Unidad 2:

Instalación y uso de entornos de desarrollo

Herramientas CASE

Entornos de desarrollo: conjunto de software que **facilitan** o **automatizan** las actividades de desarrollo.

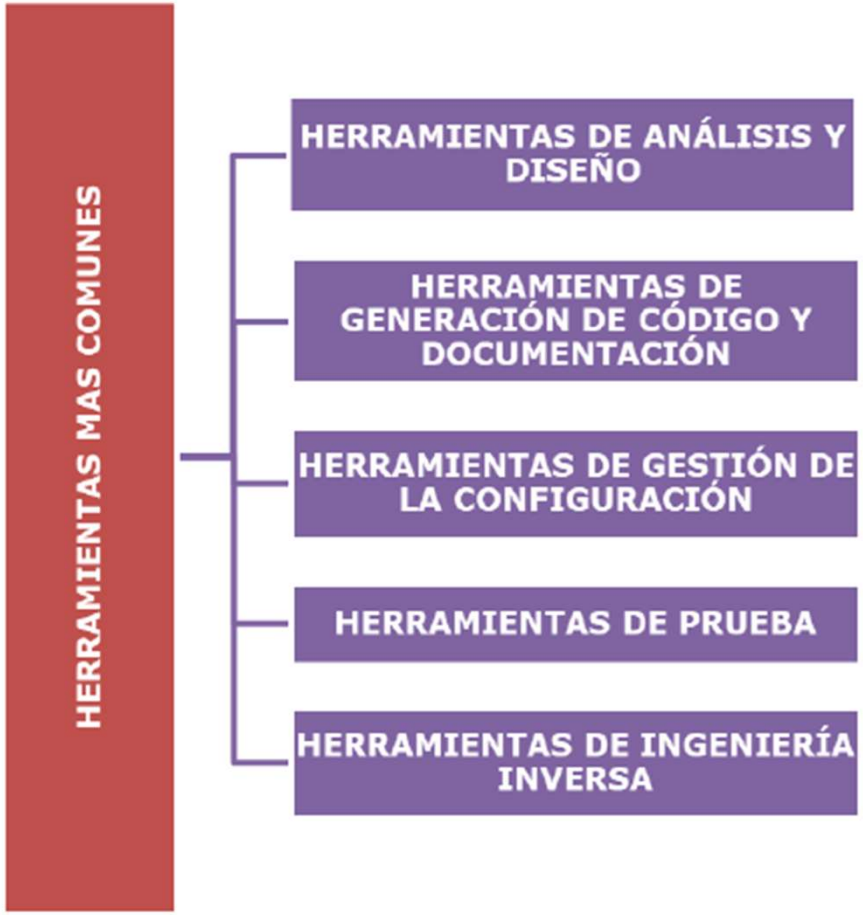
Herramientas CASE: conjunto de aplicaciones que se utilizan en el desarrollo de software con el objetivo de **reducir costes** y **tiempo** del proceso, mejorando por tanto la **productividad** del proceso.

Nuestro **objetivo** al desarrollar software para mejorar la calidad y la productividad deben ser los siguientes:

- Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue **agilizar el trabajo**.
- Facilitar la realización de **prototipos** y el **desarrollo conjunto** de aplicaciones.
- Simplificar el **mantenimiento** de los programas.
- Mejorar y estandarizar la **documentación**.
- Aumentar la **portabilidad** de las aplicaciones.
- Facilitar la **reutilización** de componentes software.

Se clasifican en función de las fases del ciclo de vida del software en la que ofrecen ayuda:

- Herramientas de **gestión**, **estimación** y **planificación** de proyectos.
- Herramientas **técnicas**, que a su vez se dividen en:
  - U-CASE (front-end): herramientas especializadas en asistir en el **análisis** y **diseño** de software.
  - L-CASE (back-end): herramientas especializadas en el **diseño detallado** y la **generación de código**.

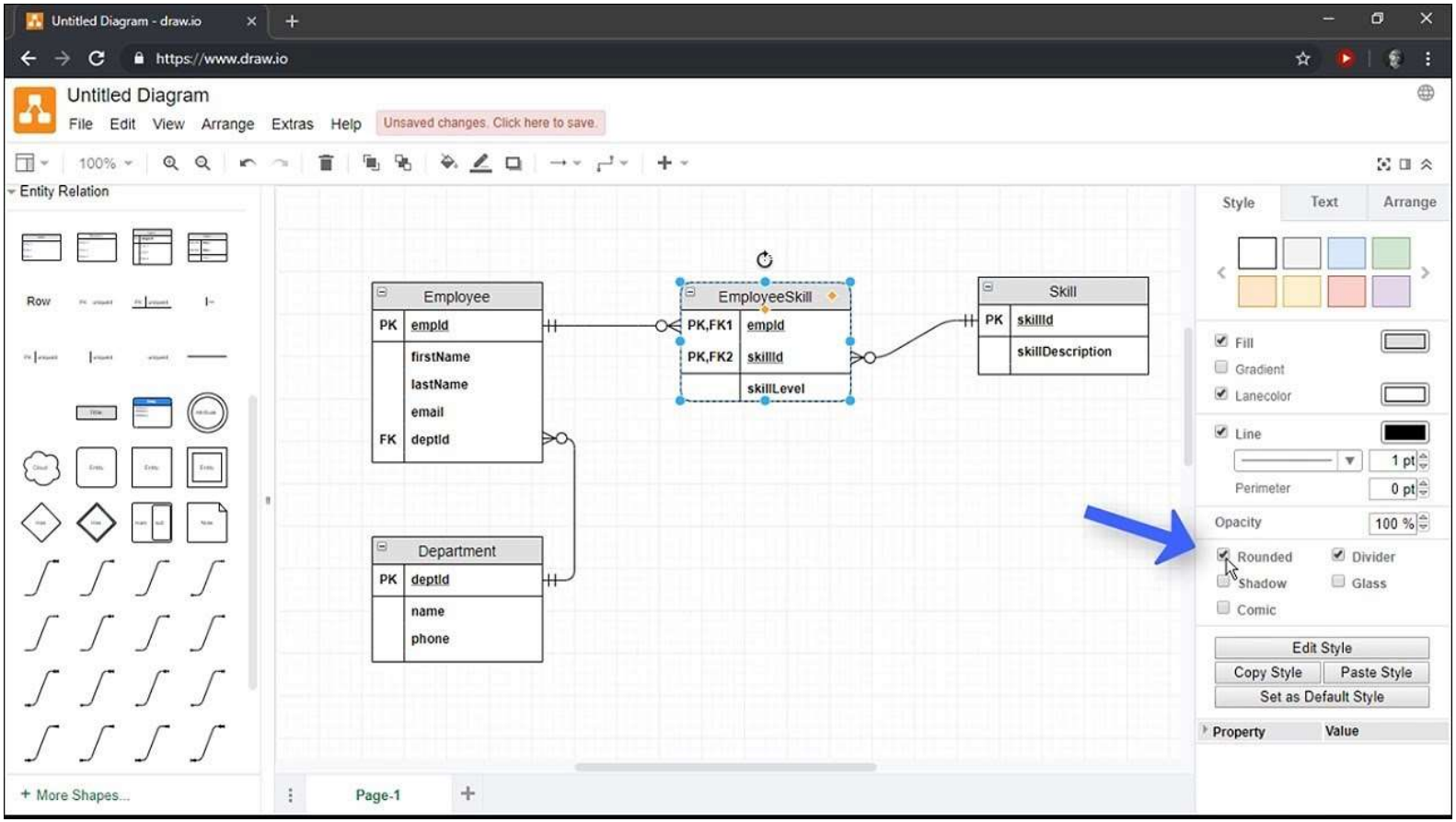

HERRAMIENTAS MAS COMUNES
HERRAMIENTAS DE ANÁLISIS Y DISEÑO
HERRAMIENTAS DE GENERACIÓN DE CÓDIGO Y DOCUMENTACIÓN
HERRAMIENTAS DE GESTIÓN DE LA CONFIGURACIÓN
HERRAMIENTAS DE PRUEBA
HERRAMIENTAS DE INGENIERÍA INVERSA

Herramientas de Análisis y Diseño: su principal objetivo es ayudar a la definición de **requisitos** del sistema y sus **propiedades**.

