**Inicio**

Antes de empezar a programar, veremos algunas funciones importantes de Eclipse, comenzando por la carpeta oculta "metadata", la cual contiene información importante sobre nuestros proyectos creados, como su configuración, etc.

Empezamos creando una clase "main" en la que irá el código que se ejecutará, y también un "working set," el cual nos permitirá organizar y gestionar un subconjunto de recursos en nuestro workspace, como las vistas. Para crear el working set, seguiremos estos pasos:

1. Vamos a "File" (Archivo) > "New" (Nuevo) > "Other" (Otro).
2. En el cuadro de diálogo "Select a wizard" (Seleccionar un asistente), buscamos "Working Set" y seleccionamos "Working Set" en la lista de resultados.
3. Hacemos clic en "Next" (Siguiente) y luego seleccionamos los recursos (archivos y carpetas) que deseamos incluir en el working set. En este caso, seleccionaremos los recursos dentro del proyecto existente.
4. Le damos un nombre al working set y configuramos las opciones según nuestras preferencias.
5. Hacemos clic en "Finish" (Finalizar) para crear el working set.

A pesar de haberlo creado, si queremos verlo, deberemos hacer clic en los tres puntos que se encuentran encima de las carpetas a la izquierda, seleccionamos "Top Level" y luego elegimos el "Working Set".

También exportamos el Javadoc del proyecto. Para ello, seguiremos estos pasos:

1. Hacemos clic derecho en el proyecto.
2. Seleccionamos "Export" (Exportar) en el menú contextual.
3. En el cuadro de diálogo "Export," expandimos la carpeta "Java" y seleccionamos "Javadoc," luego hacemos clic en "Next" (Siguiente).
4. En la siguiente pantalla, debemos especificar las opciones de exportación del Javadoc:

* **Select the types for which Javadoc should be generated**: Aquí podemos seleccionar las clases, paquetes u otros elementos para los cuales deseamos generar la documentación Javadoc. Puedes seleccionar "Whole project" (Proyecto completo) para generar la documentación para todo el proyecto.
* **Javadoc command**: Dejamos esta opción en su valor predeterminado si tenemos configurado el comando Javadoc estándar en Eclipse.
* **Destination**: Especificamos la ubicación en la que deseamos guardar la documentación Javadoc exportada. Puedes seleccionar una carpeta o un archivo ZIP.
* **Extra Javadoc options**: Si deseamos agregar opciones adicionales para la generación de Javadoc, podemos hacerlo aquí.

1. Hacemos clic en "Finish" (Finalizar) para iniciar el proceso de exportación del Javadoc. Eclipse generará la documentación Javadoc según las opciones especificadas y la guardará en la ubicación que hayamos seleccionado.

Una vez que se complete el proceso de exportación, podremos acceder a la documentación Javadoc generada en la ubicación que especificamos en el paso 4. Podemos ver el Javadoc del proyecto en el navegador.

**Instalación de Plugins**

Otra cosa que haremos es ver cómo podemos instalar plugins para mejorar la experiencia en Eclipse. Uno de los plugins que nos interesa es JAutodoc, el cual nos ayudará con el Javadoc del proyecto. Para instalarlo, seguiremos estos pasos:

1. Vamos al menú "Help" (Ayuda) en la barra de menú de Eclipse y seleccionamos "Eclipse Marketplace..." (Mercado de Eclipse).
2. Se abrirá una ventana de diálogo llamada "Eclipse Marketplace" donde podremos buscar y seleccionar diferentes plugins y extensiones para Eclipse.
3. Buscamos "JAutodoc" y lo descargamos.

**Exportación del Archivo .jar**

También exportamos el archivo .jar del proyecto, el cual nos permitirá comprimir y agrupar múltiples archivos relacionados en un solo archivo. Los archivos JAR se utilizan principalmente para distribuir y ejecutar aplicaciones y bibliotecas escritas en el lenguaje de programación Java. Para hacerlo, seguiremos estos pasos:

1. En el menú contextual, seleccionamos "Export" (Exportar).
2. En el cuadro de diálogo "Select a wizard" (Seleccionar un asistente), expandimos la carpeta "Java" y seleccionamos "JAR file" (Archivo JAR). Luego, hacemos clic en "Next" (Siguiente).
3. Aparecerá el cuadro de diálogo "JAR File Specification" (Especificación del archivo JAR), donde podemos configurar las opciones para la exportación del archivo JAR:

* Select the resources to export: Podemos seleccionar qué recursos (archivos y carpetas) deseamos incluir en el archivo JAR. Por lo general, querrás incluir los archivos compilados de tu proyecto.
* Select the export destination: Especificamos la ubicación donde deseamos guardar el archivo JAR. Puedes utilizar la ubicación predeterminada o seleccionar una carpeta diferente.
* JAR file: Especificamos el nombre del archivo JAR que se generará.
* Options: Aquí podemos configurar opciones adicionales, como incluir archivos de origen o archivos de recursos.

1. Una vez que hayamos configurado las opciones según nuestras preferencias, hacemos clic en "Next" (Siguiente).
2. En la siguiente pantalla, podemos configurar cómo deseamos manejar las clases duplicadas si ya existe un archivo JAR con el mismo nombre en la ubicación de destino. Podemos elegir entre "Replace existing JAR file" (Reemplazar el archivo JAR existente) o "Generate the manifest file" (Generar el archivo MANIFEST.MF). Luego, hacemos clic en "Finish" (Finalizar).
3. Eclipse generará el archivo JAR en la ubicación que especificamos.

Al igual que existen los compiladores, existen los decompiladores, que en vez de compilar el código, recuperan el código original. Esto puede hacer que se revele información no deseada, por lo que para evitarlo existen programas de ofuscación de código, que hacen más difícil el descompilamiento del código.

También, podemos agregar breakpoints a la hora de ejecución. Estas son interrupciones en la ejecución del programa que nos permiten detener la ejecución en un punto específico del código para inspeccionar variables y depurar problemas. Los paneles de tareas ("Task Tags") son etiquetas especiales que se utilizan para resaltar comentarios en el código fuente de manera que sea fácil identificar ciertos tipos de tareas o notas importantes. Algunas de las etiquetas comunes en Eclipse incluyen:

@TODO: Esta etiqueta se usa para señalar tareas que aún no se han completado. Puedes usarla para indicar partes del código que requieren implementación o mejoras.

@FIXME: Se usa para señalar problemas o errores conocidos que deben abordarse en el futuro. Indica que hay algo incorrecto en el código que necesita corrección.

@XXX: Esta etiqueta puede utilizarse para destacar áreas críticas o problemas urgentes en el código.

@BUG: Se usa para marcar áreas que contienen errores que deben solucionarse.

@HACK: Puede indicar que se ha utilizado una solución temporal o un "parche" en una parte del código.

@NOTE: Se utiliza para proporcionar comentarios o notas informativas sobre el código sin necesariamente indicar una tarea pendiente.