

## CONFIGURACION WOKSPACE ECLIPSE

---

### 1.1. OBJETIVO

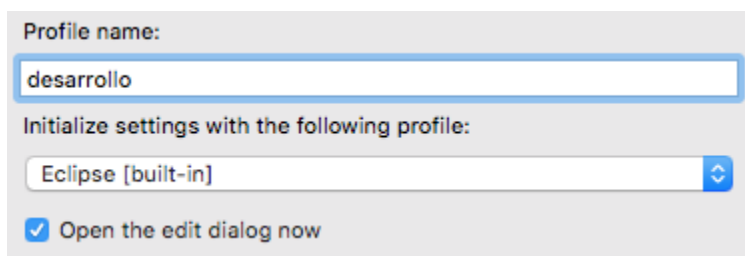
En este documento se presenta la configuración de la herramienta de desarrollo Eclipse que se debe aplicar por parte de cada programador con la finalidad de homologar y automatizar en cierto grado la aplicación de los lineamientos y mejores prácticas de programación en Java.

### 1.2. CREACIÓN DE UN PERFIL DE DESARROLLO

Dentro de las distintas opciones que ofrece Eclipse en cuanto a formateo de código, se encuentra el concepto de “Perfil”. Un perfil asocia a un conjunto de configuraciones y preferencias de formateo de código fuente, el cual puede ser aplicado a uno o más proyectos dentro de un mismo espacio de trabajo.

Antes de iniciar con las configuraciones descritas en esta sección, es conveniente crear un nuevo perfil a partir del cual se realizarán dichas configuraciones. Este perfil será el empleado por todos los proyectos definidos en Eclipse. Para crear un nuevo perfil, seguir las siguientes instrucciones:

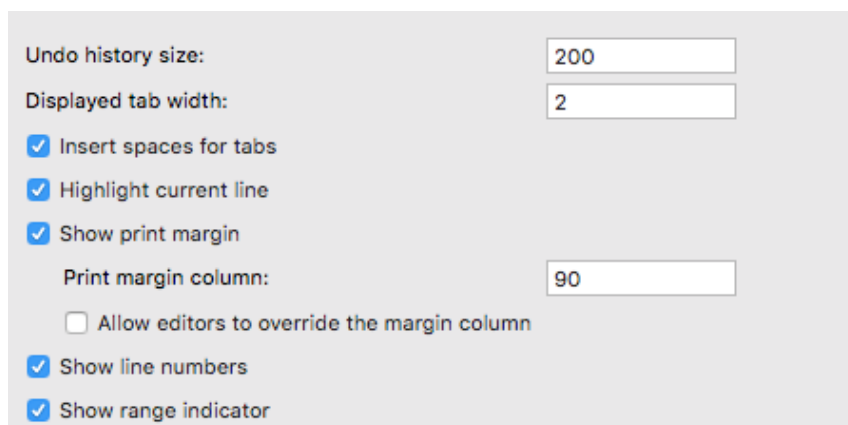
- Dentro de eclipse seleccionar la opción Window-> Preferences del menu seleccionar Java->Code Style ->Formatter
- Hacer clic en New, configurar un nuevo perfil llamado desarrollo como se muestra a continuación:



- Seleccionar Ok.
- Este perfil será empleado para aplicar las configuraciones descritas a continuación que tengan relación con el formato del código fuente.

#### 1.2.1. Configuración general de editores.

- Seleccionar General->Editors->Text Editors Configurar como se muestra a continuación.



Para mostrar el número de líneas, adicionalmente se puede configurar dando clic derecho en el margen izquierdo del editor, seleccionar show line numbers.