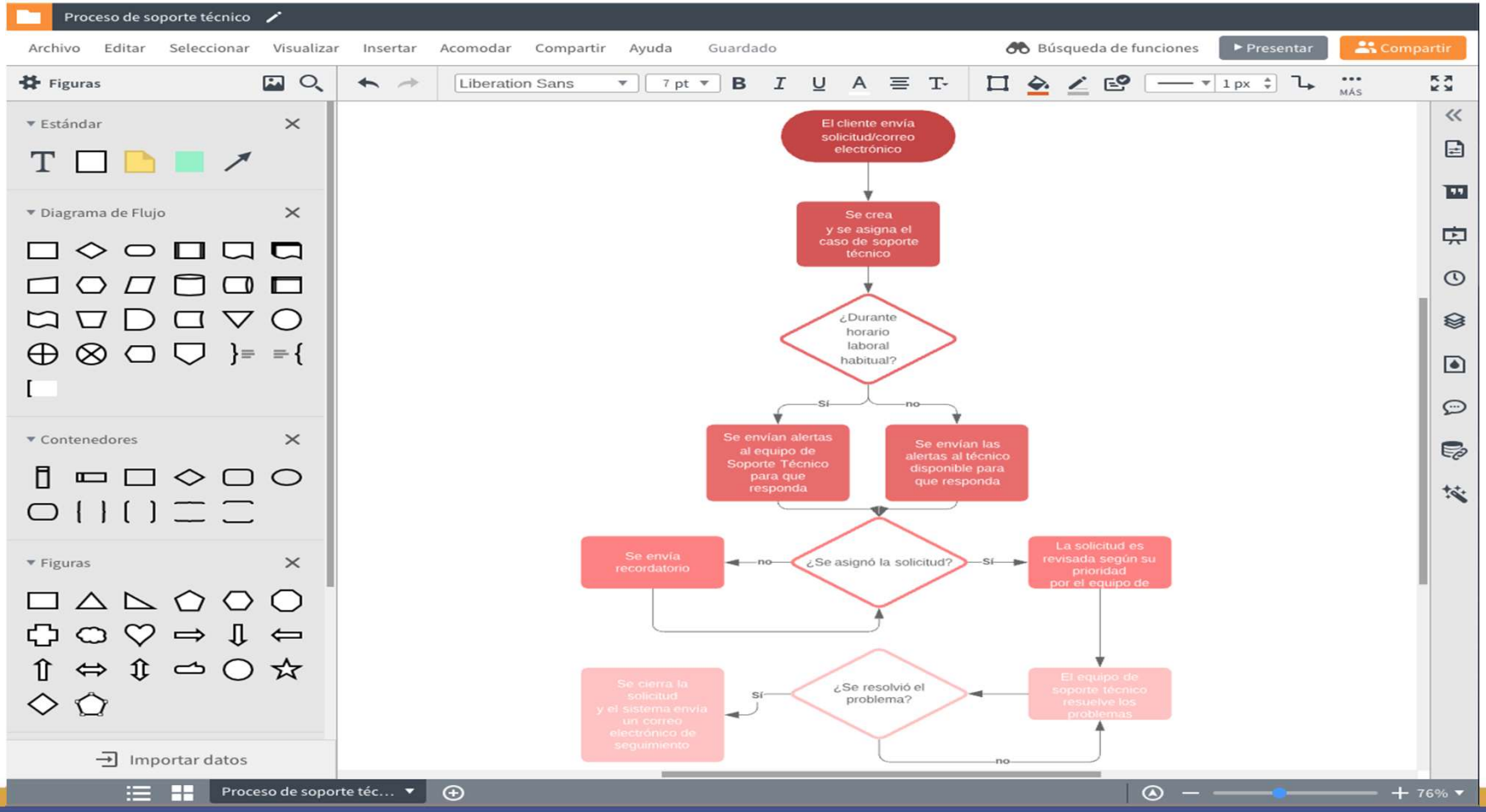
Estas herramientas permiten editar diagramas E/R (entidad-relación), diagramas de flujo de datos, diagramas de clases, etc.

También son muy importantes las herramientas de prototipado, como los diseñadores de pantallas, generadores de menús, generadores de informes, etc.

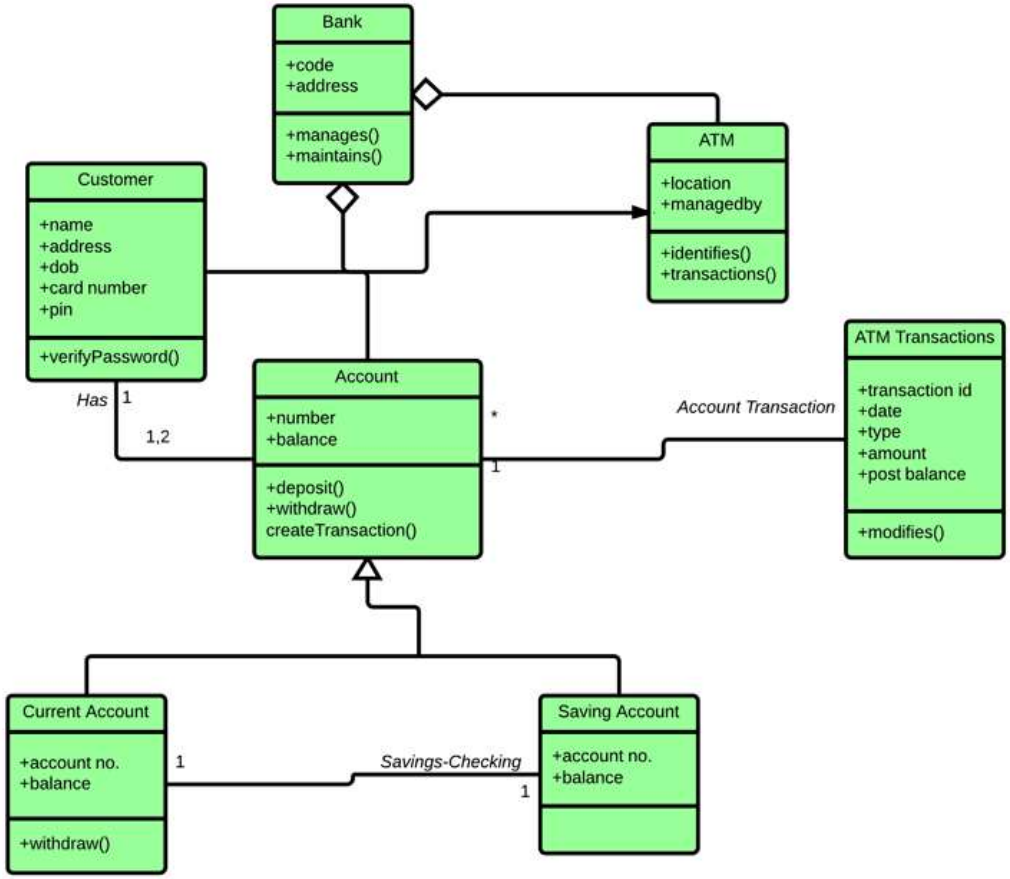
Herramienta diagramas E/R : Draw.io



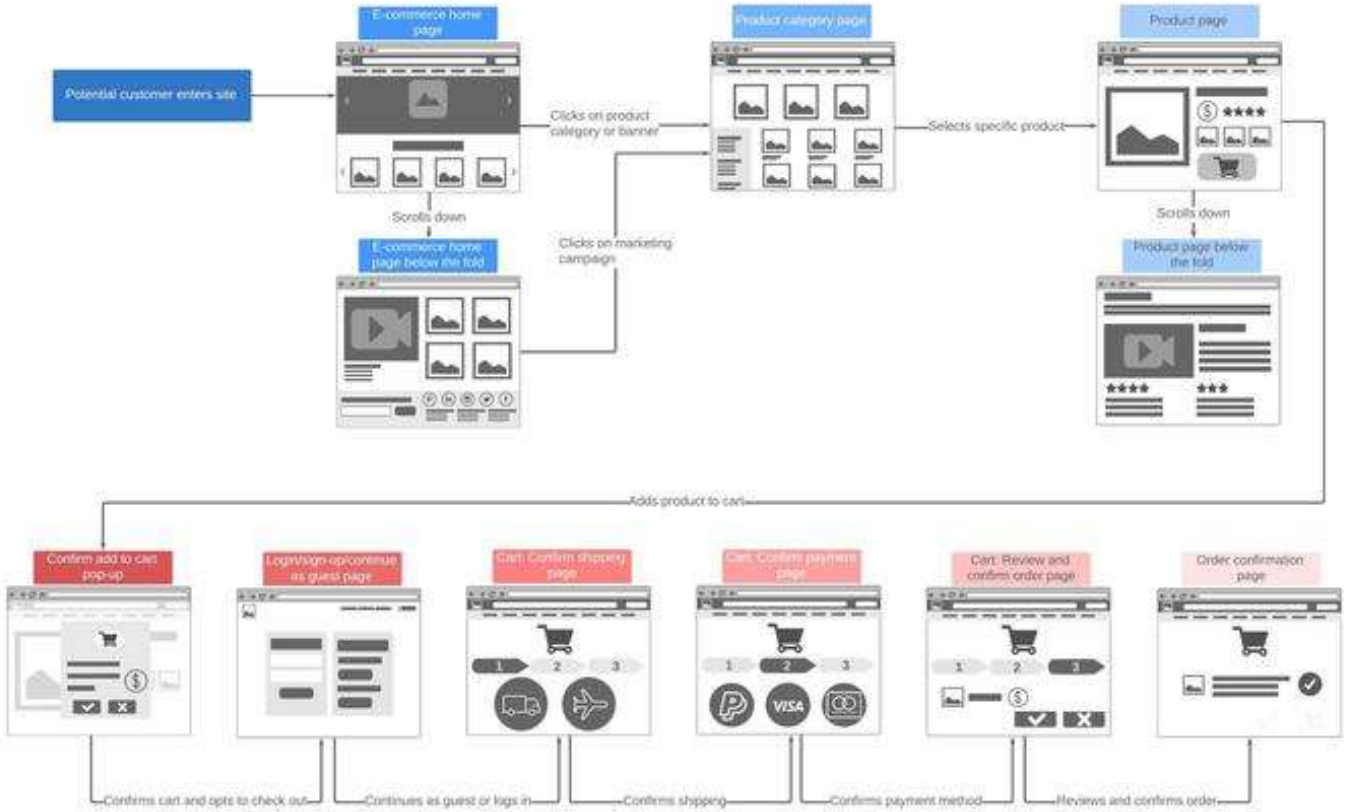
Herramienta diagramas de flujo: Lucidchart



