

INTRODUCCIÓN A ECLIPSE

Práctica 2

Prácticas Ingeniería del Software

ETS Ingeniería Informática

DSIC – UPV

Curso 2014-2015

Índice

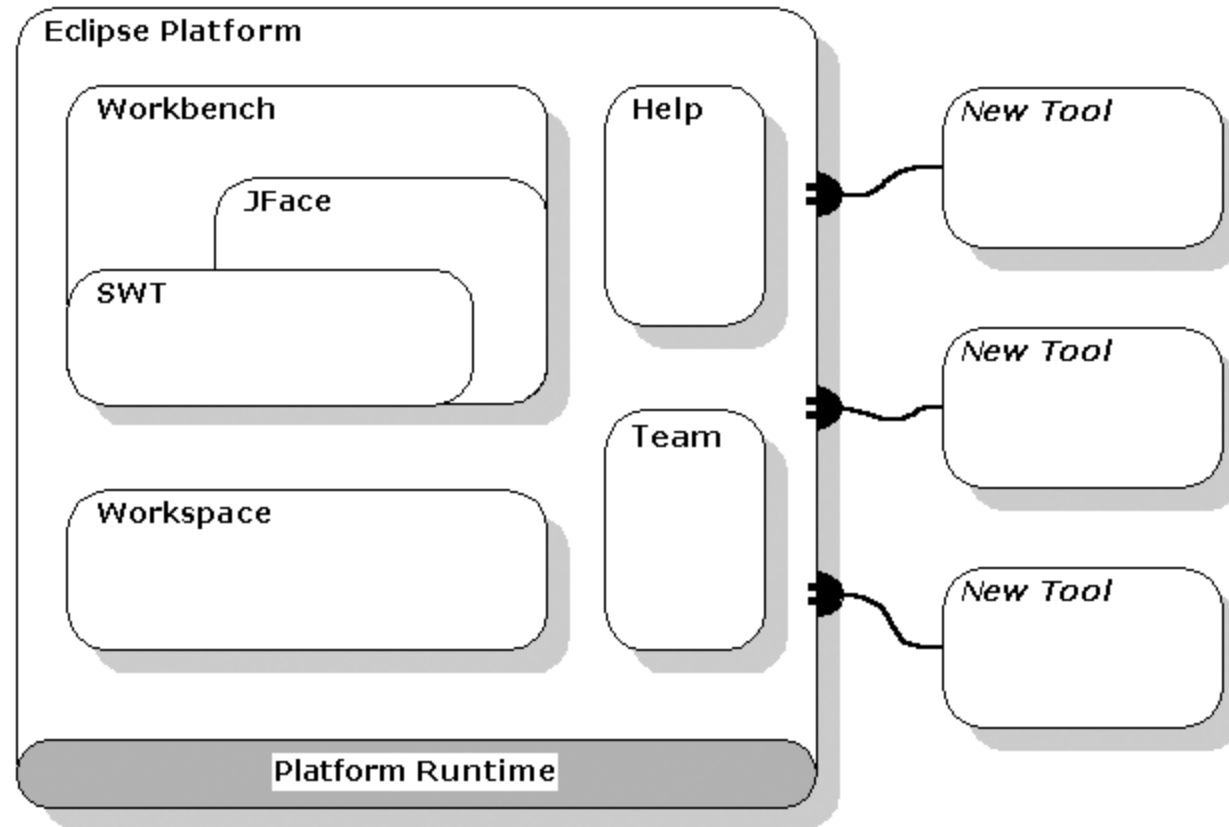
- Introducción a Eclipse.
- La plataforma básica.
- Personalización del entorno.
- Proyectos en Eclipse.
- Instalación de Eclipse.

¿Qué es Eclipse?

- Es una plataforma de desarrollo universal.
 - Libre
 - Código abierto
 - Multiplataforma (win, linux, mac)
 - Extensible
- Para descargar la última versión y la documentación asociada:
 - www.eclipse.org/downloads

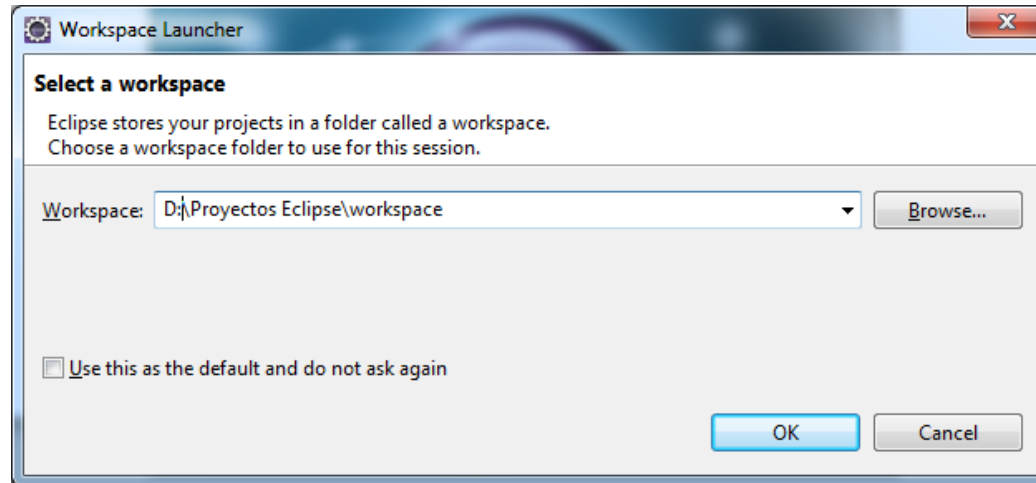
La plataforma básica

- Arquitectura de la plataforma eclipse



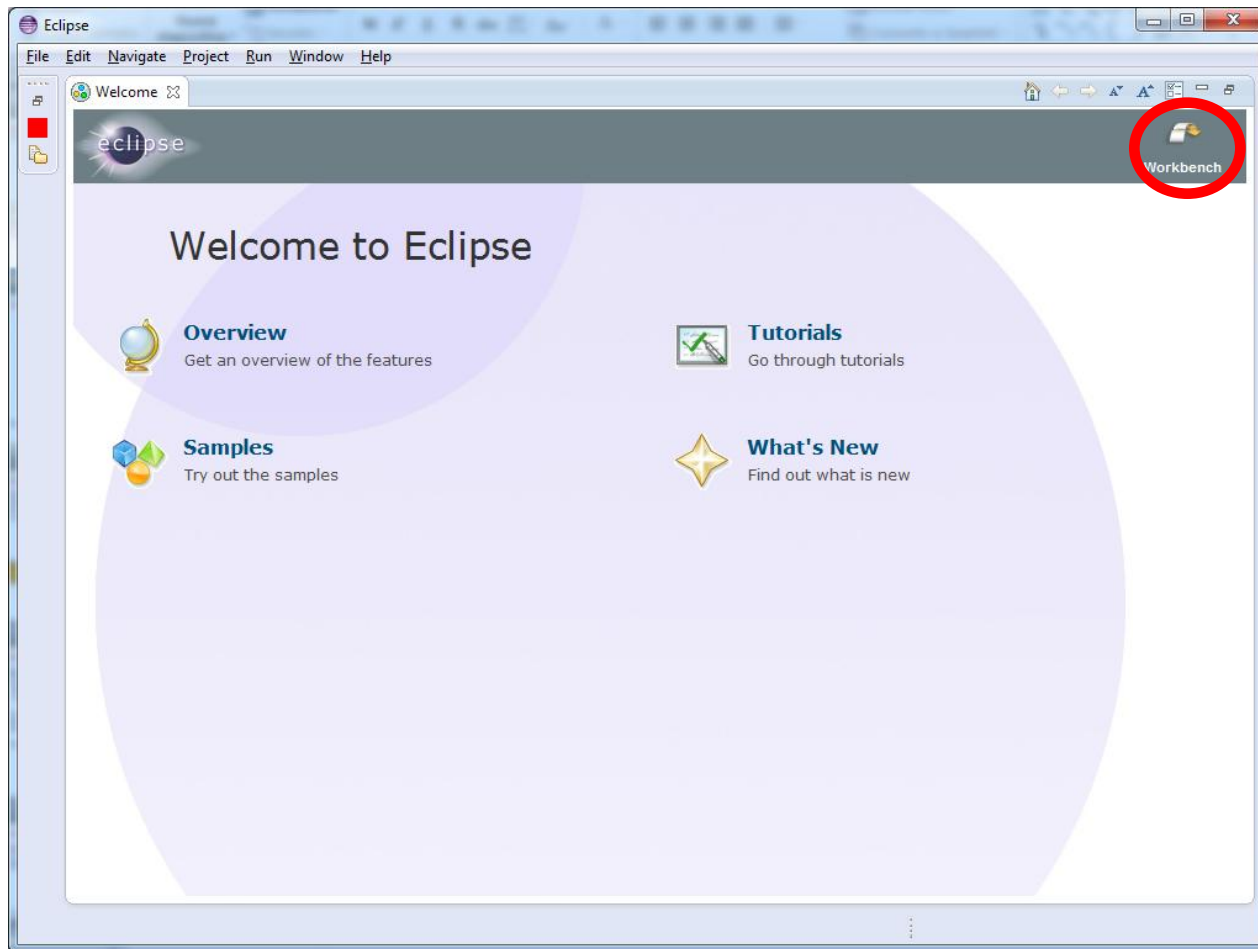
El espacio de trabajo (*workspace*)

- Al iniciar Eclipse se muestra la siguiente ventana.



- El *workspace* contiene el espacio de trabajo en una sesión con Eclipse. En una misma instalación podemos tener diversos espacios de trabajo.
- El espacio de trabajo almacena los diferentes proyectos. Los archivos de un proyecto son accesibles desde el sistema operativo.
- Pulsar **Ok** para acceder al entorno.

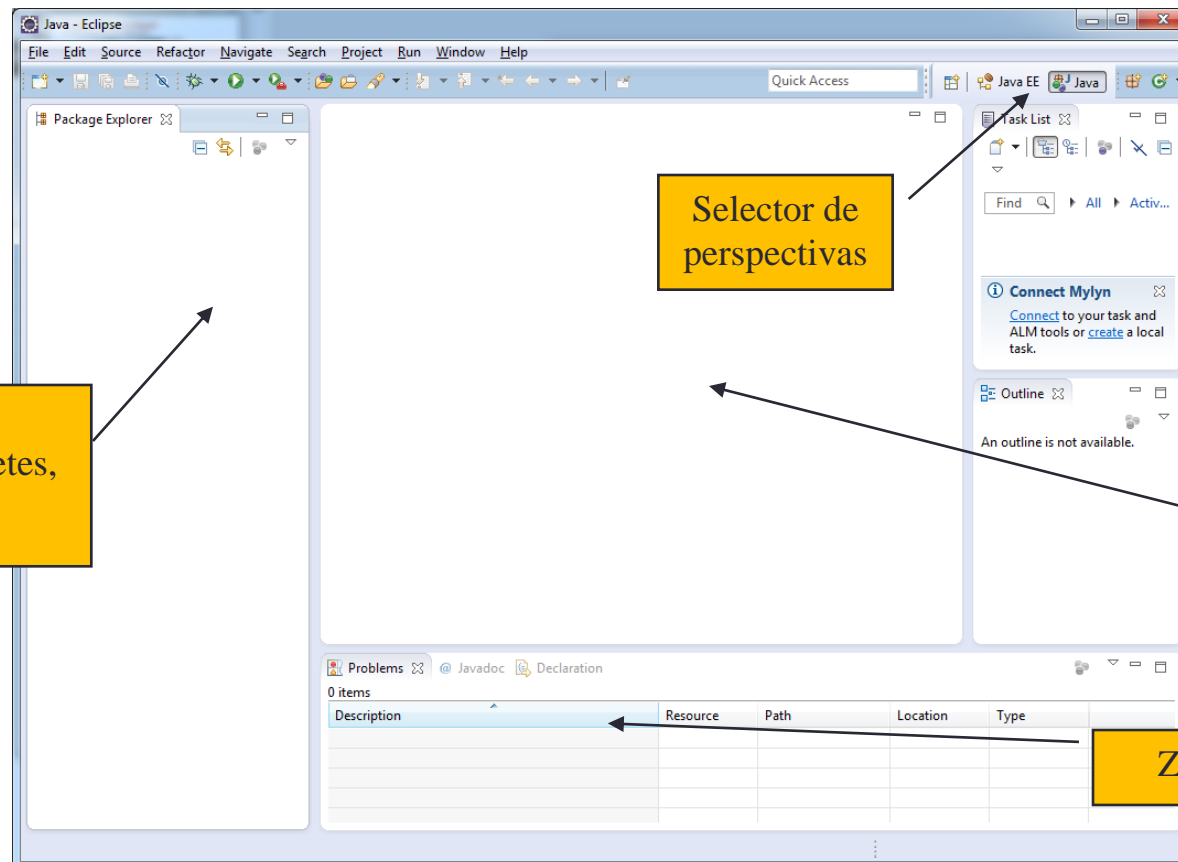
El workbench



- Pulse en la flecha para pasar al workbench

Workbench

- El *Workbench* es la plataforma básica sobre la que se construye la interfaz de usuario de Eclipse.
- El *Workbench* utiliza dos librerías básicas: SWT (widgets nativos) y JFace (API gráfica de alto nivel sobre SWT).
- La interfaz de usuario del entorno (IDE) está dividida en: editores, vistas y perspectivas.