#### 1.2.2. Formato de código

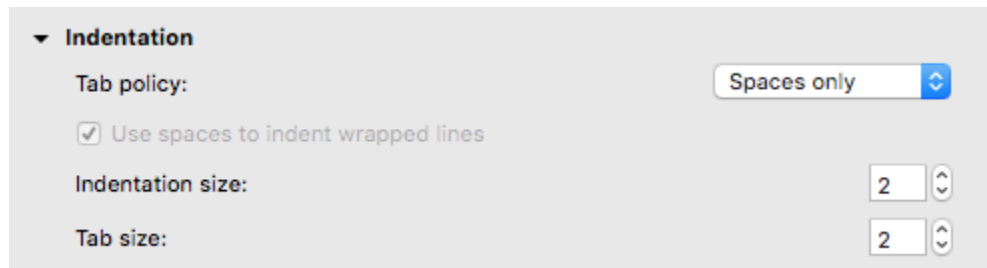
- Dentro de eclipse seleccionar la opción Window-> Preferences del menu

- Seleccionar Java->Code Style->Formatter ->Botón Edit

Aplicar las configuraciones siguientes.

#### 1.2.2.1. Indentación

- Seleccionar la pestaña “Indentation”, configurar como se muestra a continuación:



**Indentation**

Tab policy: Spaces only

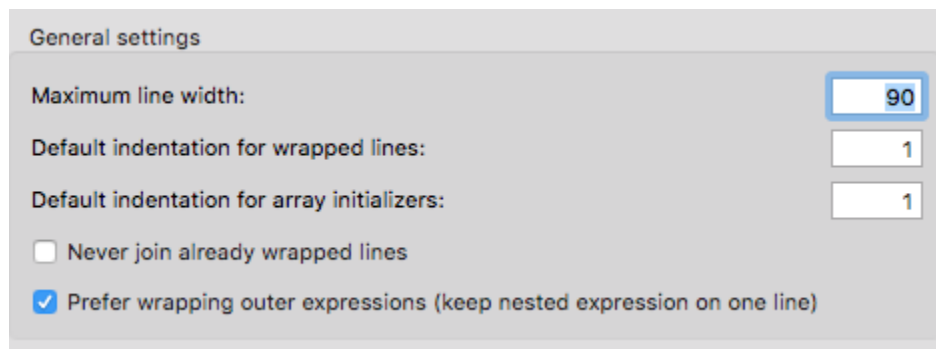
☒ Use spaces to indent wrapped lines

Indentation size: 2

Tab size: 2

#### 1.2.2.2. Longitud de la línea

- Seleccionar la pestaña “Line wrapping” configurar como se muestra a continuación:



**General settings**

Maximum line width: 90

Default indentation for wrapped lines: 1

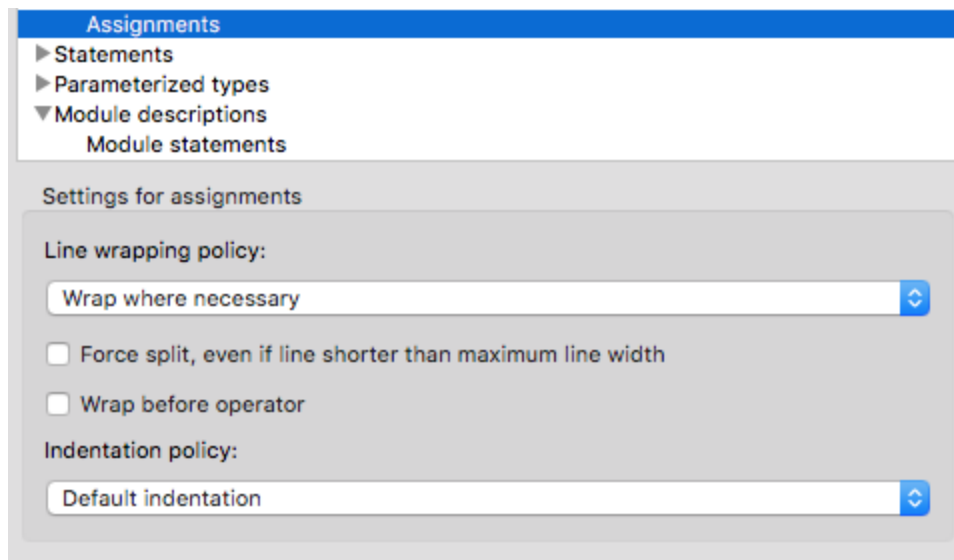
Default indentation for array initializers: 1