Herramienta diagramas de clases: StartUML



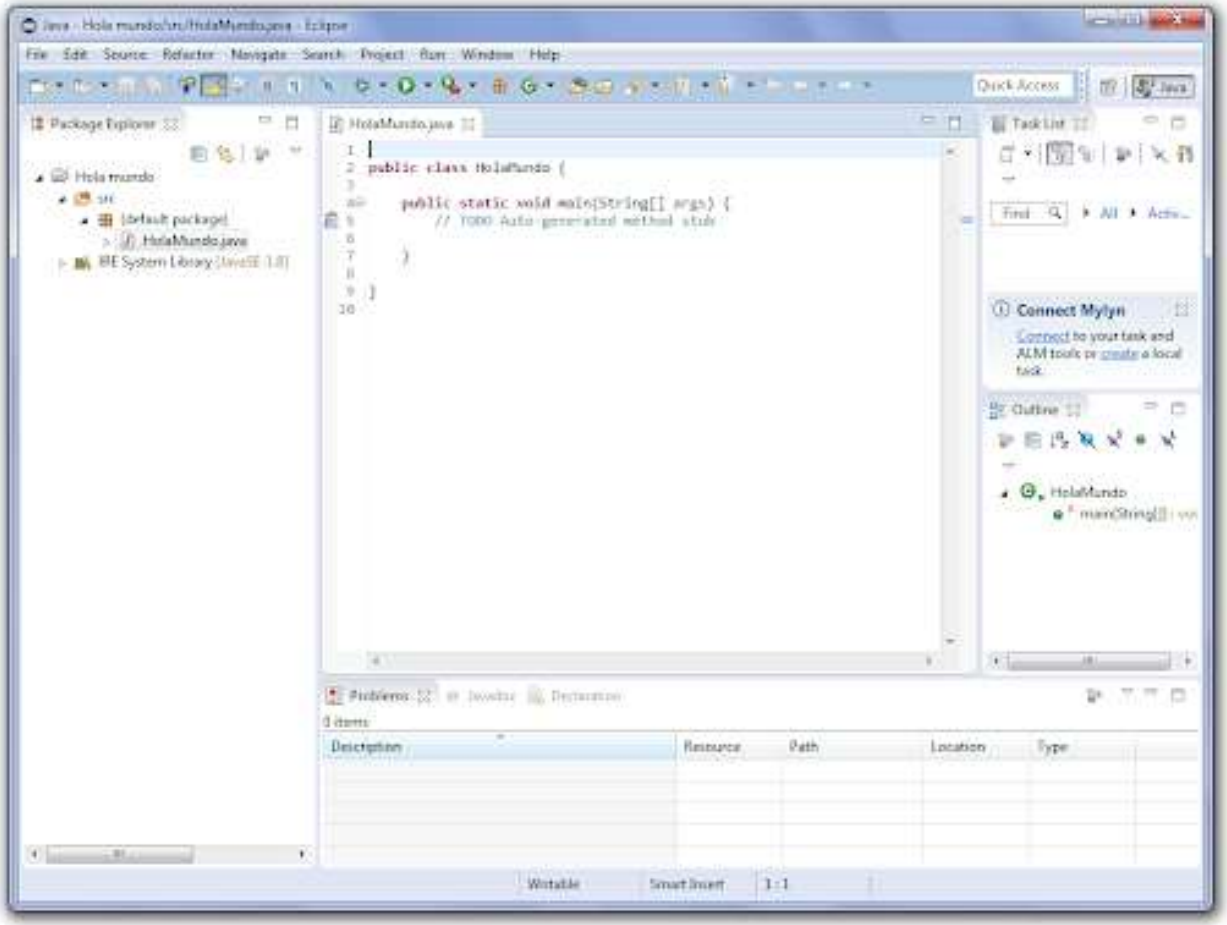
Herramienta diseñadores de pantallas: proto.io



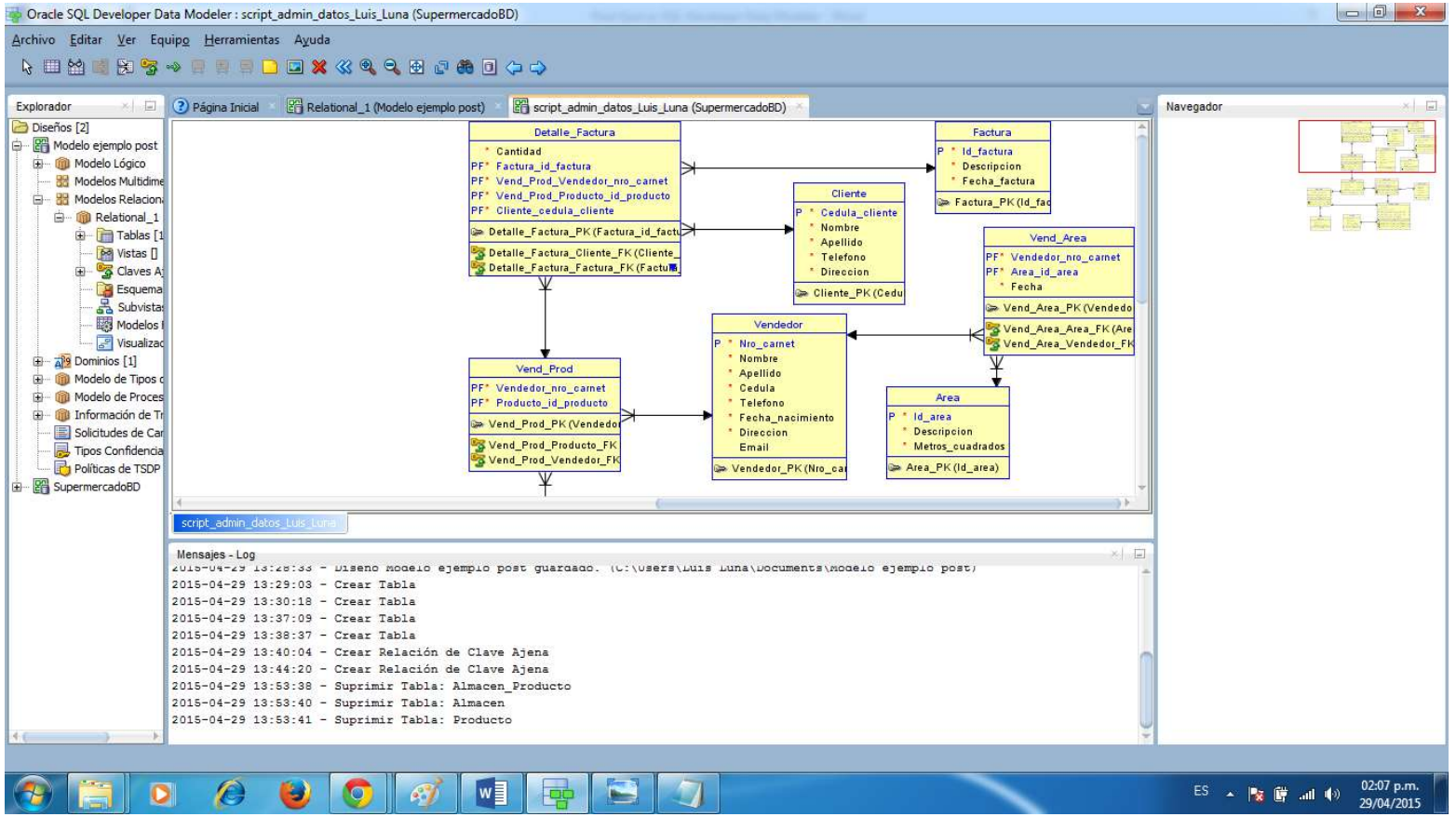
Herramientas de Generación de código y documentación: a partir de las especificaciones de diseño se puede **generar código**, tanto programas (por ejemplo, en lenguaje Java) como esquemas de base de datos (sentencias SQL de definición).