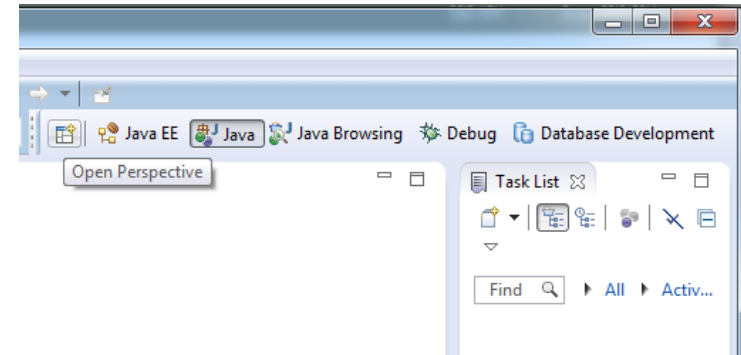
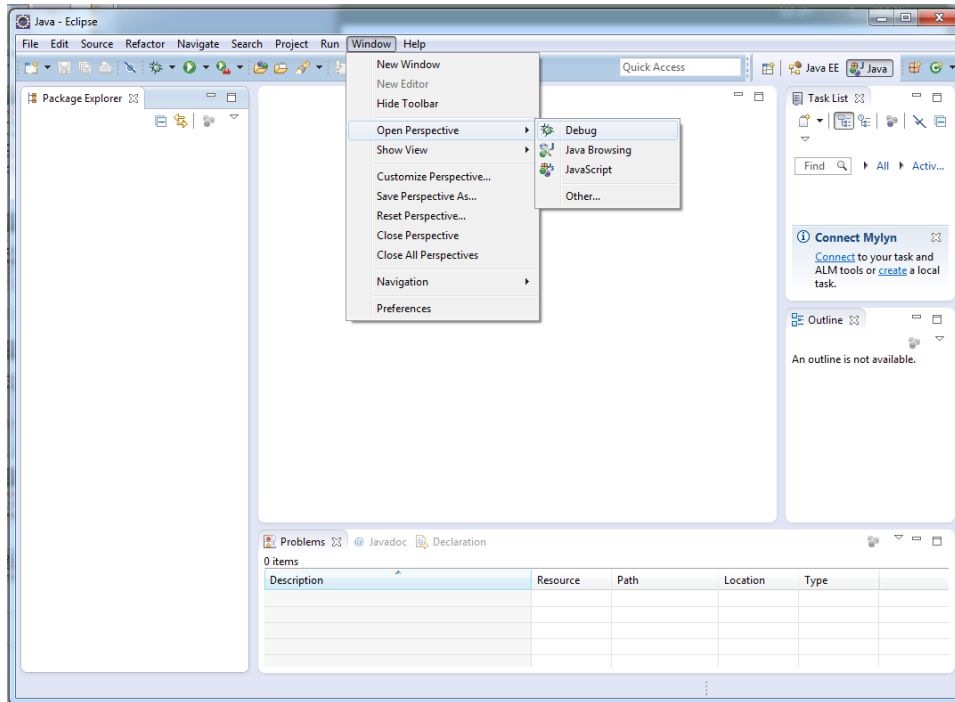


Workbench

- Una perspectiva es una configuración determinada de vistas, menús y editores con un propósito concreto como por ejemplo:
 - Programar en Java (Java)
 - Depurar un programa (Debug).
 - Lanzar consultas sobre una base de datos (SQL Explorer), etc.
- Las vistas son ventanas auxiliares que aparecen en la pantalla, pueden contener desde la consola de ejecución de Java, la lista de errores o una consulta SQL. El contenido de la vista depende de la perspectiva activa.
- Los editores son el núcleo central del IDE y pueden ser textuales o bien gráficos.

Personalización del entorno: perspectivas

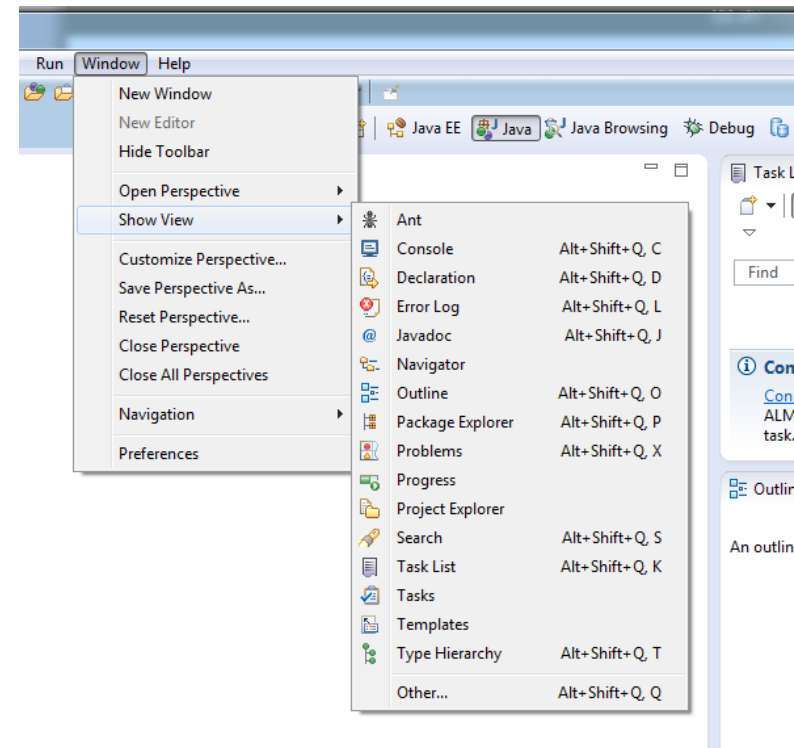
- A una perspectiva se puede acceder mediante la ventana superior derecha o bien mediante **Window -> Open Perspective**.



- Esto permite cambiar la perspectiva activa.

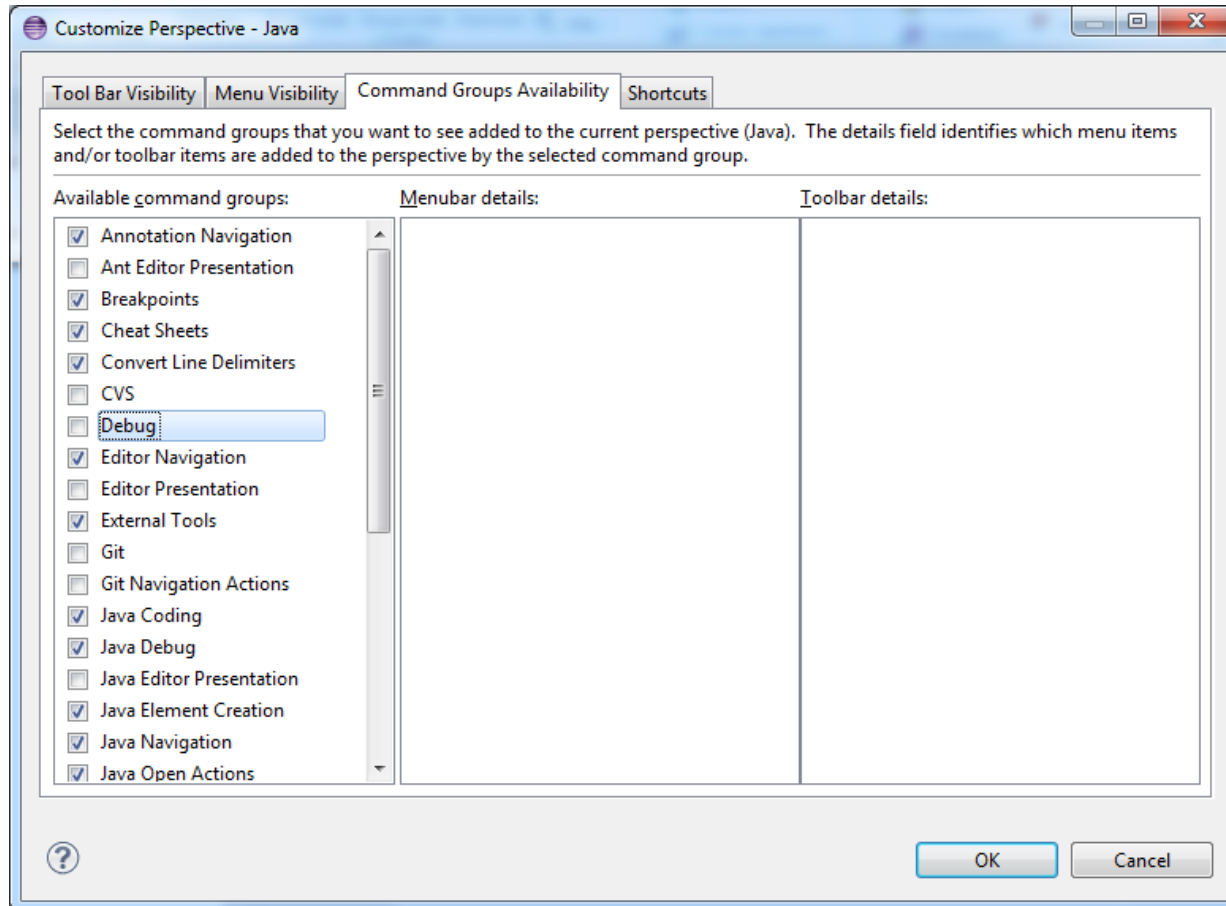
Personalización del entorno: vistas

- Se pueden añadir vistas a las predefinidas por una perspectiva mediante el menú (**Window | Show View**):
 - **Navigator**: vista jerárquica de los recursos (ficheros, carpetas y proyectos).
 - **Help**: acceso a la ayuda.
 - **Tasks**: definición de tareas pendientes.
 - **Problems**: mensajes de error y avisos.
 - **Outline**: lista de los elementos del fichero activo (clases, atributos, métodos, ...)
 - **Properties**: propiedades del elemento activo.
 - **Search**: búsqueda de cadenas de texto y ficheros.
 - **Console**: consola de ejecución.



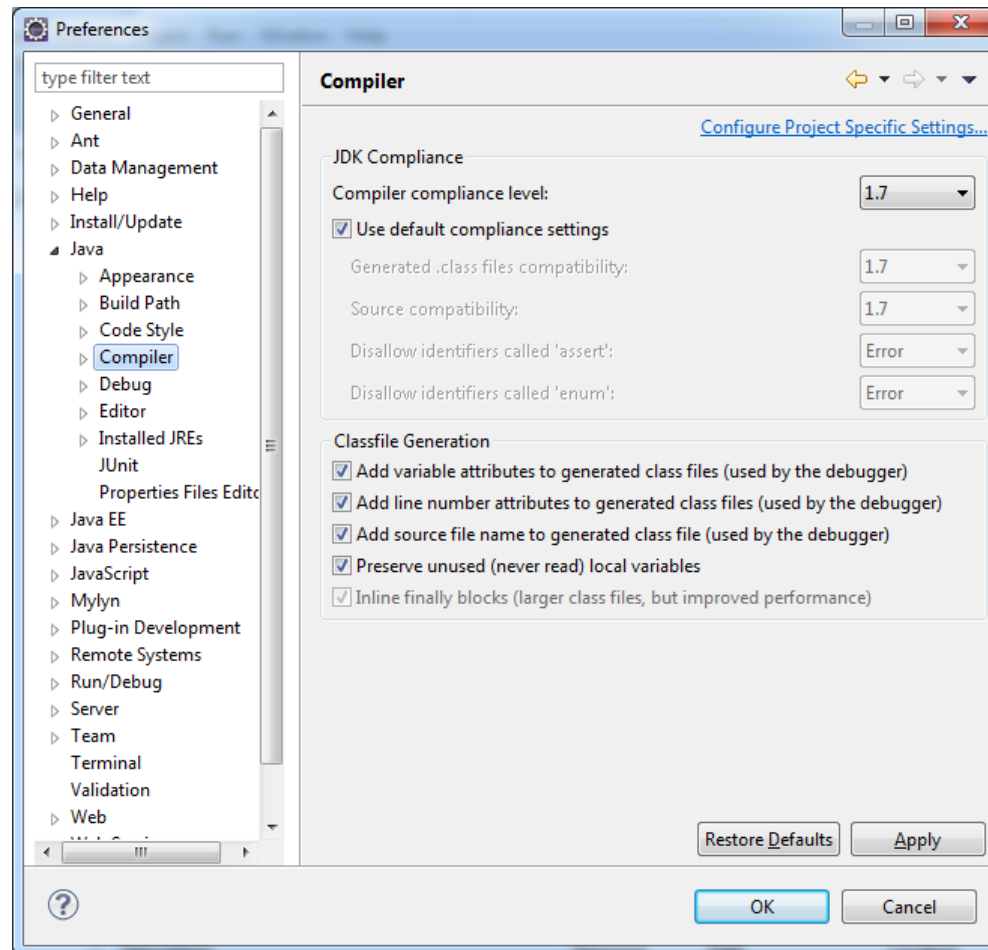
Personalización de menús

- La barra de menús también es personalizable. Mediante el menú «**Window** → **Customize perspective...**» se abre el cuadro de diálogo que permite seleccionar los iconos visibles



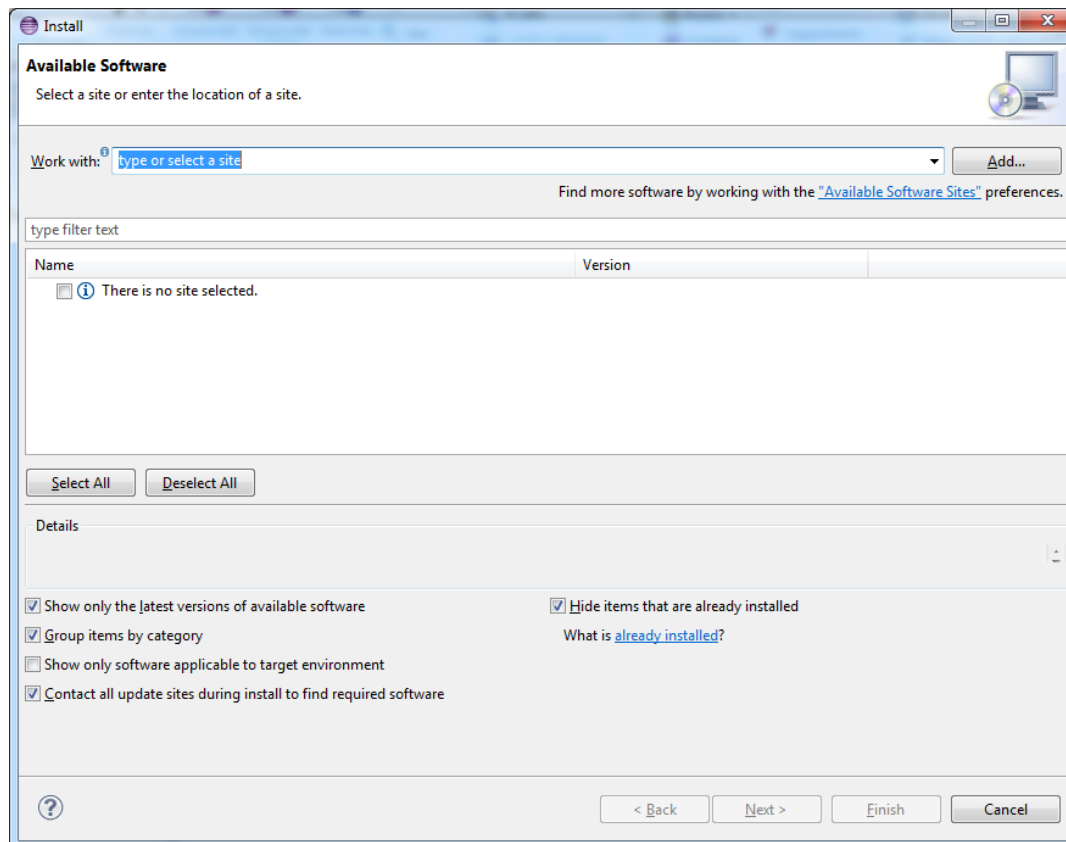
Configuración de Eclipse

- Mediante **Window → Preferences** se accede a todos los aspectos configurables de Eclipse. Los plugins añaden la configuración de sus preferencias al mismo sitio.