☐ Never join already wrapped lines

☒ Prefer wrapping outer expressions (keep nested expression on one line)

- En las opciones de la parte inferior, configurar como se muestra a continuación:

Assignments:



**Assignments**

Statements

Parameterized types

Module descriptions

Module statements

**Settings for assignments**

Line wrapping policy: Wrap where necessary

☐ Force split, even if line shorter than maximum line width

☐ Wrap before operator

Indentation policy: Default indentation

#### 1.2.2.3. Formato de comentarios.

- Seleccionar la pestaña “Comments”, configurar como se muestra a continuación:

The image displays three screenshots of the Eclipse IDE's 'Code Style' settings dialog, specifically the 'Formatter' tab. The first screenshot shows the 'General settings' section with several options checked: 'Enable Javadoc comment formatting', 'Enable block comment formatting', 'Enable line comment formatting', 'Format line comments on first column', 'Preserve white space between code and line comments', and 'Never join lines'. The second screenshot shows the 'Javadoc comment settings' section with options checked for 'Format HTML tags', 'Format Java code snippets inside 'pre' tags', 'Indent Javadoc tags', '/\* and \*/ on separate lines', and 'Remove blank lines'. The third screenshot shows the 'Block comment settings' section with '/\* and \*/ on separate lines' and 'Remove blank lines' checked, and a 'Line width' section where 'Maximum width for comments' is set to 90.

**General settings**

- ☒ Enable Javadoc comment formatting
- ☒ Enable block comment formatting
- ☒ Enable line comment formatting
  - ☒ Format line comments on first column
- ☐ Enable header comment formatting
- ☒ Preserve white space between code and line comments
- ☐ Never indent line comments on first column
- ☐ Never indent block comments on first column
- ☒ Never join lines

**Javadoc comment settings**

- ☒ Format HTML tags
- ☒ Format Java code snippets inside 'pre' tags
- ☐ Blank line before Javadoc tags
- ☒ Indent Javadoc tags
  - ☐ Indent description after @param
- ☐ New line after @param tags
- ☒ /\* and \*/ on separate lines
- ☒ Remove blank lines

**Block comment settings**

- ☒ /\* and \*/ on separate lines
- ☒ Remove blank lines

**Line width**

Maximum width for comments:

☐ Count width from comment's starting position

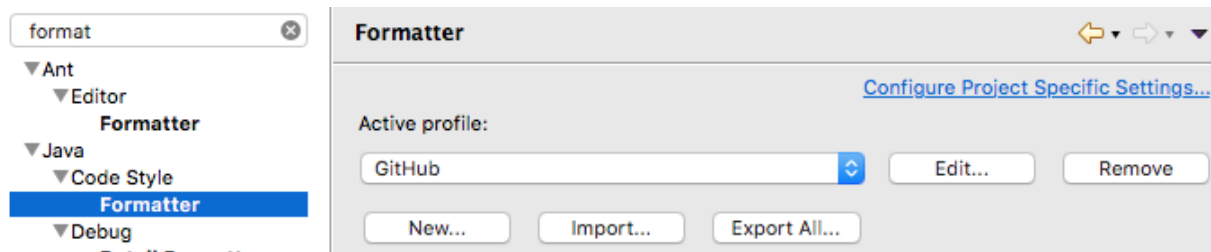
#### 1.2.2.4. Deshabilitar formato automático de código.

En algunos casos muy particulares se desea deshabilitar el formato automático de código ya que ciertas líneas pudieran requerir de algún formato muy particular para poder ser claro y entendible. Bajo estas condiciones, es posible indicarle a Eclipse que no realice el formato de código original. Esto se logra agregando las siguientes líneas (comentarios) al inicio y al final de código que no se desea formatear:

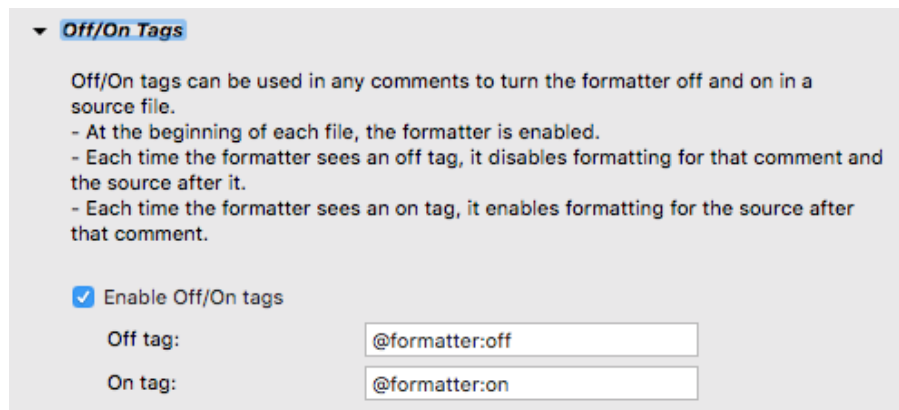
```
//@formatter:off
. . .
//@formatter:on
```

Esta funcionalidad debe ser habilitada. Las siguientes imagenes muestran la forma en la que se activa esa funcionalidad.

- Dentro del menú Java -> Formatter, seleccionar la opción "Edit" como se muestra en la siguiente figura :



- Activar la opción Enable Off/on Tags en el Menú Off/On Tags:



### 1.3. COMENTARIOS Y DOCUMENTACIÓN DE CÓDIGO.

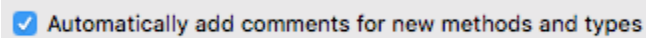
La documentación del código fuente deberá realizarse en inglés. Seguir los siguientes pasos para realizar la configuración.

- Seleccionar la opción Window-> Preferences del menú
- Seleccionar Java -> Code Styles -> Code templates

Aplicar las configuraciones siguientes.

#### 1.3.1. Comentarios por default.

- Al final de la ventana seleccionar la siguiente opción:



#### 1.3.2. Encabezado de la clase.

- Seleccionar Comments->Files, presionar "Edit". Capturar el siguiente texto:

```
/**
 * ${file_name}
 * Creation Date: ${date}, ${time}
 *
 * Copyright (C) The Project *${project_name}* Authors.
 *
 * This software was created for didactic and academic purposes.
 * It can be used and even modified by referring to the author
 * or project on GitHub. If the file is modified, add a note
 * after this paragraph saying that this file is a modified version.
 *
 * The above copyright notice and this permission notice shall be
 * included in all copies or substantial portions of the Software.
 */
```

- Seleccionar Comments->Types, presionar "Edit". Capturar el siguiente texto:

```
/**
 * TODO [Add class documentation]
 * ${tags}
 */
```

- Seleccionar Comments->Fields, presionar "Edit". Capturar el siguiente texto:

```
/**
 * TODO [Add attribute documentation]
 */
```

- Seleccionar Comments->Constructors, presionar "Edit". Capturar el siguiente texto:

```
/**
 * TODO [Add constructor documentation]
 * ${tags}
 */
```

- Seleccionar Comments->Methods, presionar "Edit". Capturar el siguiente texto:

```
/**
 * TODO [Add method documentation]
 * ${tags}
 */
```

**Importante:** Todos los métodos y constructores deben tener este encabezado. Si el código es demasiado simple y no requiere de alguna documentación adicional, se puede omitir la descripción, pero el encabezado se debe conservar.

- Seleccionar Comments->Overriding methods, presionar "Edit". Capturar el siguiente texto:

```
/* See the original documentation of the method declaration.
 * ${see_to_overridden}
 */
```

**Importante:** Todos los métodos que implementen a los métodos de una interfaz, clase abstracta o que sobrescriban un método, deberán tener este encabezado. Si se desea agregar documentación adicional, el encabezado se modifica de la siguiente manera:

```
/**
 * See the original documentatio of the method declaration.
 * <Add here additional documentation>
 * ${see_to_overridden}
 */
```

Observar que se agrega un "\*\*"

### 1.3.3. Configuración del diccionario.

Con la finalidad de reducir faltas de ortografía en los comentarios en inglés, habilitar y configurar el diccionario como se muestra a continuación.