Además este tipo de herramientas soportan la **creación automatizada** de la documentación, que va desde la descripción textual de un pseudocódigo hasta diagramas más o menos complejos. (Por ejemplo generar el diagrama de relación-entidad de toda la base de datos).

Herramienta generación de código: Eclipse



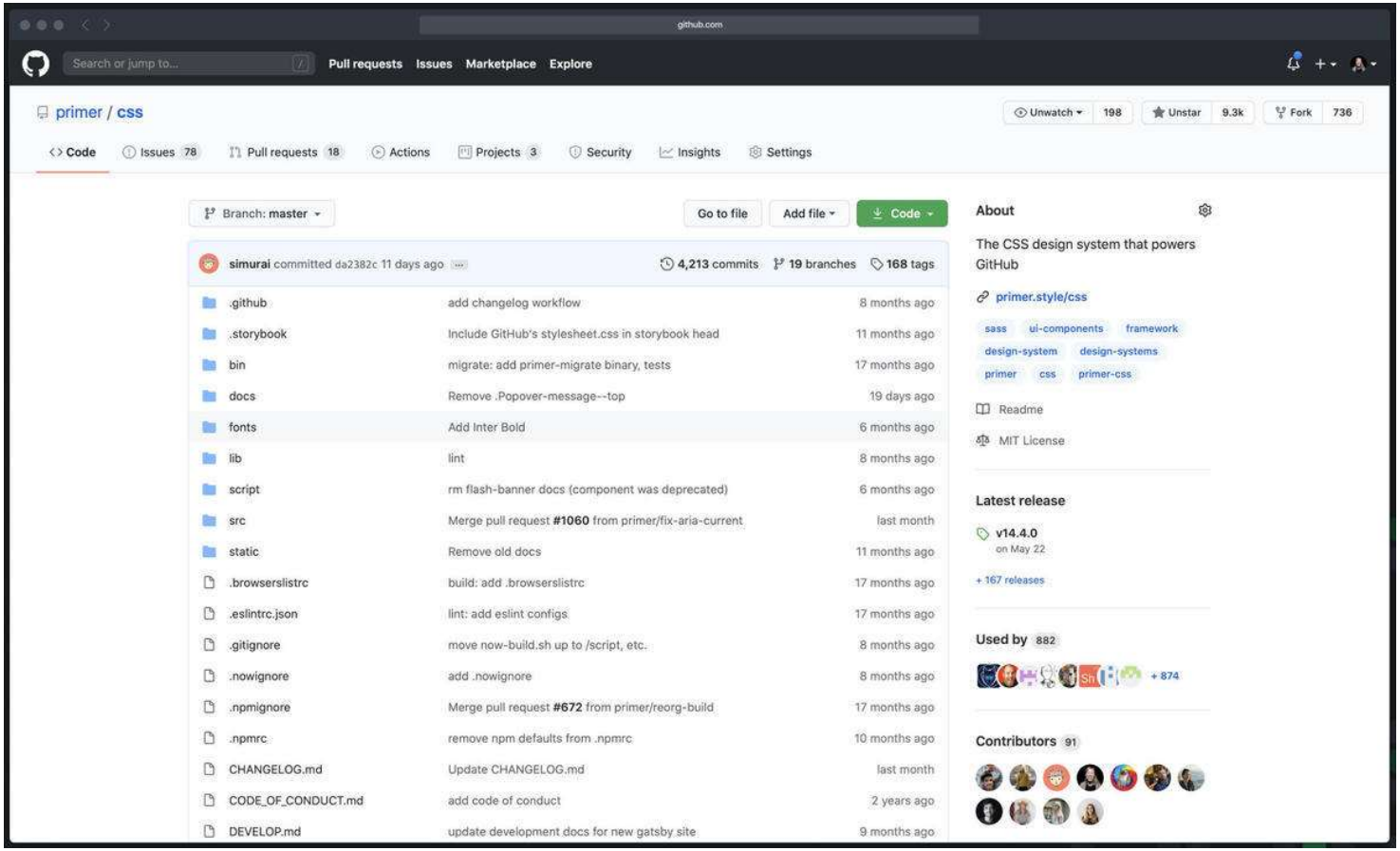
Herramienta creación automatizada: SQL Developer



Herramientas de Gestión de la Configuración: En **entornos de desarrollo complejos** se hace imprescindible la incorporación de una herramienta capaz de gestionar la configuración de los sistemas. Este tipo de herramientas ofrecen distintas capacidades:

- **Control de versiones**: capacidad de proporcionar **almacenamiento** y **acceso** controlado a los datos por parte de uno más usuarios, así como registrar los cambios sobre los mismos, y poder recuperar versiones anteriores.
- **Trazabilidad de requisitos**: Permiten que un requisito pueda ser **rastreado** hasta su implementación.
- **Análisis de impacto**: Permite conocer los elementos del **sistema** que se ven **afectados** ante un cambio.

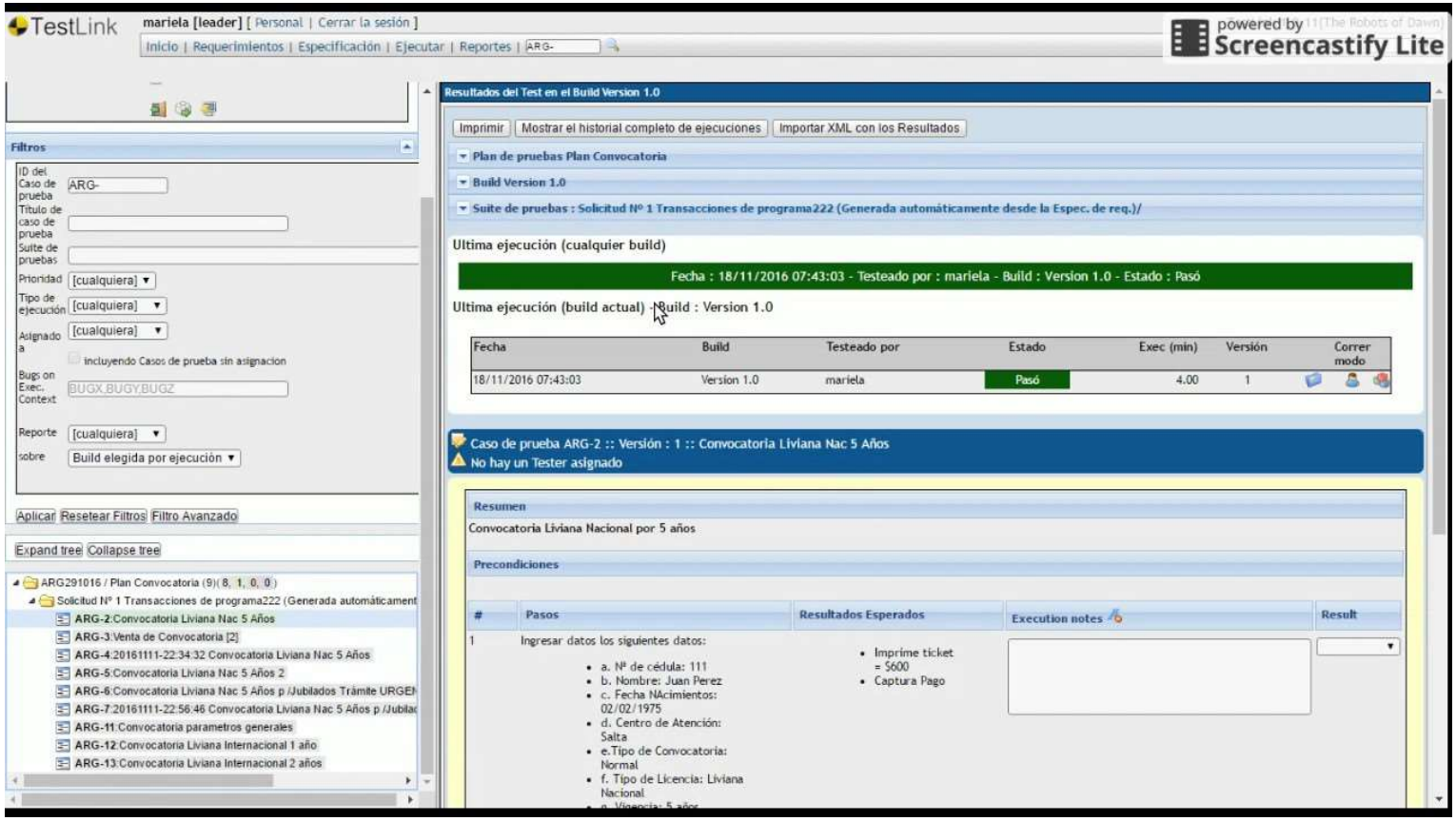
Herramienta control de versiones: Git



Herramientas de Prueba: las funcionalidades que suelen ofrecer este tipo de herramientas son las siguientes:

- Gestión de pruebas (planificación, monitorización, informes, etc).
- Definición de requisitos y objetivos de prueba.
- Diseño de pruebas (documentos de casos de prueba y resultados esperados).
- Preparar entornos de ejecución.
- Ejecución de las pruebas (fallos, estadísticas, calidad, etc.).