Actualizar y añadir funcionalidad a Eclipse

- Las actualizaciones del entorno se hacen mediante asistentes especiales accesibles desde:
 - Help → Check for Updates
 - Help → Install New Software



- Las actualizaciones se pueden obtener de un repositorio remoto o de un repositorio local.

Ayuda en Eclipse

- **Help | Welcome**
 - Acceso a los tutoriales sobre la plataforma y el desarrollo en Java.
- **Help | Dynamic Help**
 - Acceso a la ayuda integrada en el entorno de trabajo como una vista. Se actualiza dinámicamente dependiendo de la vista seleccionada.
- **Help | Search**
 - Presenta los tópicos relacionados con la cadena introducida.
- **Help | Help Contents**
 - Acceso a la ayuda en una ventana separada.
- **Ayuda sensible al contexto:**
 - Pulsando **F1** se muestra la ayuda asociada al elemento activo.

Proyectos Eclipse

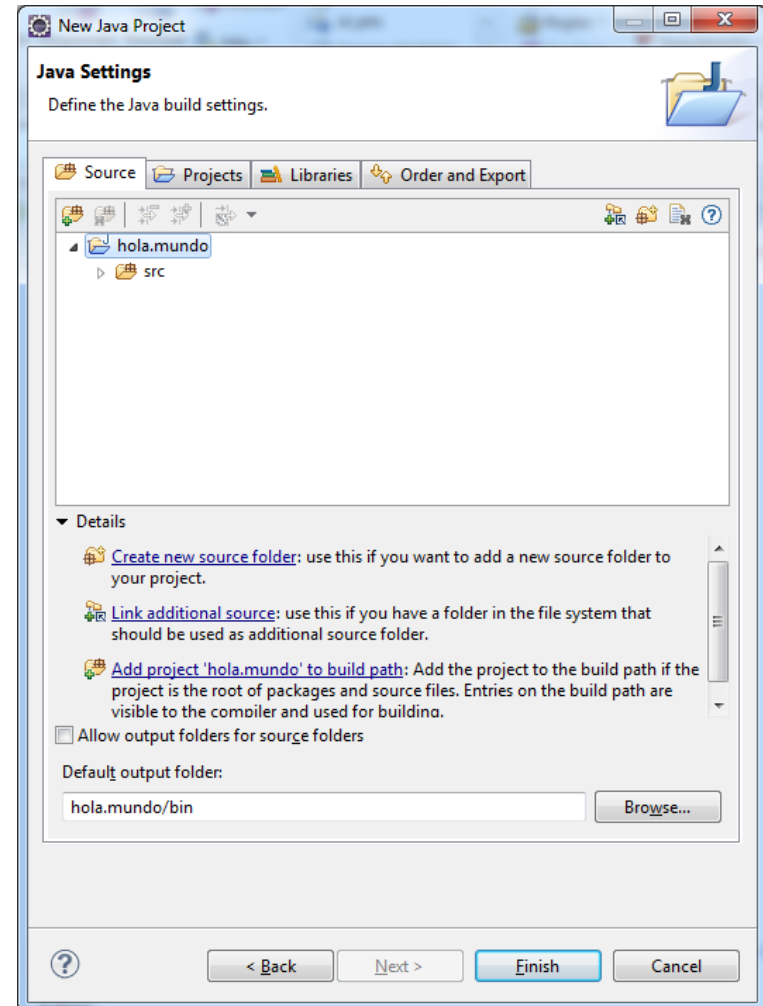
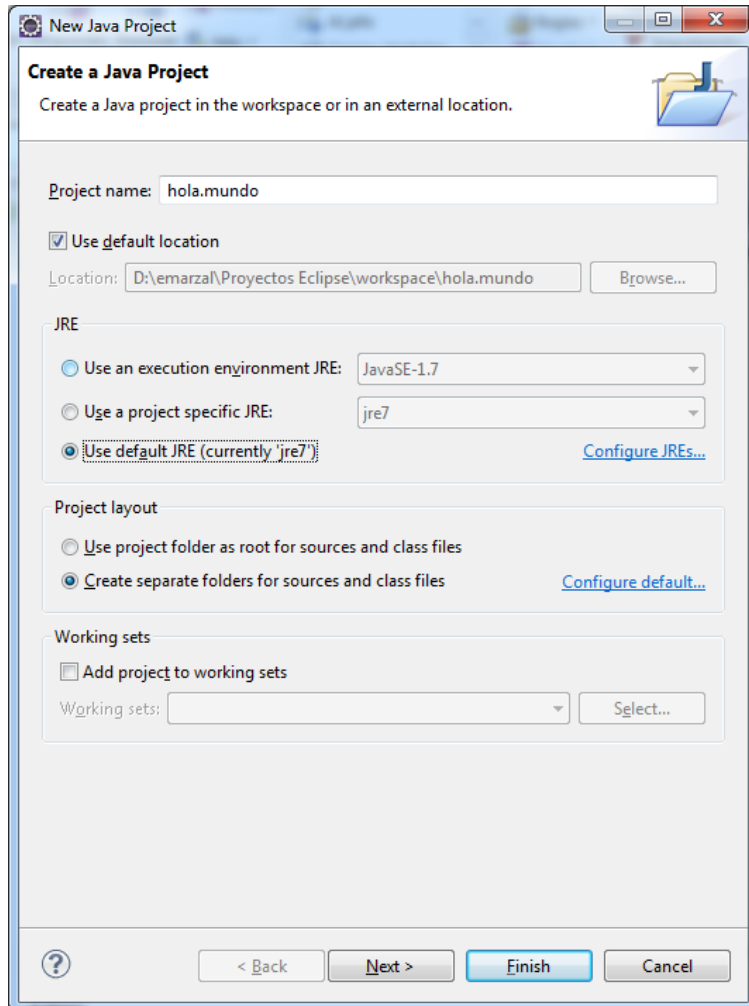
- Tipos de proyectos:
 - **Java Project**, para editar o crear programas Java. Contiene los archivos de código fuente *.java*, documentación y otros recursos.
 - **General Project**, para almacenar documentos y otros archivos, pero no código java.
 - **Plug-in Development Project**, para añadir nuevos módulos a Eclipse.
 - **Eclipse Modelling Framework (EMF)**, para generación de código Java a partir de modelos estructurados.

Preparar el entorno de desarrollo Java

- Preferencias: **Window | Preferences**
 - **Java → Installed JREs**
 - Muestra el JRE (Java Runtime Environment) instalado.
 - **General → Workspace**
 - Marcar la opción **Build automatically**
 - **Java → Build Path**
 - Marcar **Project** como carpeta de fuentes y de salida
 - En **Classpath variables** se definen variables para hacer referencia a ubicaciones que contengan ficheros fuente o jar.
 - **Java → Editor**
 - Marcar **Report problems as you type**

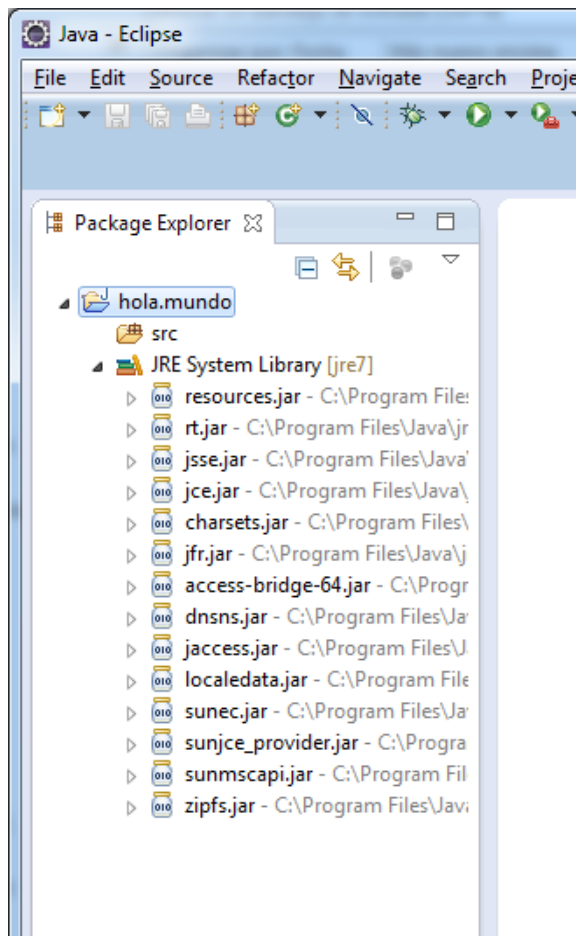
Creación de un nuevo proyecto

- Desde la perspectiva Java utilice el menú **File** → **New** → **Java Project**. Introduzca el nombre del proyecto (hola.mundo) y pulse Next.



Creación de un nuevo proyecto

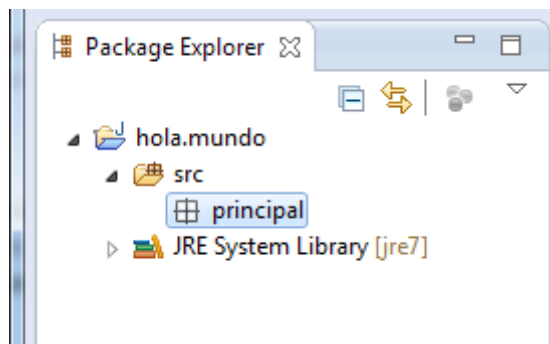
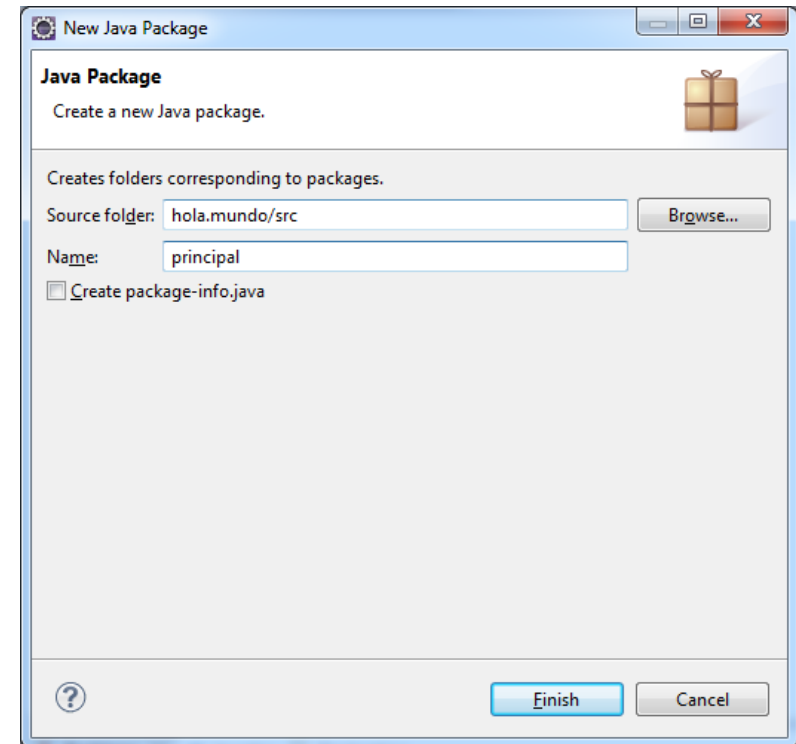
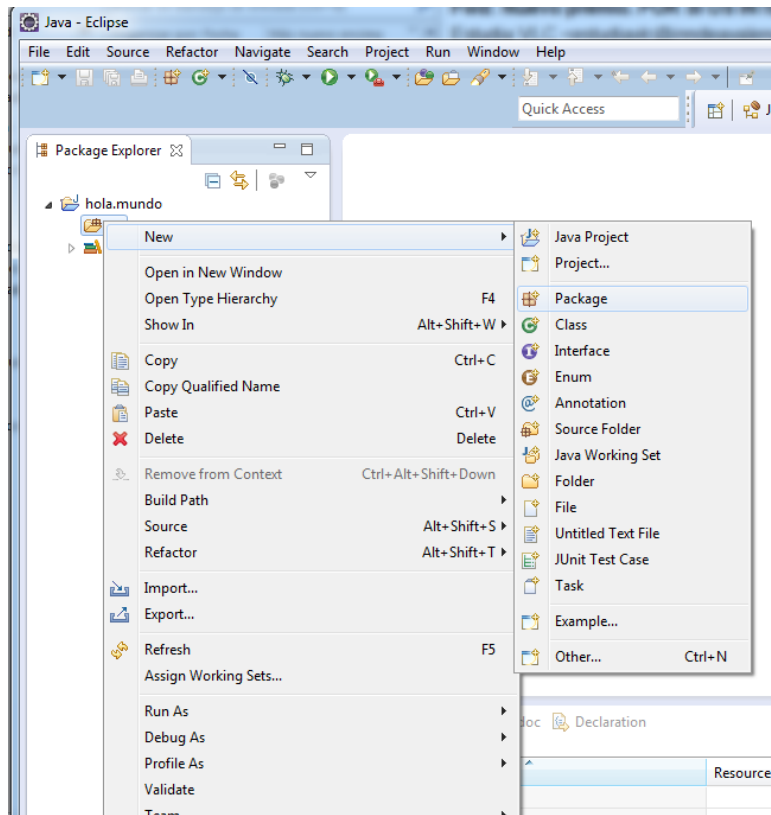
- Por defecto se crea un proyecto vacío con las librerías del JDK incluidas: *resources.jar*, *rt.jar*, *jss.jar*, *jce.jar*, *dnsns.jar*, *QTJava.zip*, *sunjce_provider.jar*, *sunmscapi.jar* y *sunpkcs11.jar*. Los jar adicionales que aparecen en la figura corresponden a un entorno con plug-ins adicionales.