- Seleccionar la opción Window-> Preferences del menú
- Seleccionar General -> Editors -> Text editors ->Spelling

☒ Enable spell checking

Options

- ☒ Ignore words with digits
- ☒ Ignore mixed case words
- ☒ Ignore sentence capitalization
- ☐ Ignore upper case words
- ☒ Ignore internet addresses
- ☒ Ignore non-letters at word boundaries
- ☒ Ignore single letters
- ☐ Ignore Java string literals
- ☒ Ignore '&' in Java properties files

Dictionaries

Platform dictionary: inglés (Estados Unidos)

User defined dictionary: `${workspace_loc}/en-dictionary.txt`

The user dictionary is a text file with one word on each line

Encoding: ☒ Default (ISO-8859-1) ☐ Other: ISO-8859-1

#### 1.3.4. Configuración de detección de comentarios incompletos o faltantes.

- Seleccionar Window-> Preferences del menu
- Seleccionar Java->Compiler->Javadoc , aplicar la configuración que se muestra a continuación.

☒ Process Javadoc comments

Severity levels for problems in Javadoc comments:

Malformed Javadoc comments:

Only consider members as visible as:

☒ Validate tag arguments (@param, @throws, @exception, @see, @link)

☐ Report non visible references

☒ Report deprecated references

Missing tag descriptions:

Missing Javadoc tags:

Only consider members as visible as:

☒ Ignore in overriding and implementing methods

☒ Ignore method type parameters

Missing Javadoc comments:

Only consider members as visible as:

☒ Ignore in overriding and implementing methods

**Tip:** Para agilizar la documentación de elementos, basta con situarse en el elemento que se desea documentar, presionar Alt + Shift + J para agregar un bloque de comentarios.

#### 1.4. ASPECTOS GENERALES DEL ESTILO DE PROGRAMACIÓN

- Seleccionar Window-> Preferences del menu
- Seleccionar Java->Compiler->Errors/Warnings ->Code Style , aplicar la configuración que se muestra a continuación:

Non-static access to static member:	<input type="text" value="Warning"/>
Indirect access to static member:	<input type="text" value="Warning"/>
Unqualified access to instance field:	<input type="text" value="Ignore"/>
Access to a non-accessible member of an enclosing type:	<input type="text" value="Ignore"/>
Parameter assignment:	<input type="text" value="Ignore"/>
Non-externalized strings (missing/unused \$NON-NLS\$ tag):	<input type="text" value="Ignore"/>
Undocumented empty block:	<input type="text" value="Warning"/>
Resource not managed via try-with-resource (1.7 or higher):	<input type="text" value="Warning"/>
Method with a constructor name:	<input type="text" value="Warning"/>
Method can be static:	<input type="text" value="Warning"/>
Method can potentially be static:	<input type="text" value="Ignore"/>

- Seleccionar la opción Potential Programming Problems, configurar como se muestra a continuación.

**▼ Potential programming problems**

Comparing identical values ('x == x'):	Warning
Assignment has no effect (e.g. 'x = x'):	Warning
Possible accidental boolean assignment (e.g. 'if (a = b)'):	Warning
Boxing and unboxing conversions:	Ignore
Using a char array in string concatenation:	Warning
Inexact type match for vararg arguments:	Warning
Unlikely argument type for collection methods using 'Object'	Warning
<input type="checkbox"/> Perform strict analysis against the expected type	
Unlikely argument type for method equals()	Info
Empty statement:	Ignore
Unused object allocation:	Ignore
Incomplete 'switch' cases on enum:	Warning
<input type="checkbox"/> Signal even if 'default' case exists	
'switch' is missing 'default' case: →	Ignore
'switch' case fall-through:	Warning
Hidden catch block:	Warning
'finally' does not complete normally:	Warning