Herramienta de pruebas: TestLink



Entornos de desarrollo

Entorno integrado de desarrollo (IDE): es un tipo de software compuesto por un conjunto de **herramientas** de programación. En concreto, el IDE se compone de:

- Editor de código de programación.
- Compilador/intérprete.
- Depurador.
- Generador automático de herramientas: Para la visualización, creación y manipulación de componentes visuales
- Interfaz gráfica.

Normalmente, un IDE está dedicado a un determinado lenguaje de programación. No obstante, las últimas versiones de los IDEs tienden a ser **compatibles con varios lenguajes** mediante la instalación de **plugins**.

Entorno de desarrollo

Tabla de los IDE más relevantes hoy en día		
Entorno de desarrollo	Lenguaje que soporta	Tipo de licencia
NetBeans	C/C++, Java, JavaScript, PHP, Python.	De uso público.
Eclipse	Ada, C/C++, Java, JavaScript, PHP.	De uso público.
Microsoft Visual Studio	Basic, C/C++, C#.	Propietario.
C++ Builder	C/C++.	Propietario.
JBuilder	Java	Propietario.

No existe unos entornos de desarrollo más importantes que otros. La elección del IDE más adecuado va a depender del **lenguaje de programación** que vayamos a utilizar y el **tipo de licencia** con la que queramos trabajar.

Las funciones de los IDE son:

- Editor de código.
- Auto-completado de código, atributos y métodos de clases.
- Identificación automática de código.
- Herramientas de concepción visual para crear y manipular componentes visuales.
- Asistentes y utilidades de gestión y generación de código.
- Archivos fuente en unas carpetas y compilados a otras.
- Compilación de proyectos complejos en un solo paso.
- Control de versiones.
- Soporta cambios de varios usuarios de manera simultánea.
- Generador de documentación integrado.
- Detección de errores de sintaxis en tiempo real.

Otras funciones importantes:

- Ofrece refactorización de código: cambios menores en el código que facilitan su legibilidad sin alterar su funcionalidad (por ejemplo cambiar el nombre a una variable).
- Permite introducir automáticamente tabulaciones y espaciados para aumentar la legibilidad.
- Depuración: seguimiento de variables, puntos de ruptura y mensajes de error del intérprete.
- Aumento de funcionalidades a través de la gestión de sus módulos y plugins.
- Administración de las interfaces de usuario (menús y barras de herramientas).
- Administración de las configuraciones del usuario.

Entorno de desarrollo

Entornos Integrados Libres: Son aquellos con licencia de uso público.

IDE	Lenguajes que soporta	Sistema Operativo
NetBeans	C/C++, Java, JavaScript, PHP, Python	Windows, Linux, Mac OS X.
Eclipse	Ada, C/C++, Java, JavaScript, PHP	Windows, Linux, Mac OS X
Gambas	Basic	Linux
Anjuta	C/C++, Python, Javascript	Linux
Geany	C/C++, Java	Windows, Linux, Mac OS X
Geany	Fortran	Windows, Linux, Mac OS X

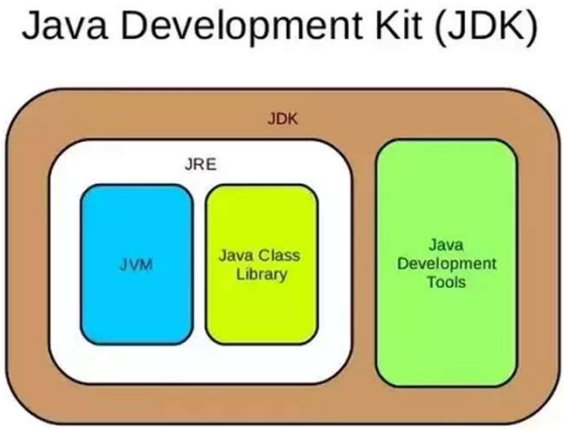
Entornos Integrados Propietarios: Son aquellos entornos integrados de desarrollo que necesitan licencia, hay que pagar con ellos

IDE	Lenguajes que soporta	Sistema Operativo
Microsoft Visual Studio	Basic, C/C++, C#	Windows
FlashBuilder	ActionScript	Windows, Mac OS X
C++ Builder	C/C++	Windows
Turbo C++ profesional	C/C++	Windows
JBuilder	Java	Windows
JCreator	Java	Windows, Linux, Mac OS X
Xcode	C/C++, Java	Mac OS X

Preparación del entorno

JDK: es el kit de desarrollo de Java, es decir, el conjunto de herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java.

Entre sus componentes incluye un intérprete en tiempo de ejecución (**JRE**) para la ejecución de programas Java (*.class) no gráficos (aplicaciones).



Actividad 1

Instalación JDK

https://www.oracle.com/java/technologies/downloads/#jdk17-windows

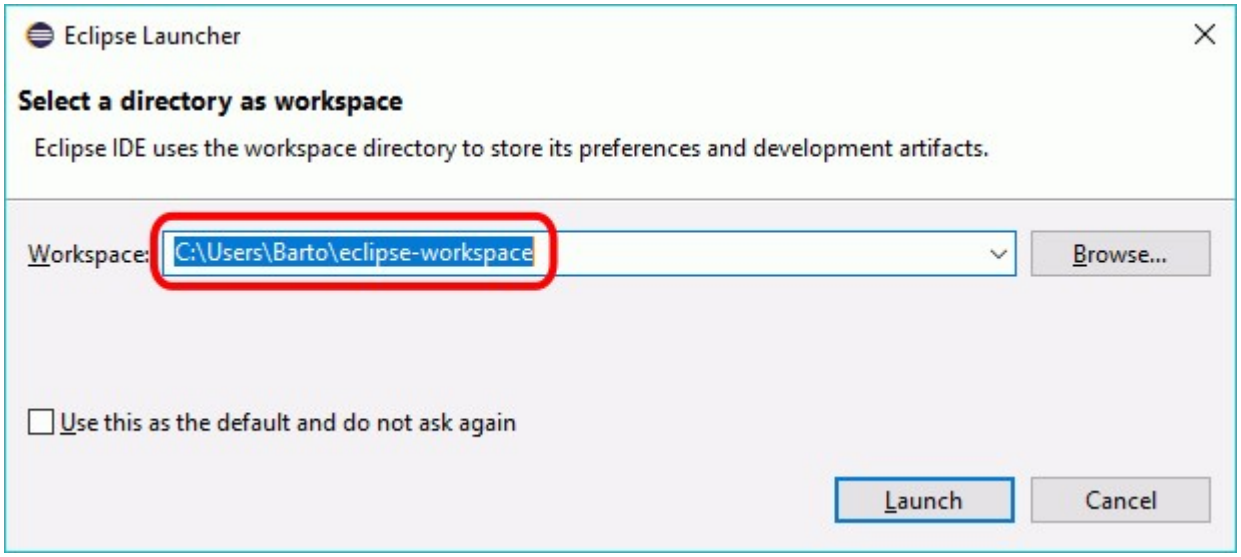
Actividad 2

Instalación eclipse

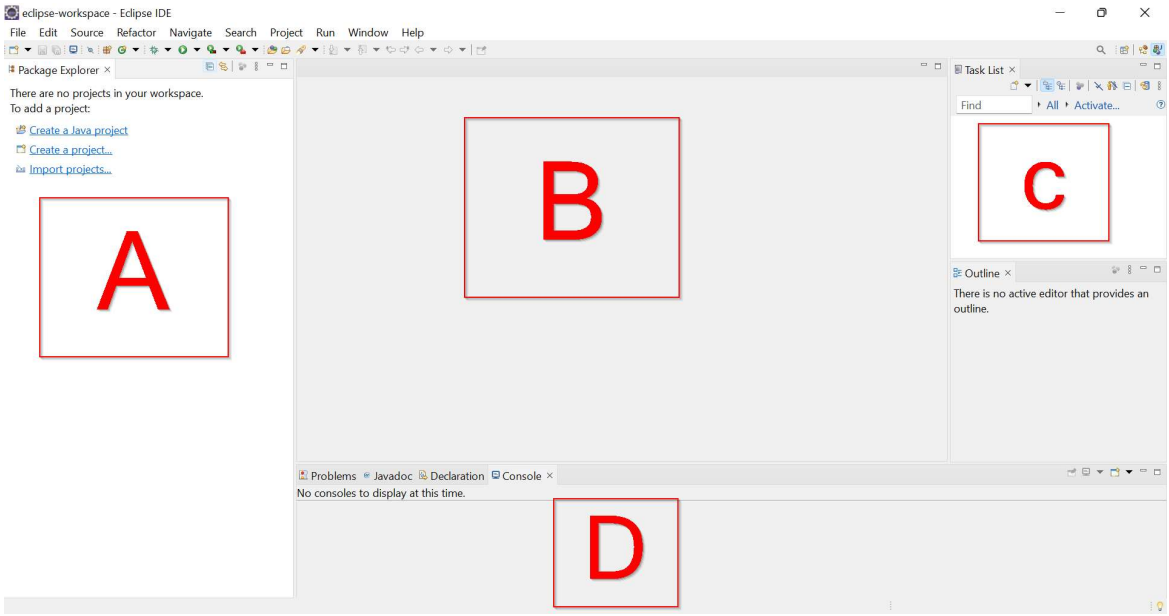
https://www.eclipse.org/downloads/

Interfaz por defecto

Espacio de trabajo: es donde se **almacena** la configuración de Eclipse y los datos del espacio de trabajo.