Crear elementos de Java

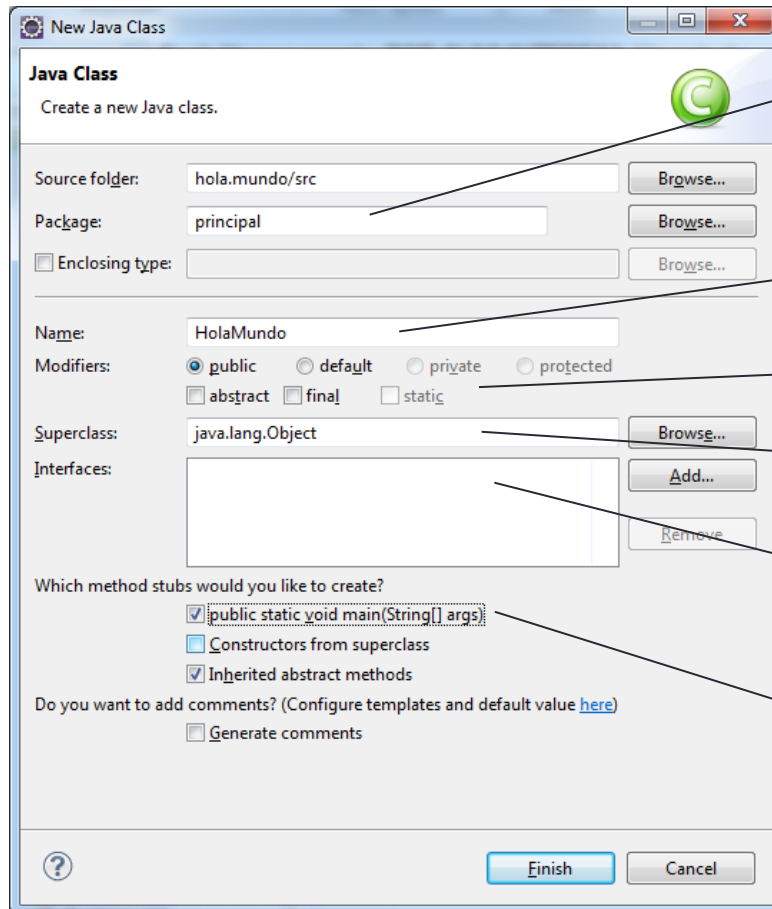
- Con la perspectiva Java activa.
- **File | New** (o pulsando con el botón derecho del ratón en una carpeta del Explorador de Paquetes)
 - **Package**: las clases java se organizan en paquetes, cada paquete se corresponde con una carpeta
 - **Class**: archivos *.java* de código fuente
 - **Interface**: grupo de signaturas de métodos
 - **Enum**: tipo enumerado
 - **Source Folder**: carpeta que contendrá los archivos de código fuente
 - **File**: archivos de texto (p.e. archivos de notas)
 - **Folder**: carpetas para organizar archivos dentro del proyecto

Añadir un paquete



Añadir una clase

- Sobre el paquete recién creado y utilizando el menú contextual use **New** → **Class**. Teclee como nombre HolaMundo y marque public static void main(String[] args)



Paquete que contiene la clase. Si está en blanco se toma como paquete por defecto la carpeta del proyecto.

Nombre de la clase

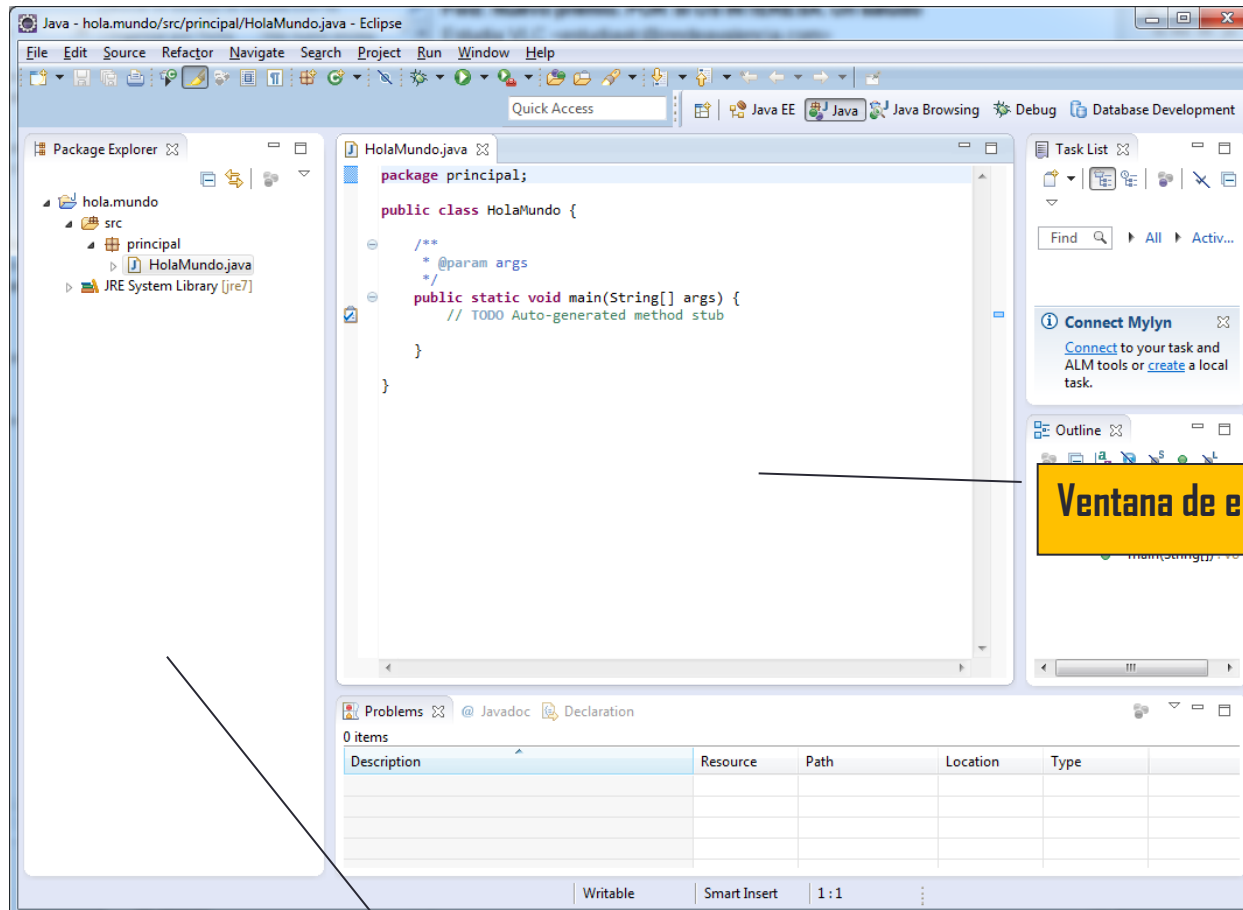
Modificadores de clase

Clase 'padre'

Interfaces que implementa

Marcar si es la clase principal de la aplicación

Añadir una clase



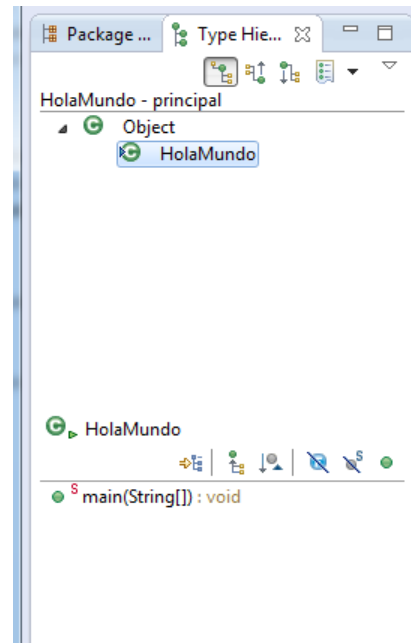
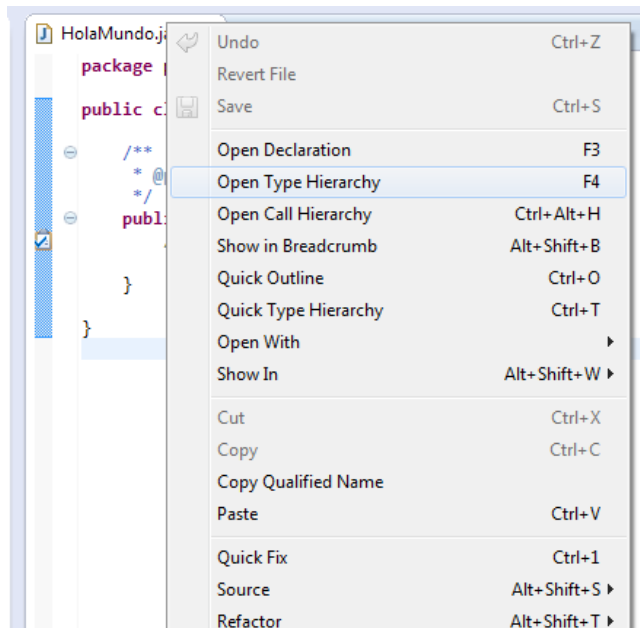
Ventana de edición

Explorador de paquetes:

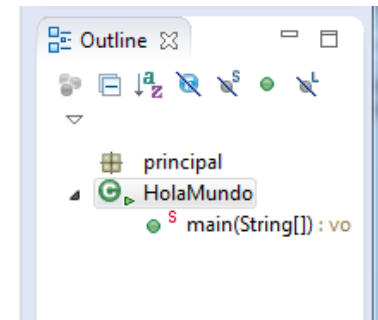
Acceso a ficheros, clases, métodos, atributos.

Elementos principales de la perspectiva Java

- Explorador de paquetes
 - Muestra una vista de los proyectos del workspace, los paquetes, las clases java y las librerías.
- Jerarquía de tipos
 - Muestra la jerarquía de herencia desde la clase Object hasta la clase que se está inspeccionando.

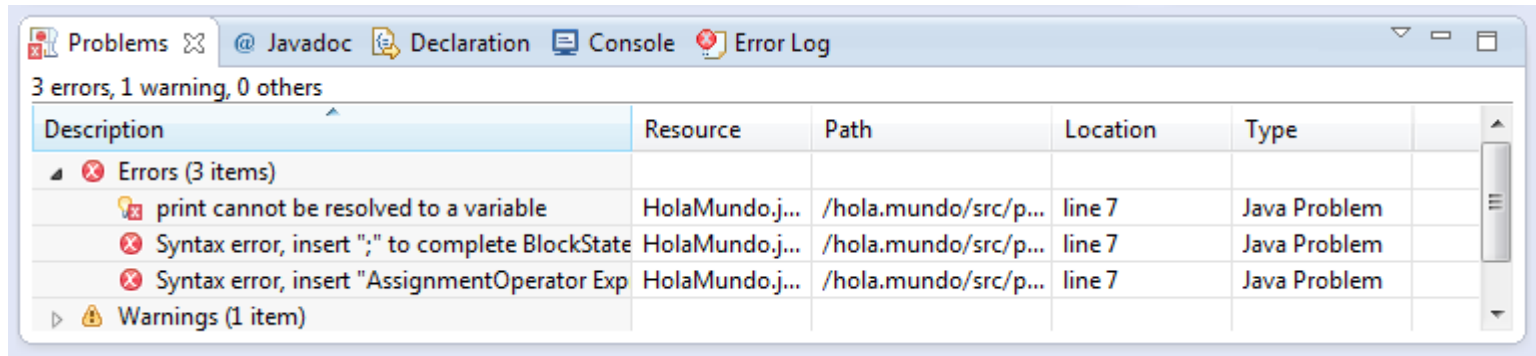


- Vista outline
 - Permite visualizar, filtrar y editar los elementos de una archivo Java.

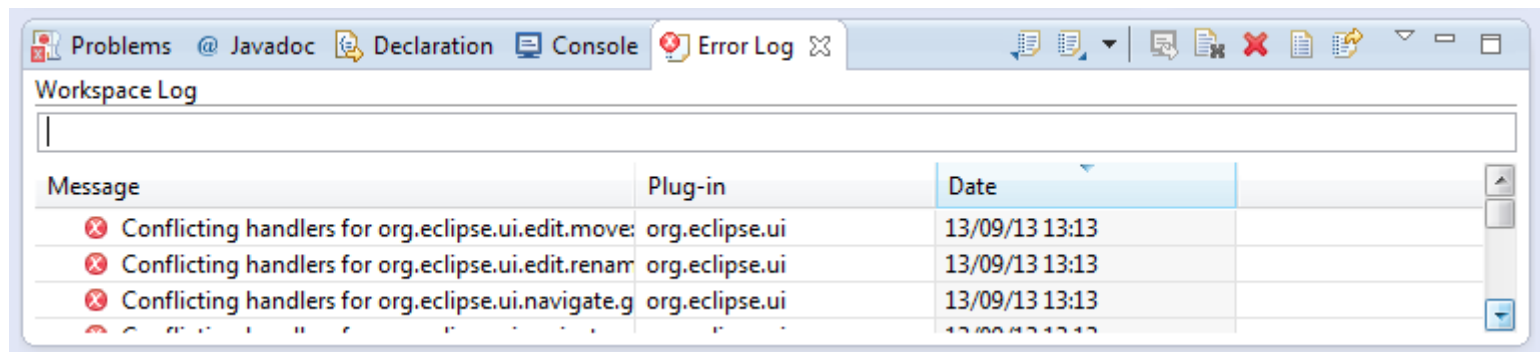


Elementos principales de la perspectiva Java

- Vista de problemas: muestra información sobre los problemas encontrados en los recursos del workspace como por ejemplo errores de compilación.