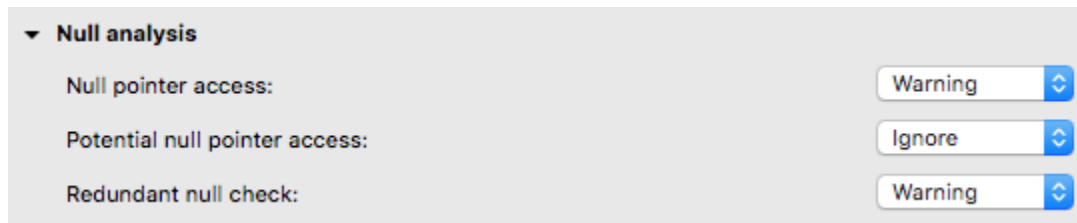
- Seleccionar la opción Unnecessary Code, configurar como se muestra a continuación

**▼ Unnecessary code**

Value of local variable is not used:	Warning
Value of method parameter is not used:	Warning
<input checked="" type="checkbox"/> Ignore in overriding and implementing methods	
Unused type parameter:	Ignore
<input checked="" type="checkbox"/> Ignore unused parameters documented with '@param' tag	
Value of exception parameter is not used:	Ignore
Unused import:	Warning
Unused private member:	Warning
Unnecessary 'else' statement:	Ignore
Unnecessary cast or 'instanceof' operation:	Ignore
Unnecessary declaration of thrown exception:	Ignore
<input checked="" type="checkbox"/> Ignore in overriding and implementing methods	
<input checked="" type="checkbox"/> Ignore exceptions documented with '@throws' or '@exception' tags	
<input checked="" type="checkbox"/> Ignore 'Exception' and 'Throwable'	
Unused 'break' or 'continue' label:	Warning
Redundant super interface:	Ignore



- Seleccionar la opción Null Analysis, configurar como se muestra a continuación



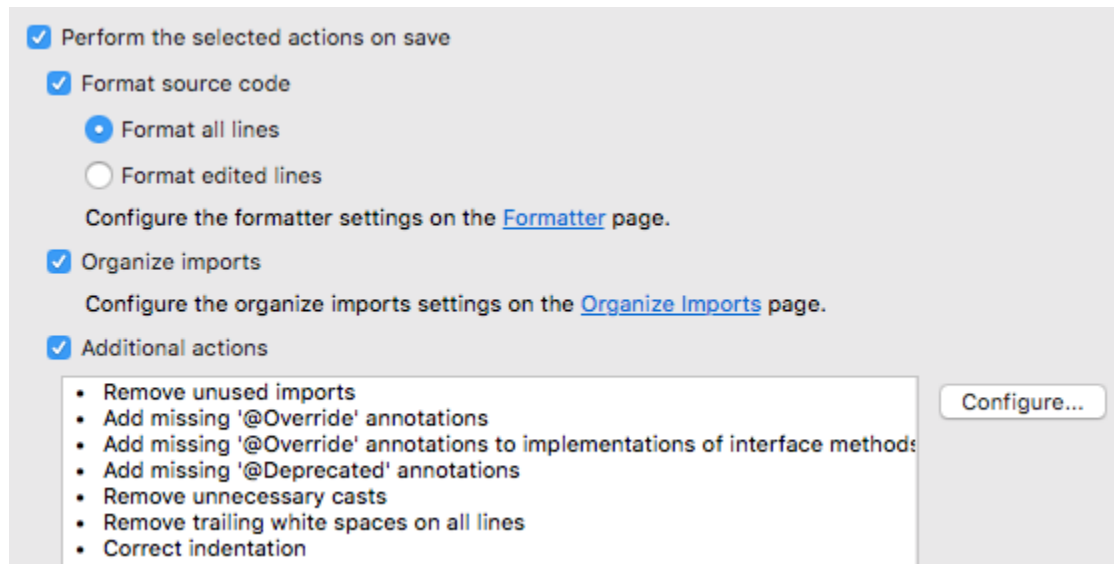
### 1.5. ACCIONES AL SALVAR UN ARCHIVO

Las siguientes acciones serán ejecutadas al salvar un archivo Java:

- Formateo automático de la clase
- Organización de imports
- Adición de llaves para sentencias if/do/while/for que hayan sido omitidas (sentencias con una sola instrucción).