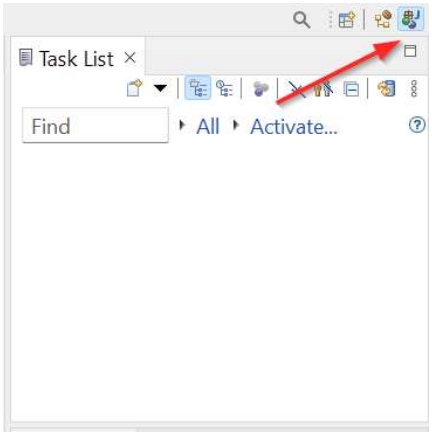
Preparación del entorno



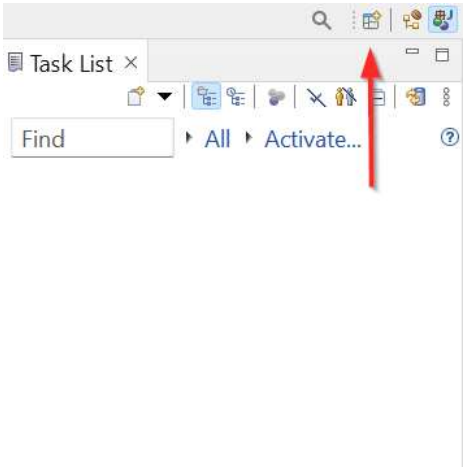
- Sección A - Explorador de paquetes
- Sección B - Editor
- Sección C - Lista de tareas y esquema
- Sección D - Panel de vistas con pestañas

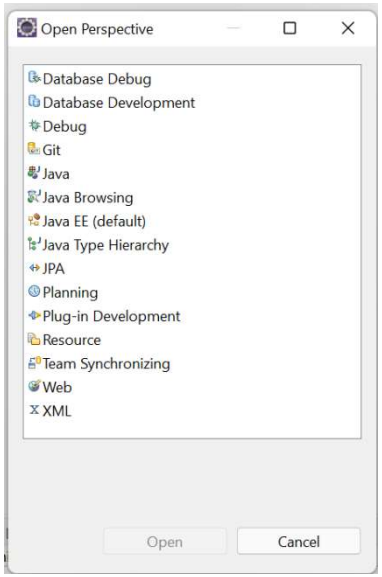
Preparación del entorno

Perspectiva: **agrupación** de **ventanas** y **características** relacionadas que le permite a un desarrollador realizar un conjunto específico de tareas.



Cambiar perspectiva





Preparación del entorno

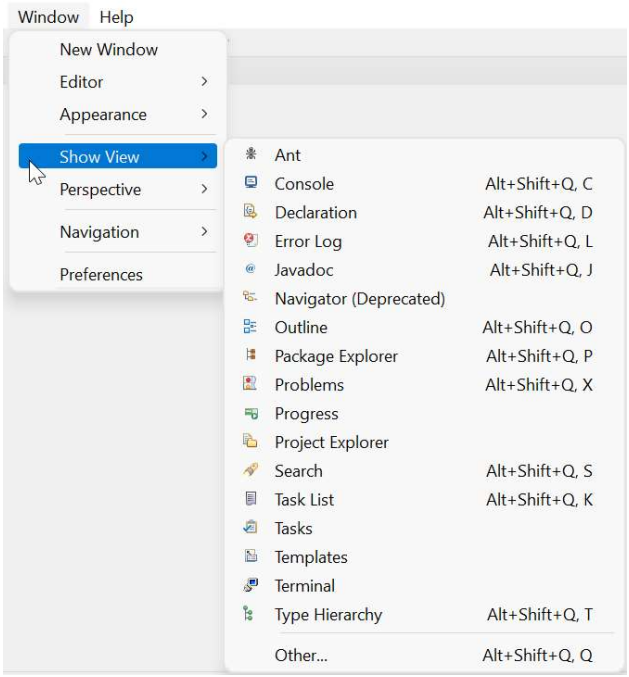
- **Sección A - Explorador de paquetes**: permite **navegar** por todos los archivos asociados dentro de un proyecto.
- **Sección B - Editor**: permite **modificar** el código fuente de Java o archivos basados en texto.
- **Sección C - Lista de tareas y esquema**: la lista de tareas se vincula a sistemas de seguimiento de **errores externos** y muestra las tareas **asignadas**. **OJO**: Lista de tareas ≠ tareas, las tareas se mostrarán en la sección D.
  La ventana outline muestra la **estructura** del archivo actualmente seleccionado en la ventana Editor.
- **Sección D - Panel de vistas con pestañas**: conjunto de vistas que se pueden mostrar u ocultar según preferencias.

Preparación del entorno

Pestañas por defecto:

- Problems: muestra cualquier mensaje de **error** o **advertencia** asociado con el código fuente encontrado en su proyecto.
- Javadoc: muestra la **documentación** de un elemento seleccionado en la ventana Editor.
- Declaration: informa sobre la declaración del objeto Java actualmente seleccionado en el Editor.
- Console: muestra la **salida** de su programa o cualquier **excepción de tiempo de ejecución** producida por su código.

Para añadir pestañas: Windows -> Show View



Recomendación: añadir pestaña Task, se pueden añadir tareas de forma **manual** o a través del comentario **//TODO**

Actividad 3

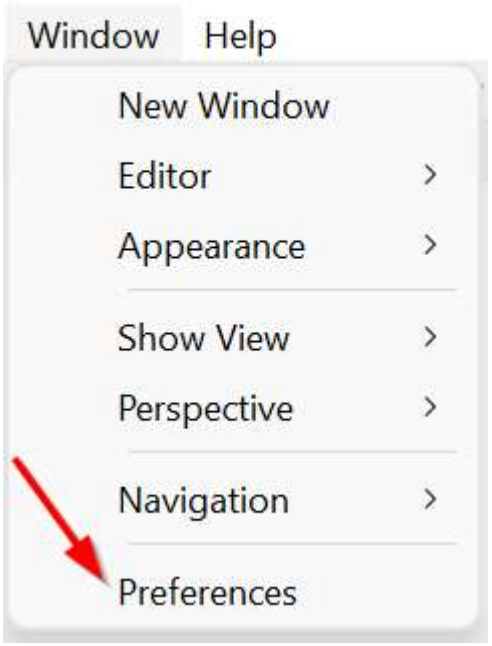
A partir del proyecto dado indicar:

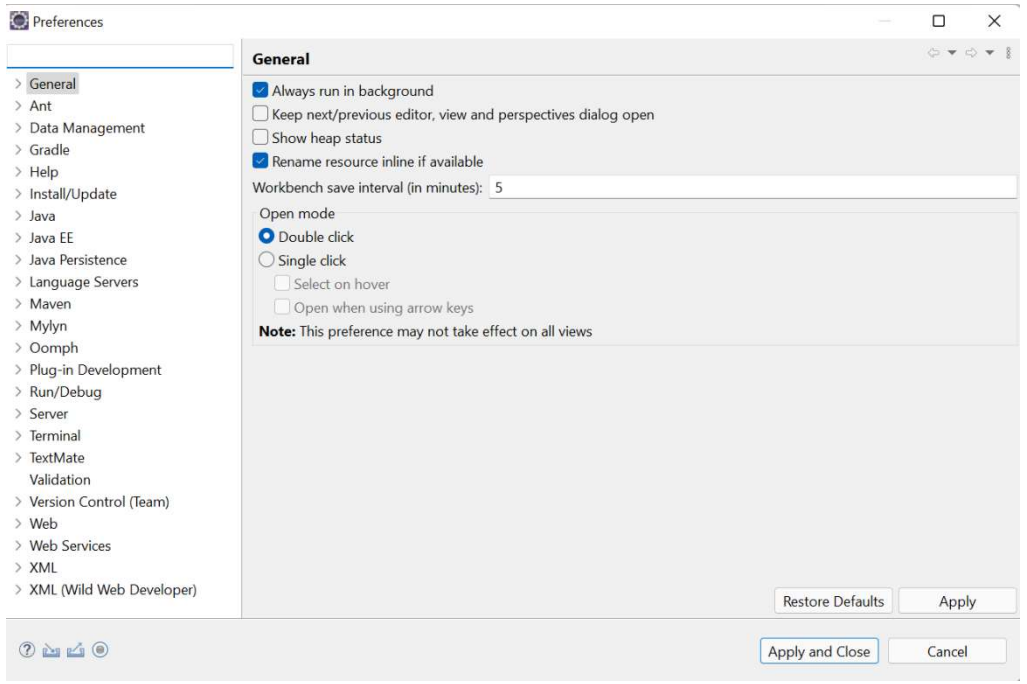
- Nombre de los archivos *.java del proyecto
- Estructura de la clase test
- Error existente en el proyecto, donde se encuentra localizado en el proyecto y en que línea.
- Número que aparece comentado en el archivo test.
- Obtener la documentación de la clase System usada en el archivo test.
- Obtener la declaración de la clase Impresion a través de la clase test.