- Vista de errores: muestra los errores y advertencias asociados a la ejecución del entorno.



Entorno de compilación

- Compilación en tiempo de edición

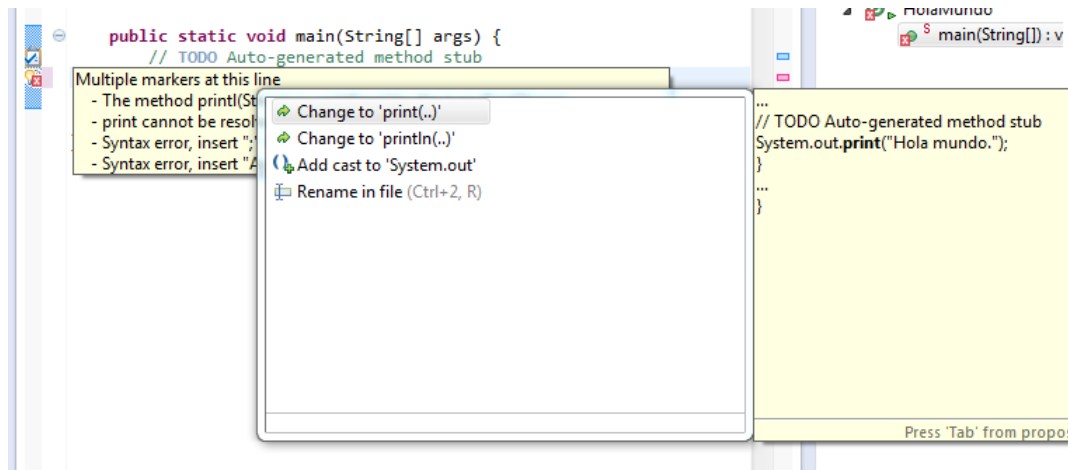
```
public class HolaMundo {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        System.out.println("Hola mundo.");  
    }  
}
```

Los errores de compilación
aparecen subrayados en rojo

La explicación del error aparece:

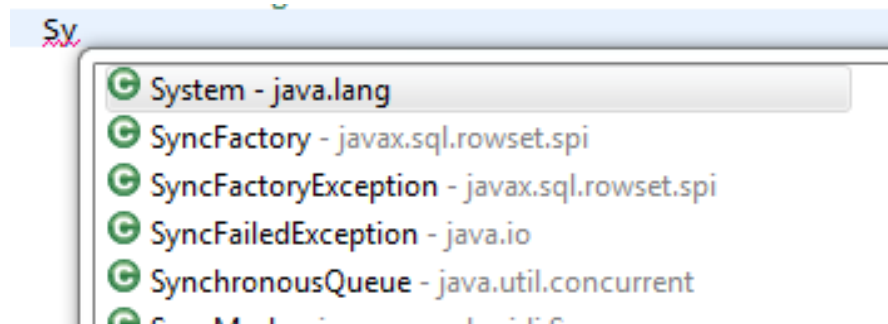
- Al situar el Cursor sobre el símbolo
- En la vista Problems ([Window | Show View | Problems](#))

- Auto-corrección pulsando con el botón izquierdo sobre el símbolo

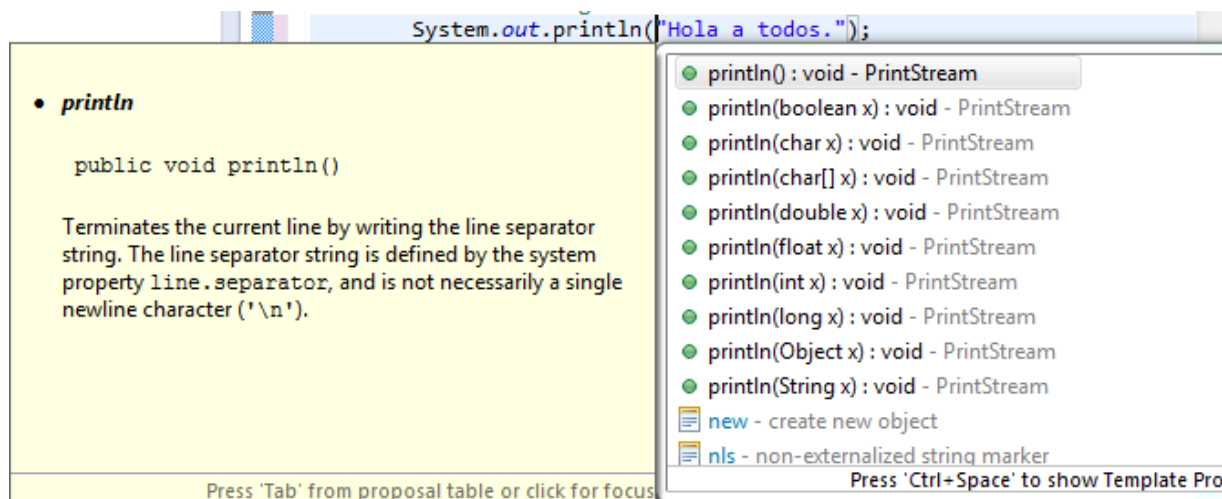


Capacidades del editor

- Función auto-completar: utilice **Ctrl + espacio**
 - Al escribir las primeras letras del nombre de una clase, un atributo o de un método muestra una lista de alternativas



- En el caso de métodos muestra las posibles firmas.



Capacidades del editor

- Menú **Source**:
 - Comentar líneas o bloques
 - Dar formato al código automáticamente según las preferencias definidas
 - Añadir y organizar instrucciones de importación de clases
 - Generar métodos getters y setters
 - Generar el constructor a partir de los atributos.
 - ...
- Menú **Refactor**:
 - Renombrar archivos y elementos de java (clases, métodos, ...) actualizando referencias
 - Mover archivos y elementos de Java.
 - ...

Operaciones sobre proyectos Java

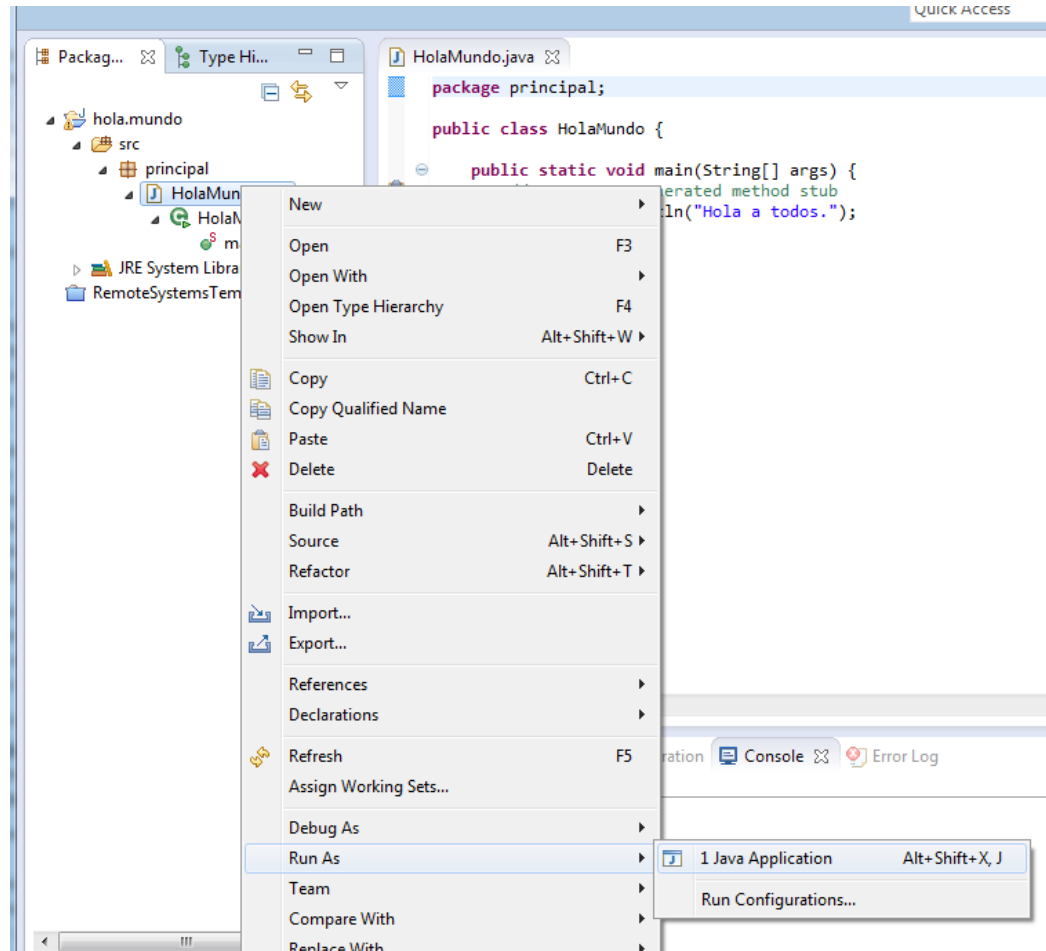
- Abrir proyectos desde el workspace:
 - En el explorador de paquetes con el botón derecho del ratón sobre el proyecto: **Open Project**.
- Exportar proyectos **File | Export**
 - **General > Archive File**
 - Exporta el proyecto actual en formato comprimido (.zip, .tar, etc.)
 - **General > Java > JAR File**
 - Exporta el proyecto actual en un archivo *.jar*
 - Puede ejecutarse desde la línea de comandos con *java -jar nombre_archivo.jar*

Operaciones sobre proyectos Java

- Importar proyectos **File | Import**
 - **General > Existing Projects into Workspace**
 - Abre un proyecto ya existente situado fuera del workspace (en una carpeta o en un archivo comprimido)
 - Para copiarlo en el workspace marque la opción:
Copy projects into workspace
- Cambiar de workspace
 - **File | Switch workspace**
- Importar librerías **.jar** (no incluidas en JRE estándar)
 - **Project | Properties > Java Build Path**
 - Pestaña **Libraries > Add External Jars**

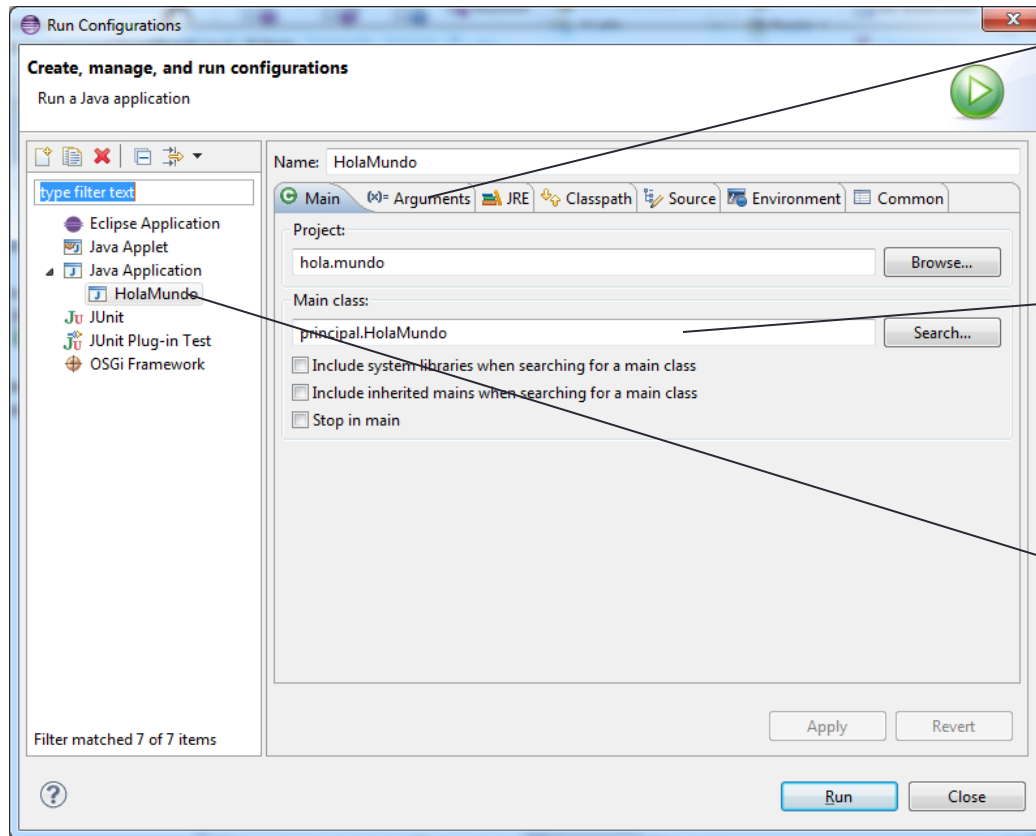
Ejecución de programas en el entorno

- La forma más sencilla es seleccionando la clase que contiene el *main* y mediante el menú contextual seleccionar **Run as** → **Java Application**, o bien en el menú principal **Run** → **Run**.



Ejecutar programas en el entorno

- De manera alternativa puede definirse una configuración para ejecutar programas seleccionando **Run → Run configurations**.