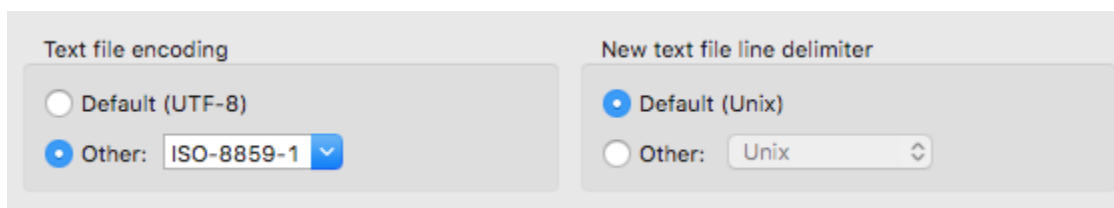
Para realizar esta configuración realizar los siguientes pasos:

- Seleccionar la opción Window-> Preferences del menu
- Seleccionar Java->Editor ->Save Actions,
- Aplicar la configuración que se muestra a continuación, hacer clic en 'Configure'.



### 1.6. CONFIGURACIÓN DEL JUEGO DE CARACTERES

- Seleccionar la opción Window-> Preferences del menu
- Seleccionar General -> Workspace, configurar el juego de caracteres como se muestra a continuación.



### 1.7. CONFIGURACIÓN DE TEMPLATES

El uso de templates permite insertar ciertas líneas de código comunmente empleadas durante el desarrollo de un producto. La configuración de este workspace incluye los siguientes templates personalizados:

- template-debug.xml
- template-logdef.xml

Ambos archivos hacen referencia a instrucciones comunmente empleadas para realizar el manejo de mensajes de log.

El primer archivo permite insertar la siguiente línea de código que contiene la definición de un atributo de tipo `Logger` empleada para el manejo de mensajes a bitácora:

```
private static final Logger log = LoggerFactory.getLogger(CareerPlanServiceImpl.class);
```

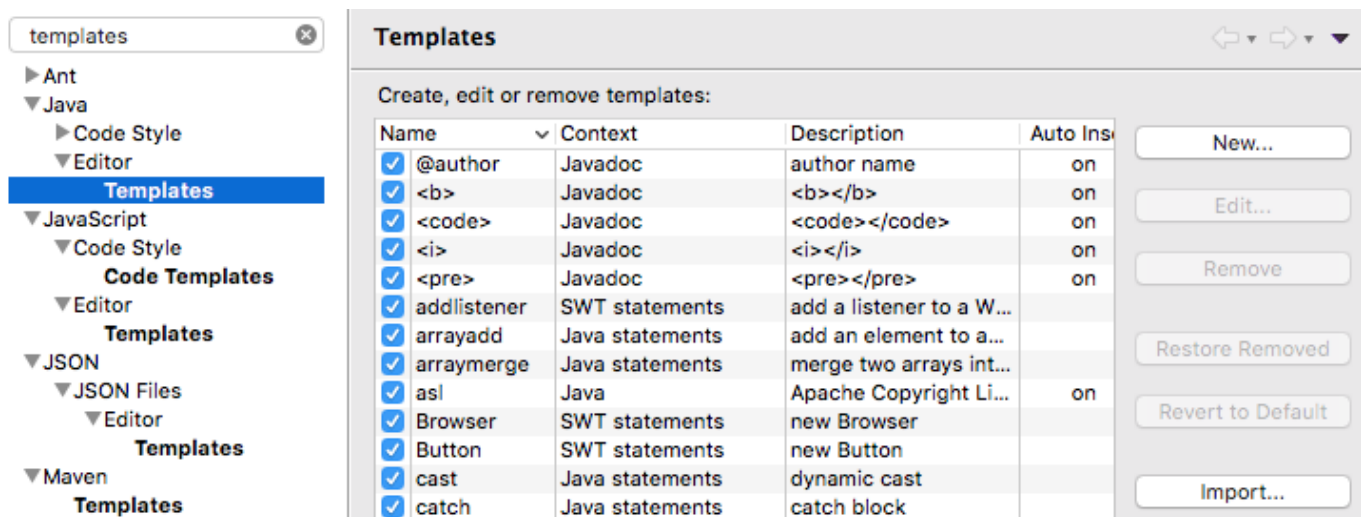
- Por convención a la variable se le deberá asignar el nombre `log`.
- Para activar la inserción de esta línea bastará con escribir `logdef` en el editor de código Java.