Configuración del entorno

Configuración del entorno

Configuración: Window -> Preferences

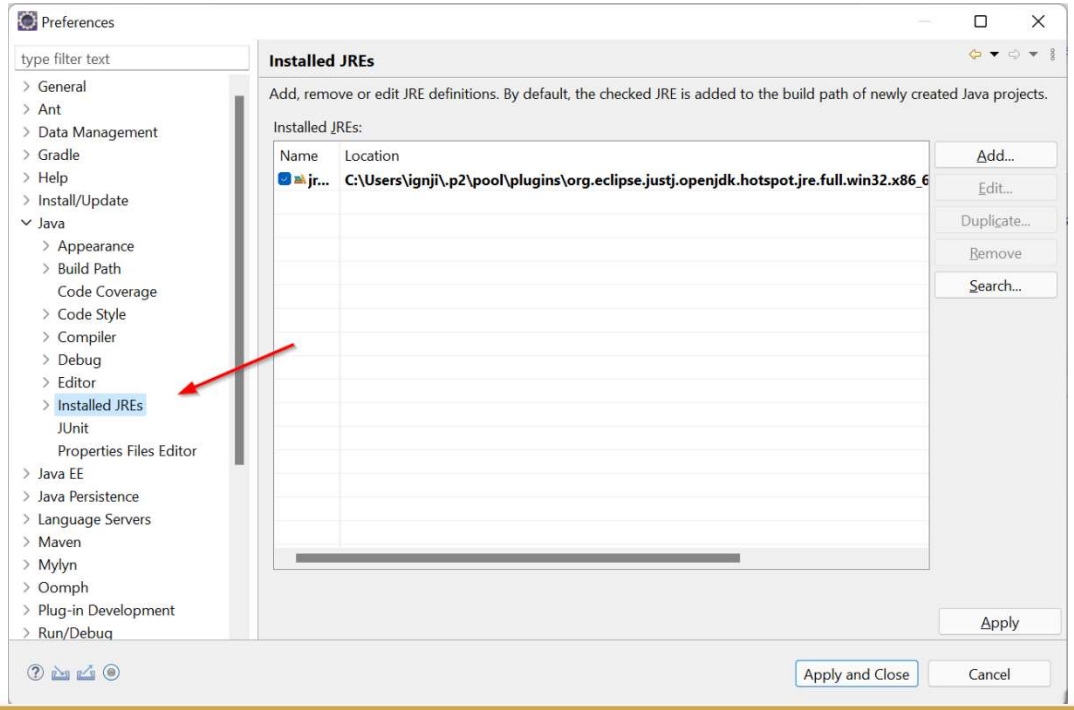




Configuración del entorno

Configuración JRE de Java: Java > Installed JREs

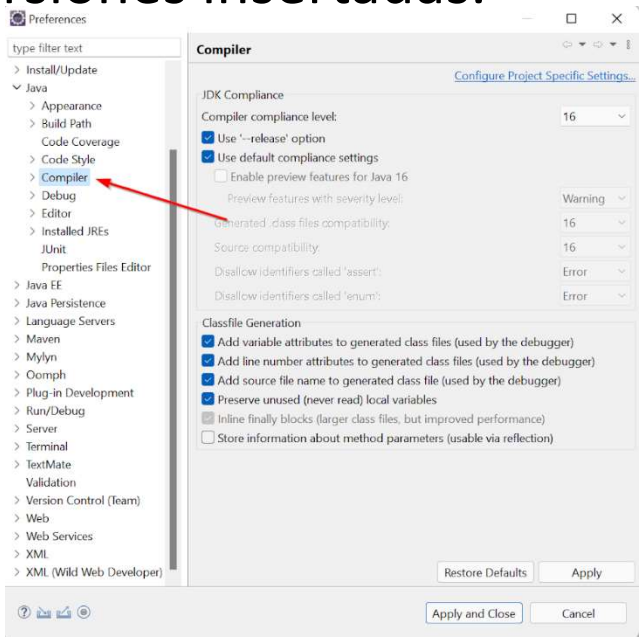
Eclipse detectará los JRE instalados en el ordenador, aunque existe la opción de añadirlo manualmente a través de “Add” localizando el JDK en nuestro equipo.



Configuración del entorno

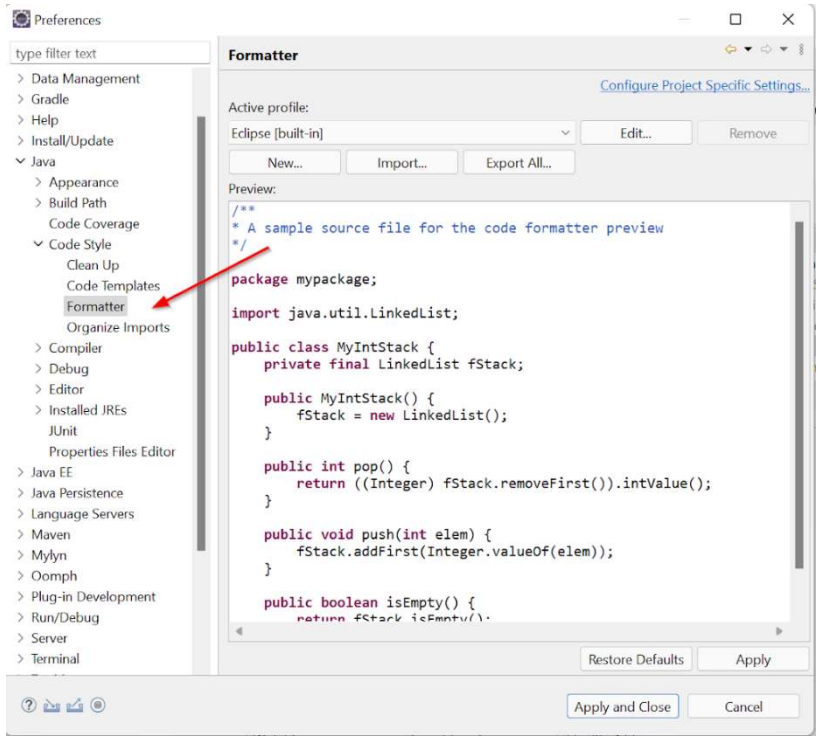
Configuración Compilador: Java > Compiler

Eclipse no usa el compilador JDK del ordenador, están "incrustados", por lo que si manualmente introducimos un JRE debemos tener en cuenta que dicho compilador debe de estar entre las versiones insertadas.



Configuración del entorno

Configuración Formatter: Java > Code Style > Formatter
Contiene las preferencias del espacio de trabajo para formatear el código fuente.



Otras configuraciones interesantes:

- Configurar navegador: General > Web Browser
- Configurar el tema: General > Appearance
- Configurar las variables de rutas: Java > Build Path > Classpath Variables
- Configurar librerías: Java > Build Path > User Libraries
- Configurar fuente: General > Appearance > Colors and fonts > Basic > Text Font

Actividad 4

Realizar una configuración inicial del IDE, para ello:

- Comprobar que la versión del JDK y JRE que usa es la 16.
- Seleccionar el perfil “Java Conventions” para el formateador del IDE.
- Seleccionar el navegador que usemos habitualmente.
- Según preferencias establecer un tema para el IDE y tamaño de texto para el editor.

Administración de plugins

Administración de plugins

Eclipse por si solo no proporciona una gran cantidad de funcionalidades, pero lo bueno que tiene este IDE es la facilidad en el desarrollo de plugins para **añadir funcionalidades**.

Estos plugins pueden ser ejecutados en cualquier **sistema operativo** compatible con Eclipse.

Administración de plugins

Podemos clasificar las distintas categorías de funcionalidades de módulos y plugins en los siguientes grupos:

1. **Construcción de código**: facilitan la labor de programación.
2. **Bases de datos**: ofrecen nuevas funcionalidades para el mantenimiento de las aplicaciones.
3. **Depuradores**: hacen más eficiente la depuración de programas.
4. **Aplicaciones**: añaden nuevas aplicaciones que nos pueden ser útiles.
5. **Edición**: hacen que los editores sean más precisos y más cómodos para el programador.
6. **Documentación de aplicaciones**: para generar documentación de los proyectos en la manera deseada.