Parámetros del programa

Clase que contiene el main

Tipo de configuración

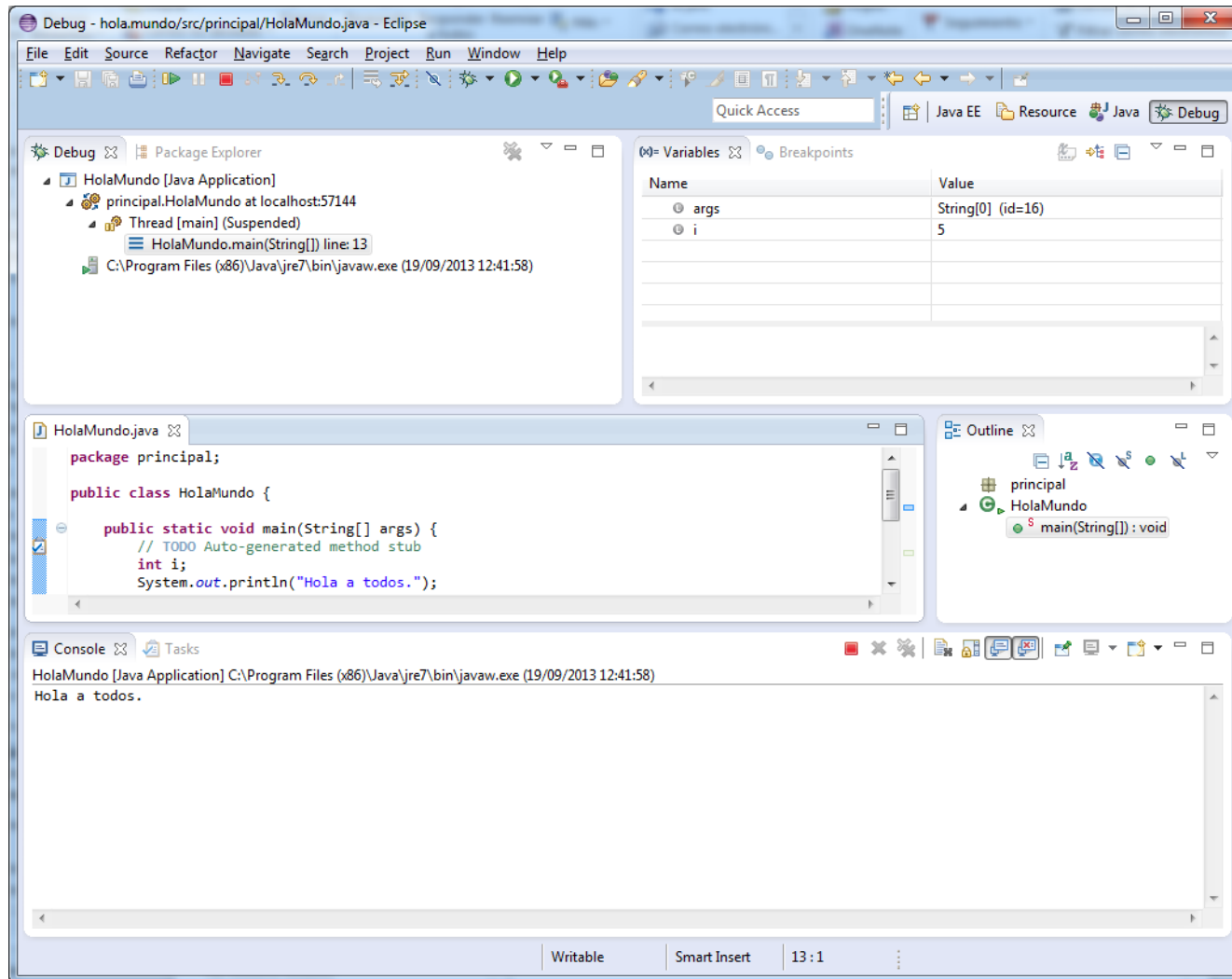
- El resultado de la ejecución se muestra en la vista de consola.

Depurador


- Window | Open Perspective > Debug
- Punto de interrupción (Breakpoint)
 - Ctrl+Shift+B
 - Run | Toggle Breakpoint
 - Con el botón derecho del ratón sobre el margen izquierdo del editor y Toggle Breakpoint
 - La ventana Breakpoint Properties permite introducir condiciones de activación del punto de interrupción
- Ejecutar para Depurar: 
 - Run | Debug As > Java Application
 - F11 ejecuta la última configuración
- Step over: F6 
 - Continúa la ejecución en la línea siguiente
- Step into: F5 
 - Continúa la ejecución dentro del método

Depurador

- La perspectiva Debug contiene:



Depurador

- Hasta el Cursor: Run | Run to line (Ctrl + R)
- Detener la depuración: Run | Terminate 
- Continuar la ejecución normal: Run | Resume
- Vista **Variables**:
 - Muestra el valor de las variables en cada momento de la ejecución
- Puntos de interrupción para excepciones
 - Interrumpen el programa cuando se produce la excepción indicada, p.e. NullPointerException
 - Run | Add Java Exception Breakpoint

Ejemplo depuración

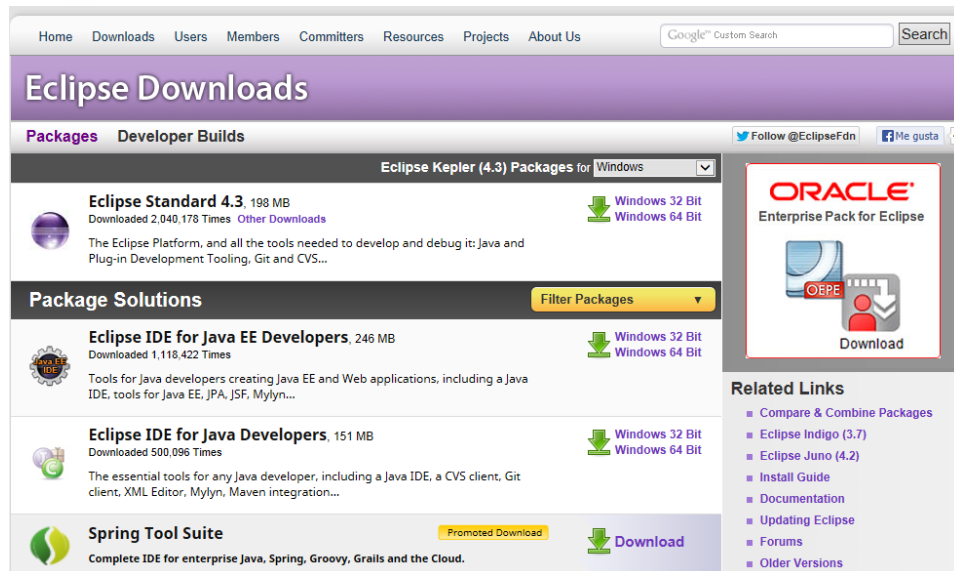
- Añada en el *main* de la clase ejemplo el siguiente código:

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    int i;  
    System.out.println("Hola a todos.");  
    for (i=5; i>=0; i--)  
        System.out.println("5 entre " + i + " es igual a " + 5/i);  
}
```

- Ponga un punto de ruptura en el cuerpo del for. **Run | Toggle Breakpoint**
- Ejecute en modo depuración **Run | Debug As.**
- Ejecute paso a paso con **F6** hasta que se produzca el error .
- En la vista **Variables** puede inspeccionar el valor de la variable *i*.

Instalación de Eclipse

- Requerimientos:
 - JRE versión 1.5 o superior.
 - En los laboratorios está instalado para las prácticas Eclipse 4.3 aunque puede utilizar otras versiones de Eclipse.
- Pasos de instalación:
 - En <http://www.eclipse.org/downloads/>



- Pulse en Older Versions para versiones anteriores.

Instalación de Eclipse

- Seleccione Eclipse Standard 4.3
- En la nueva ventana seleccione la versión que corresponda a su sistema operativo.
- Si la instalación es sobre windows pulse sobre el enlace correspondientes, el archivo que aparece en la descarga es `eclipse-standard-kepler-R-win32.zip`
- Descomprimir en la ruta escogida. Se recomienda que la carpeta esté vacía (no sobrescribir una instalación antigua).

Download Links

[Windows 32-bit](#)

[Windows 64-bit](#)

[Mac OS X\(Cocoa 32\)](#)

[Mac OS X\(Cocoa 64\)](#)

[Linux 32-bit](#)

[Linux 64-bit](#)

Downloaded 2,040,178 Times

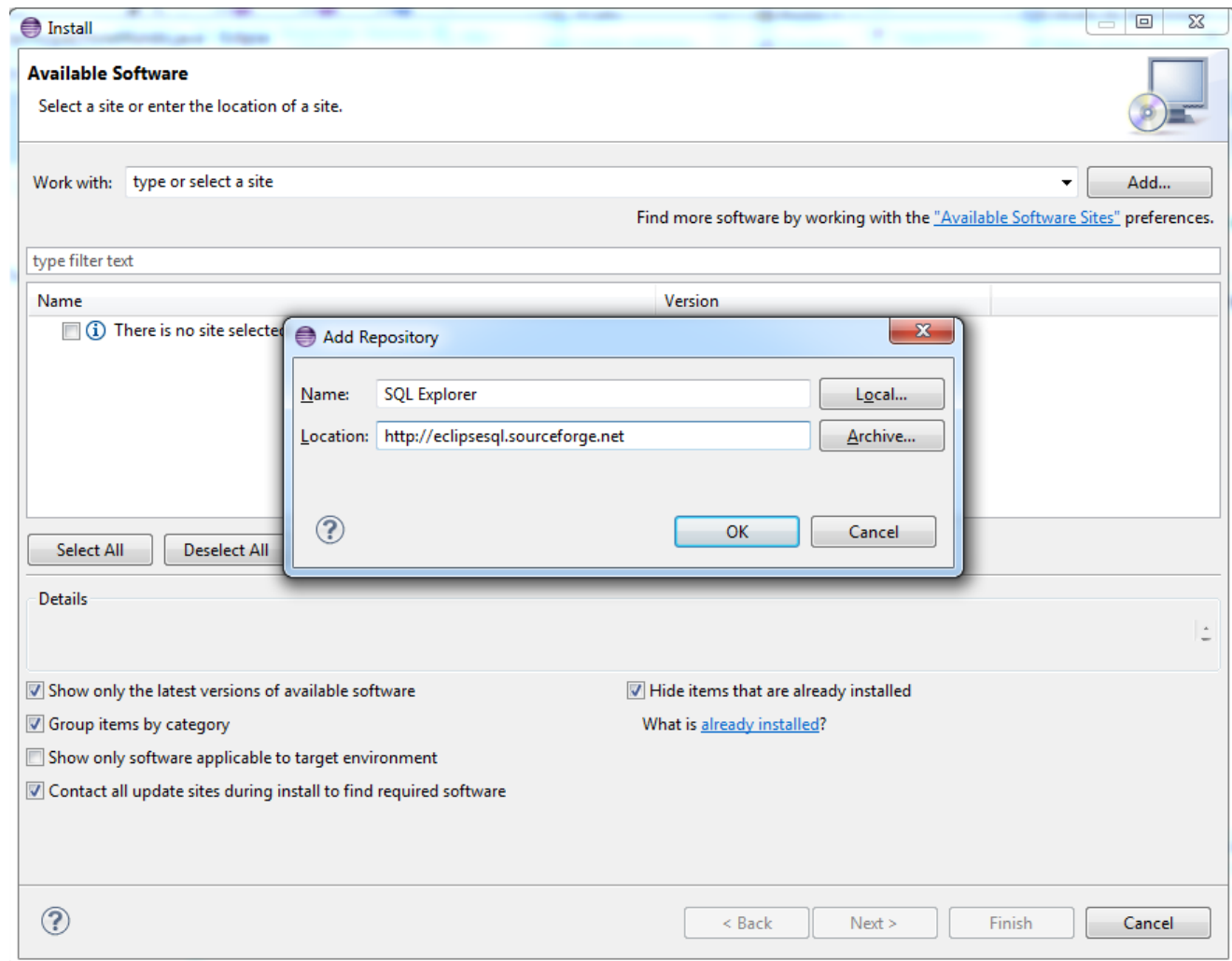
► [Checksums...](#)

Instalación de Eclipse

- Descargar el plug-in para diseñar el modelo relacional
 - (Clay) Instalar Eclipse Clay Azzurri (alias Clay Mark II)
 - Desde instalar nuevo software incluir la dirección.
 - ~~<http://www.azzurri.co.jp/eclipse/plugins/>~~
- Descargar SQL explorer
 - Al igual que antes utilizar la dirección para incluir nuevo software:
 - <http://eclipsesql.sourceforge.net>
- Instalar el plug-in Windows Builder de interfaces de usuario:
 - Utilizar la dirección para incluir nuevo software:
 - <http://download.eclipse.org/windowbuilder/WB/release/R201309271200/4.3/>
- Descargar el gestor de base de datos hsqldb y descomprimirlo fuera de Eclipse, utilizar la dirección en el navegador (fuera de eclipse) <http://hsqldb.org/>. En el laboratorio se encuentra instalada la versión 1.8. La instalación se explicará en el tema de acceso a datos, no es necesario que lo instale ahora.

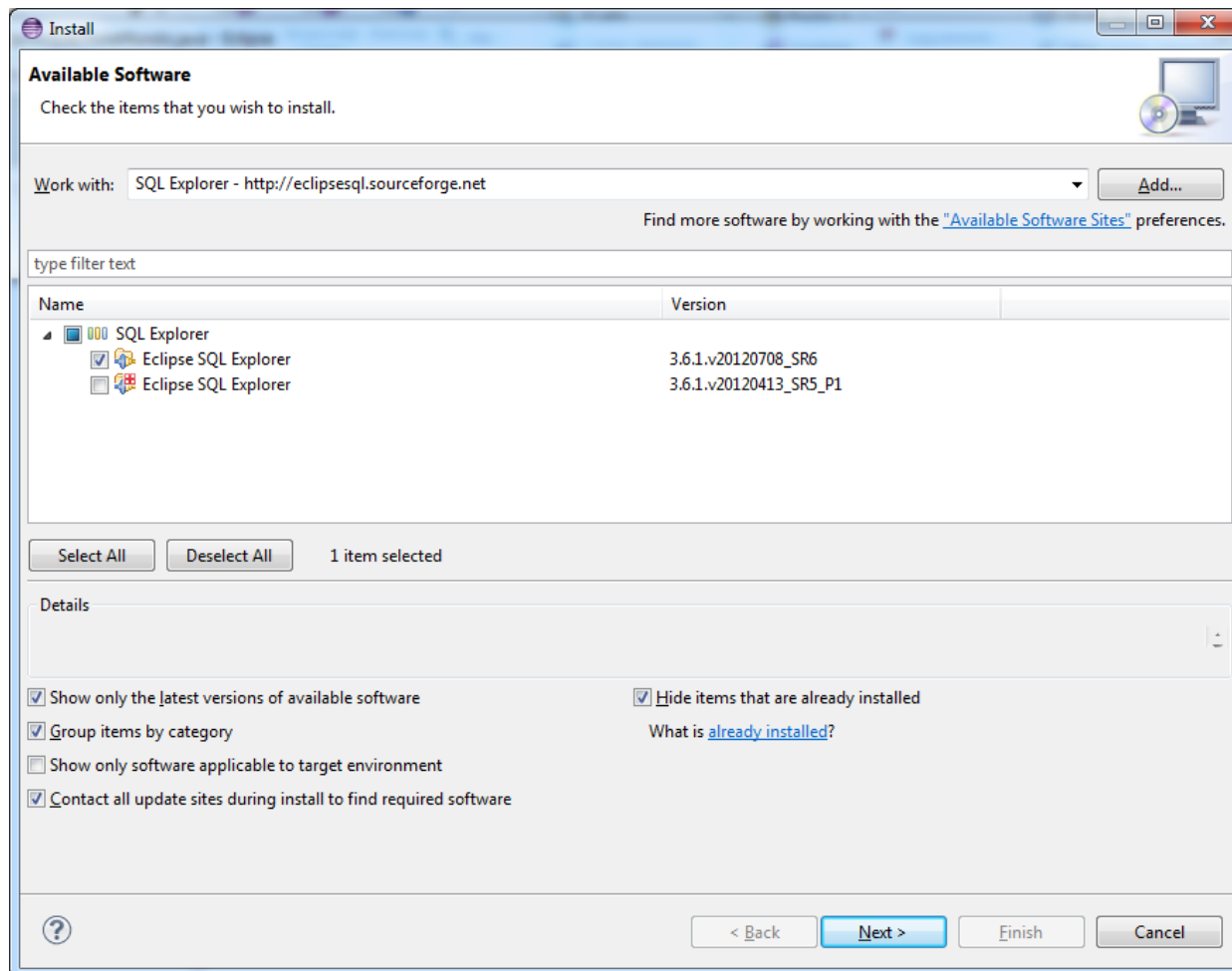
Instalación Eclipse: SQLExplorer

- Utilice el menú **Help -> Install New Software** y pulse el botón **Add**.
- Introduzca en la ventana la dirección: <http://eclipsesql.sourceforge.net> Pulse **OK**.



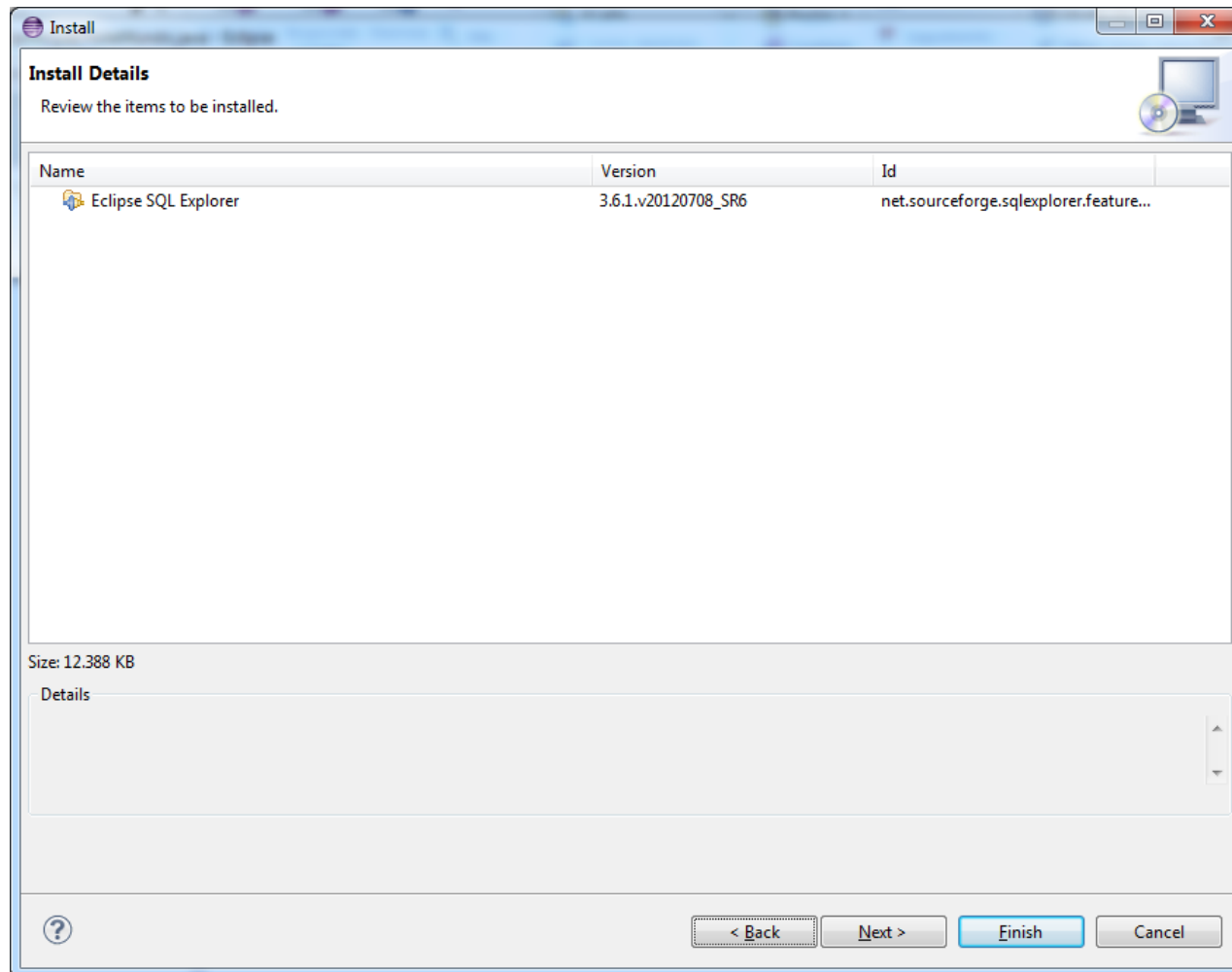
Instalación Eclipse: SQLExplorer

- Marque las casilla tal como aparecen en la figura y pulse **Next**.



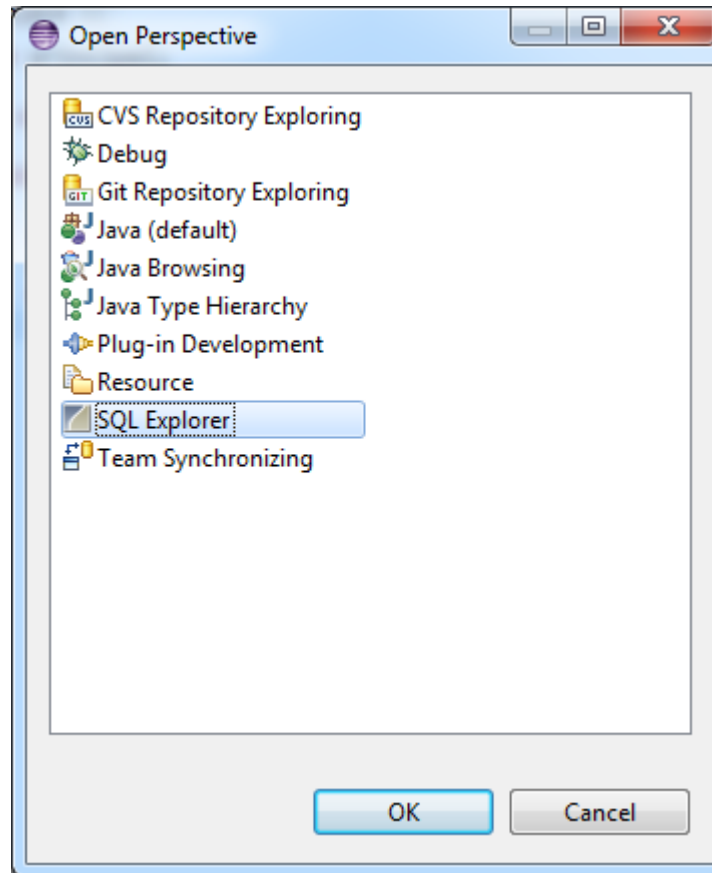
Instalación Eclipse: SQLExplorer

- En la pantalla pulse **Next**, acepte los términos de la licencia y pulse **Finish**



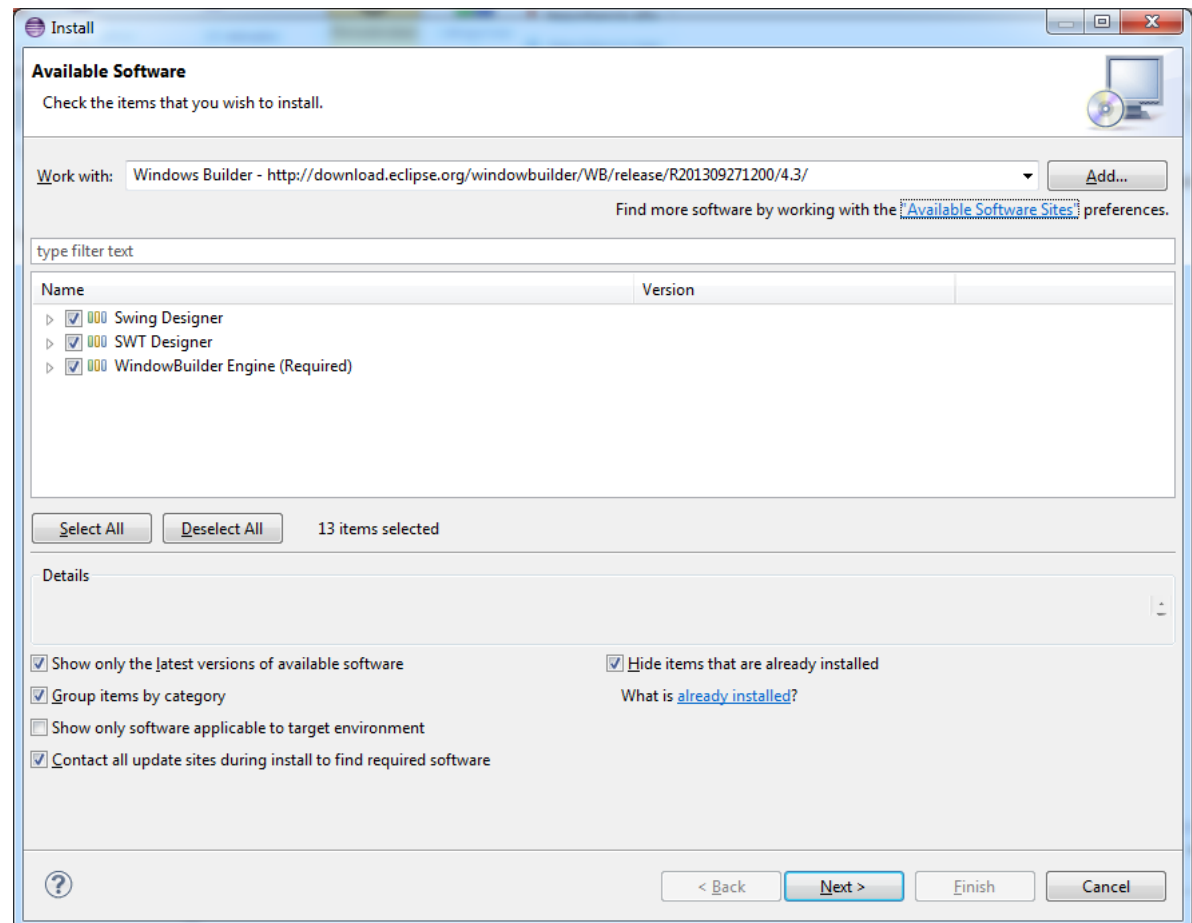
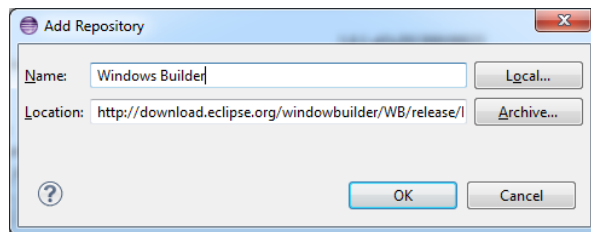
Instalación Eclipse: SQLExplorer

- Si el plug-in se instaló bien debe aparecer en la ventana de perspectivas (SQL Explorer)



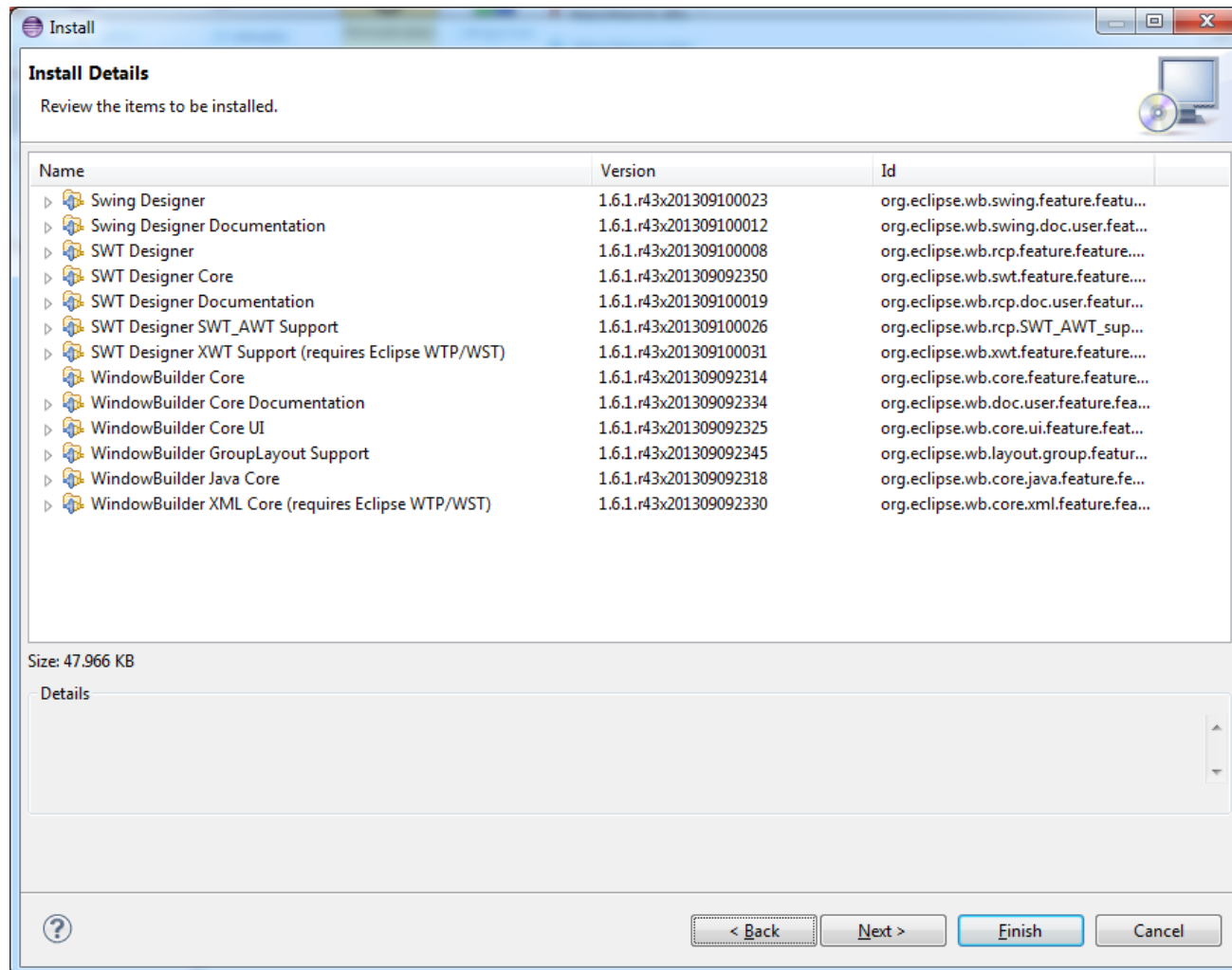
Instalación Eclipse: Windows Builder

- Proceda igual que en los casos anteriores pero ahora con la dirección:
<http://download.eclipse.org/windowbuilder/WB/release/R201309271200/4.3/>
- Marque las casillas como aparecen en la figura y pulse **Next**



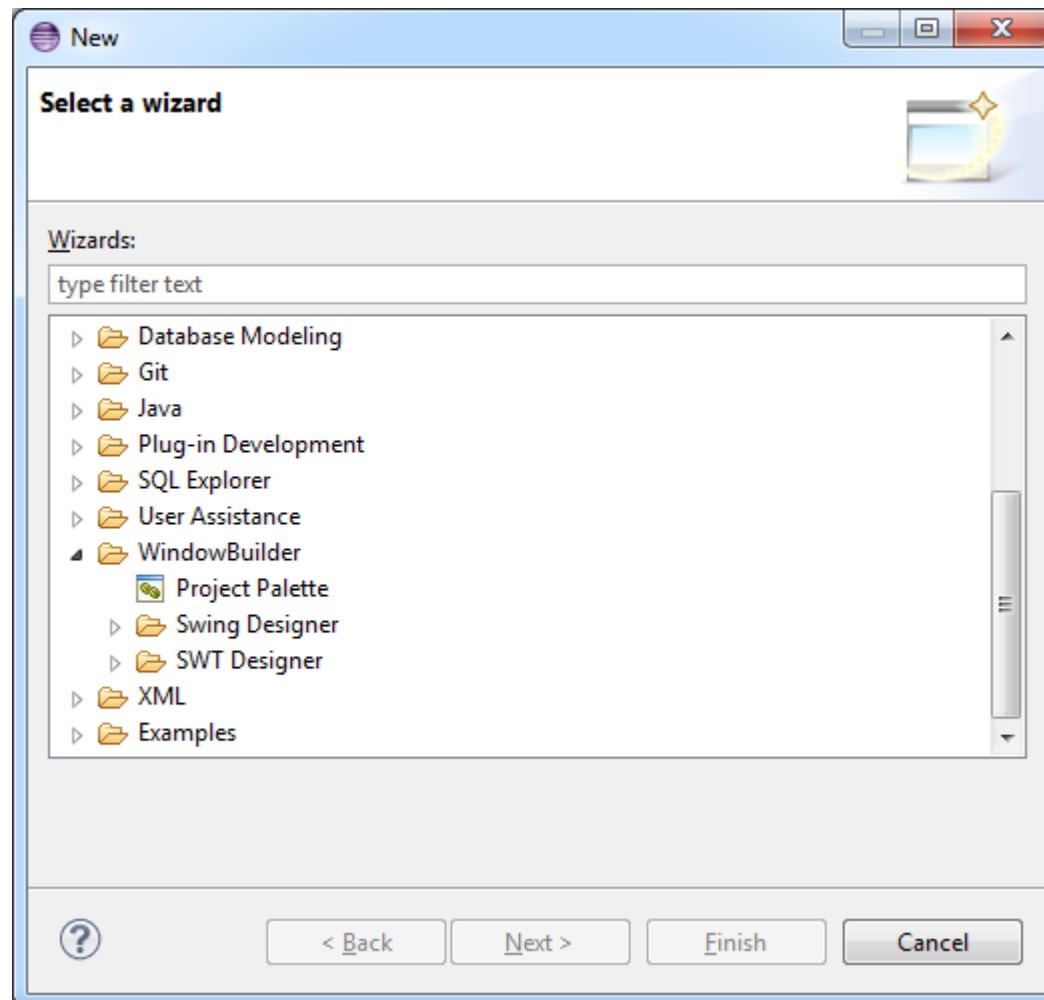
Instalación Eclipse: Windows Builder

- En la pantalla pulse **Next**, acepte los términos de la licencia y pulse **Finish**



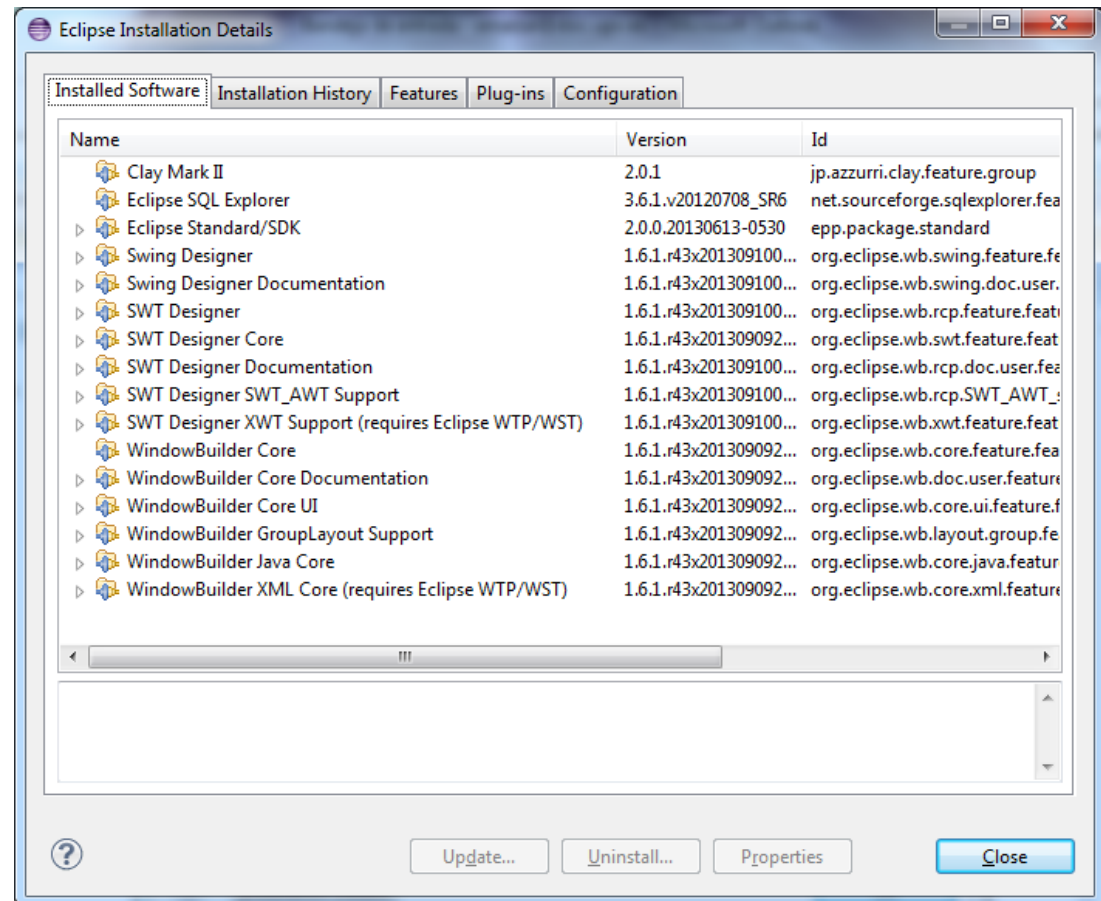
Instalación Eclipse: Windows Builder

- Para comprobar la instalación seleccione el Wizard de creación y compruebe que le aparece la carpeta Windows Builder con sus componentes.



Instalación de Eclipse: comprobación final

- Si la instalación ha terminado correctamente en **Help** → **About Eclipse** pulsando el botón **Installation Details** deben aparecerle las entradas de la figura. Observe que en algunas instalaciones del laboratorio pueden aparecer entradas adicionales.



Ejercicios: Lista de Figuras en Java (I)

Ejercicio 1: Aplicación Lista de Figuras paso a paso con Eclipse

1. Cree un nuevo proyecto llamado **ListaFiguras**
 - **File | New Project > Java Project**
 - **Separe en carpetas diferentes ficheros fuentes y compilados.**
2. Cree un nuevo paquete llamado **logica** (en src)
3. Importe la jerarquía de clases **Figura** / o simplemente **copie y pegue** el fichero
 - **File | Import > File System**
 - Escoja el directorio origen
 - Seleccione el fichero `Figura.java`
 - Indique como carpeta destino, la carpeta `logica` del proyecto (into folder)
4. Repita el paso 3 para las clases **Circulo** y **Rectacgulo**
5. Cree la clase principal **Practical1**
 - **File | New > Class**
 - **Name:** practical, **Package:** (logica o cualquier otro)
 - Marque las opciones **public**, y,
 - **public static void main** para generar el método principal
 - Copie el código siguiente en el método main() para probar la clase figura:

