcional | * Verificar que todas las funciones, procedimientos o métodos fueron entendidos y utilizados correctamente. * Verificar que todas la abstracciones referenciadas externamente son definidas con precisión |  |  |
| Nombres | Verificar que:   * todos los nombres especiales son claros, definidos y autenticados * que los ámbitos de todas las variables son definidos explícitamente o son evidentes en sí mismos * todos los ítems llamados son usados dentro de los ámbitos declarados. |  |  |
| Estándares | Asegúrese de que el diseño cumple con todas las normas de diseño aplicables. |  |  |

## 1.3 Estándares de reusó

Los siguientes serán la especificación de las clases, métodos y atributos reutilizables.

### 1.3.1 Clase Buscador:

Esta clase es la encargada de la búsqueda y recorrido de la ruta del proyecto a analizar.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | | Buscador | |
| **Attributes** | | | |
|  | **Declaration** | | **Description** |
|  | RUTA\_DIRECTORIO\_FUENTE | | Constante que contendrá la ruta del directorio fuente para este ciclo del proyecto a analizar. |
| **Items** | | | |
|  | **Declaration** | | **Description** |
|  | Buscar Archivos | | Este método se encargará de encontrar la ruta del directorio fuente y recorrer todas las carpetas y subcarpetas con el fin de encontrar en ellas todos los archivos .Jaca |

### 1.3.2 Clase Contador:

Esta clase se encarga de abrir y leer los archivos “.java” encontrados

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | | Contador | |
| **Attributes** | | | |
|  | **Declaration** | | **Description** |
|  | N/A | | N/A |
| **Items** | | | |
|  | **Declaration** | | **Description** |
|  | Ejecutar Conteo | | Método que se encarga de realizar el conteo de las líneas “LOC” de cada clase encontrada. |

### 1.3.3 Clase – Clase

Esta clase es la que lleva los atributos que define la cantidad de líneas que tiene la clase y una lista de la clase métodos.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | | Clase | |
| **Attributes** | | | |
|  | **Declaration** | | **Description** |
|  | Cantidad líneas clase | | Este atributo mostrará la cantidad de líneas LOC que tiene la clase analizada. |
|  | Lista métodos | | Este atributo mostrará la cantidad de métodos que tiene la clase analizada |
| **Items** | | | |
|  | **Declaration** | | **Description** |
|  | Get y Set de Cantidad de líneas clase | | Lee o Escribe la variable cantidad de líneas clase |
|  | Get y Set Lista de métodos | | Lee o Escribe la variable lista de métodos |

### 1.3.4 Clase Método

Esta clase contendrá la información de los métodos de una clase analizada

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Name** | | Método | |
| **Attributes** | | | |
|  | **Declaration** | | **Description** |
|  | Cantidad líneas método | | Este atributo mostrará las líneas que tiene un método |
|  | Cantidad llamadas método | | Este atributo mostrará la cantidad de llamadas que se le realizan a un método específico |
|  | Cantidad puntos decisión | | Este atributo mostrará la cantidad de sentencias “IF”, “FOR” que contiene un método específico. |
| **Items** | | | |
|  | **Declaration** | | **Description** |
|  | Get y Set de cantidad líneas método | | Lee o Escribe la variable cantidad de líneas método |
|  | Get y Set Cantidad llamadas método | | Lee o Escribe la variable Cantidad llamadas método |
|  | Get y Set cantidad puntos decisión | | Lee o Escribe la variable cantidad puntos decisión |