El segundo archivo permite insertar las siguientes líneas de código al escribir en el editor debug

```
log.debug("");
```

Se emplea SLF4J para realizar la construcción de estas 2 instrucciones.

Para realizar la configuración de estos 2 archivos, seleccionar la opción **Import** como se muestra en la siguiente figura. Los archivos se encuentran en la carpeta `templates` ubicada el mismo directorio donde se encuentra este documento.



## 1.8. RECOMENDACIONES GENERALES.

Antes de enviar una revisión de código o de realizar un commit, se deberá revisar lo siguiente:

- El código modificado debe estar probado, agregar pruebas en caso de ser necesario.
- El código no debe contener "Warnings". Se deberán eliminar antes de hacer commit.

## 1.9. PLUGINS INSTALADOS

Los siguientes plugins serán instalados en el workspace.

### 1.9.1. Object Aid

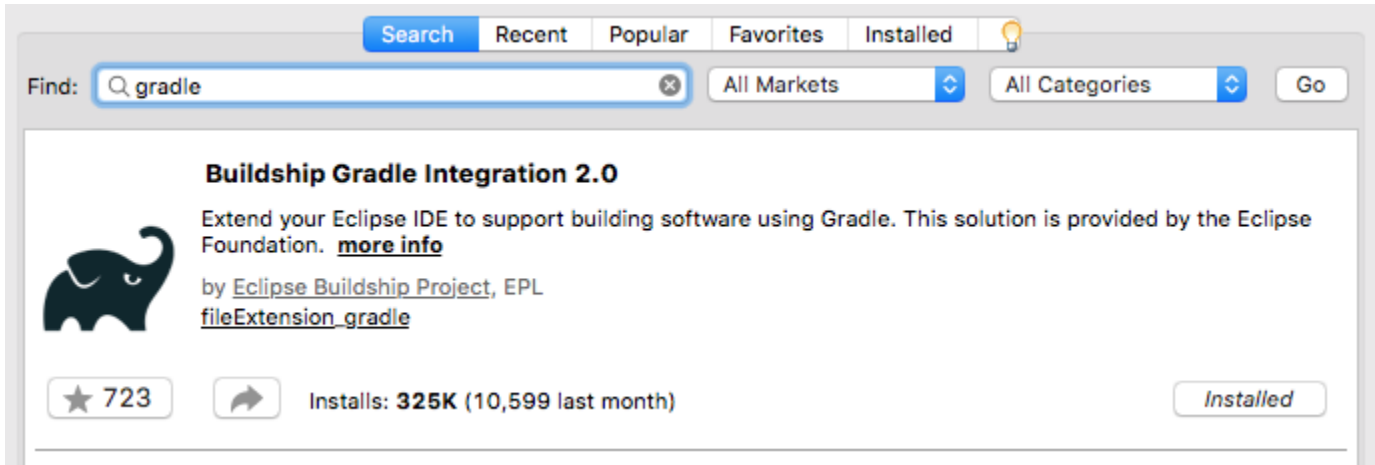
Empleado para realizar diagramas UML: <http://www.objectaid.com/>

Seguir las instrucciones que aparecen en la siguiente página para realizar su instalación: <http://www.objectaid.com/installation>

### 1.9.2. Gradle plugin

Seguir las siguientes instrucciones para instalar el plugin de Gradle, empleado para realizar la integración de Gradle .

- Seleccionar del menú Help-> Eclipse Marketplace
- En el campo de búsqueda escribir 'gradle', presionar 'go'.
- Instalar el plugin que aparece en la siguiente figura: *Buildship Gradle Integration X.x*



- Reiniciar la IDE para que los cambios tomen efecto.

Este documento estará en mejora continua conforme se detecten durante el proceso de desarrollo.