```
Circulo ci = new Circulo(100,100,50);  
Rectangulo cu = new Rectangulo(200,200,50,50);  
System.out.println("Área del círculo: " + ci.area());  
System.out.println("Área del rectángulo: " + cu.area());
```

6. Compile y ejecute la aplicación.
 - **Run | Run As > Java Application**
 - Debe estar activa la clase principal

Ejercicio: Lista de figuras en Java (II)

```
public abstract class Figura {  
    protected int x,y;  
    public Figura (int nx, int ny) {  
        x=nx; y=ny;  
    }  
    public abstract double area();  
    public void desplazar(int nx, int ny)  
    {  
        x = x + nx;  
        y = y + ny;  
    }  
}
```

```
public class Circulo extends Figura {  
    private int radio;  
    public Circulo(int nx, int ny, int nr){  
        super(nx,ny);  
        radio=nr;  
    }  
    public double area(){  
        return Math.PI*radio*radio;  
    }  
}
```

```
public class Rectangulo extends Figura {  
    private int alto, ancho;  
    public Rectangulo(int nx, int ny,int al, int an){  
        super(nx,ny);  
        alto=al; ancho=an;  
    }  
    public double area(){  
        return altot*ancho;  
    }  
}
```

```
public class Practical{  
  
    public static void main(String[] args) {  
        Circulo ci = new Circulo(100,100,50);  
        Rectangulo cu = new Rectangulo(200,200,50,50);  
        System.out.println("Área del círculo: " + ci.area());  
        System.out.println("Área del rectángulo: " + cu.area());  
    }  
}
```


Ejercicio: Lista de figuras en Java (III)

Ejercicio 2: Crear la lista de figuras

1. In el método **Main** use una colección Java: `ArrayList<T>`, `Stack<T>`, implementaciones de `Queue<T>`, etc.

```
List<Figura> miLista = new ArrayList<Figura>();  
miLista.add(new Circulo(4, 5, 6));  
miLista.add(new Rectangulo(4, 5, 6, 7));  
miLista.add(new Circulo(7, 8, 9));  
miLista.add(new Circulo(10, 11, 12));  
miLista.add(new Rectangulo(4, 5, 6, 18));  
  
for (Figura fig : miLista) {  
    System.out.println("Mi tipo es: " + fig.getClass().getName());  
    if (fig instanceof Circulo) System.out.println("Testo es un círculo");  
}
```

2. Añada a la jerarquía de Figuras la clase *Cuadrado* (derivada de Rectángulo).
3. Añada objetos cuadrado a la lista.
4. Modifique la lista de figuras para añadir un método que amplíe(n) todas las figuras de la lista.
5. Compruebe que se pueden ampliar las figuras de la lista.