7. **Interfaz gráfica de usuario**: para mejorar la forma de presentación de diversos aspectos del entorno al usuario.
8. **Lenguajes de programación y bibliotecas**: para poder programar bajo un Lenguaje de Programación que, en principio, no soporte la plataforma.
9. **Refactorización**: hacer pequeños cambios en el código para aumentar su legibilidad, sin alterar su función.
10. **Aplicaciones web**: para introducir aplicaciones web integradas en el entorno.
11. **Prueba**: para incorporar utilidades de pruebas al software.

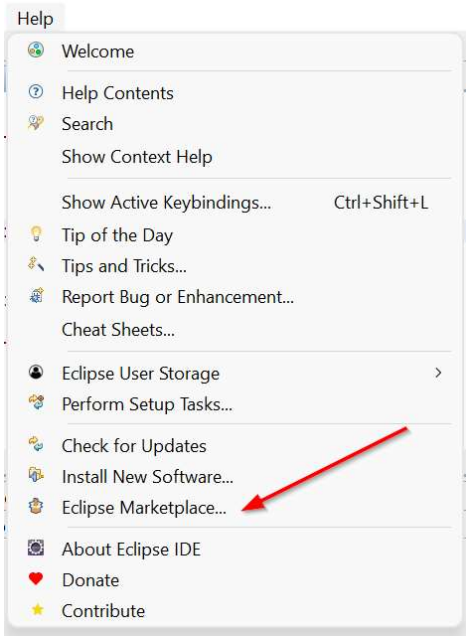
Administración de plugins

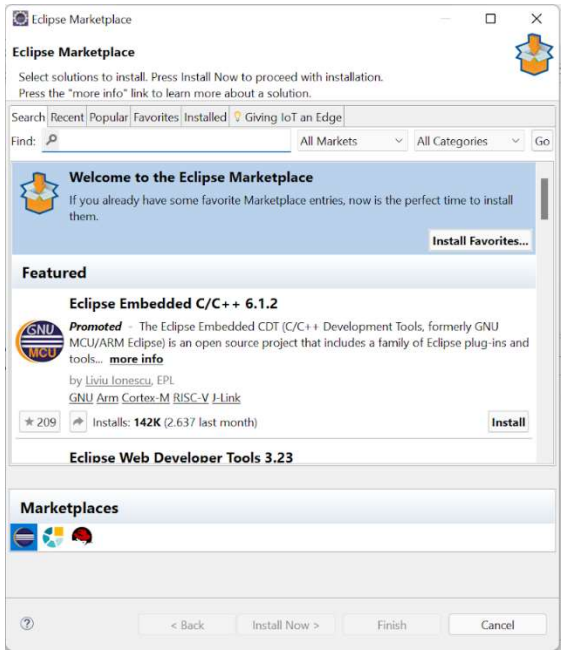
El IDE ofrece **dos formas** para la instalación de plugins:

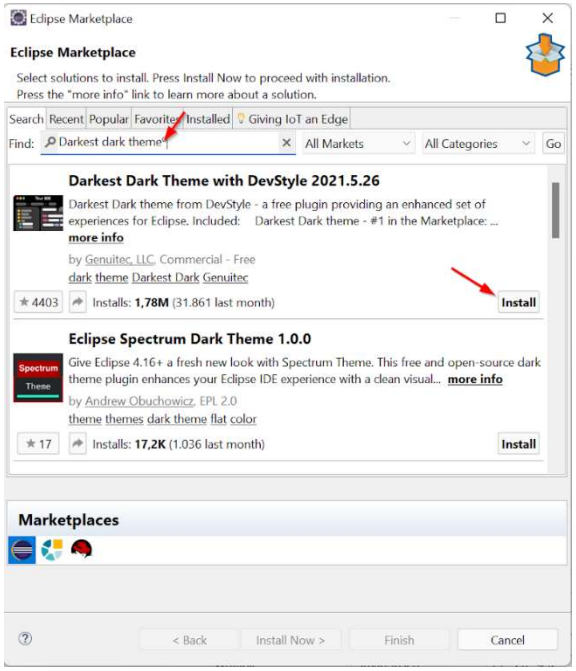
- Marketplace
- Ventana de software

Administración de plugins

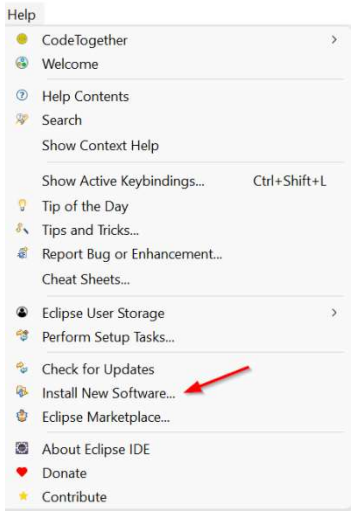
Marketplace: Help > Eclipse Marketplace

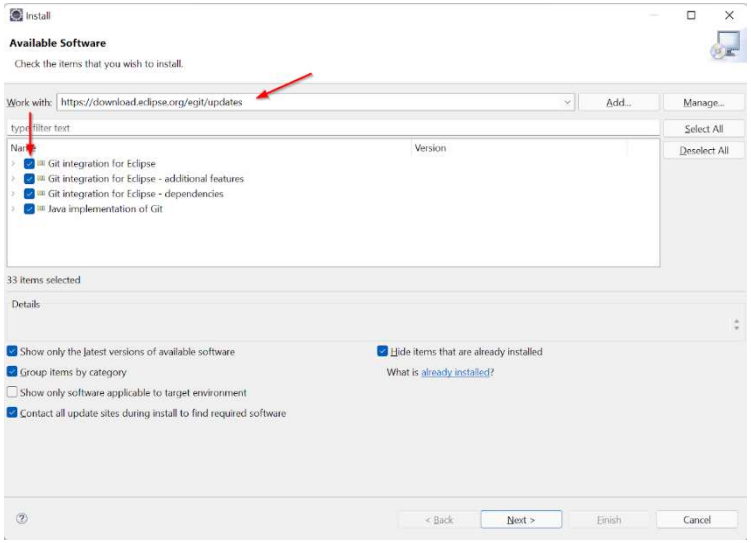


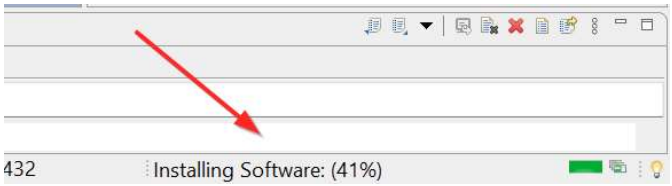




Ventana de software: Help > Install New Software





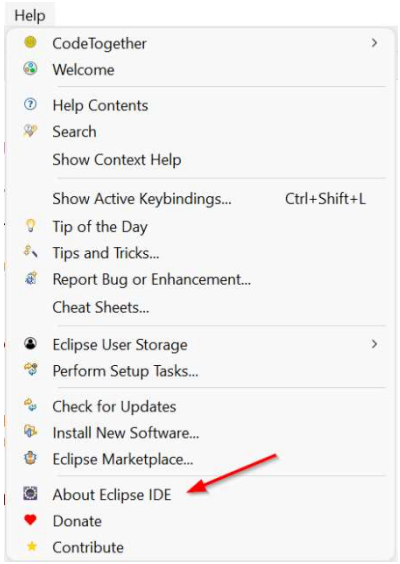


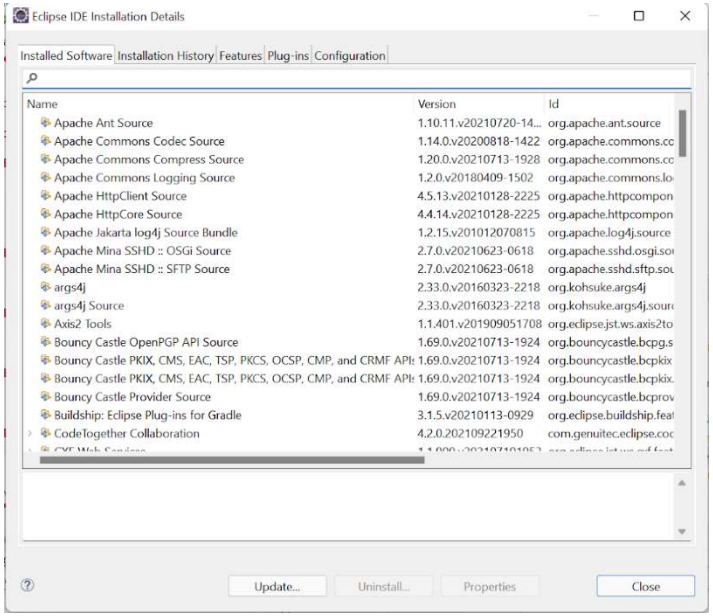
Administración de plugins

Ver plugins instalados: Help > About Eclipse IDE > Installation Details

Se puede ver: nombre, versión, identificador y proveedor.

También se pueden ver en la pestaña “Installed” del Marketplace



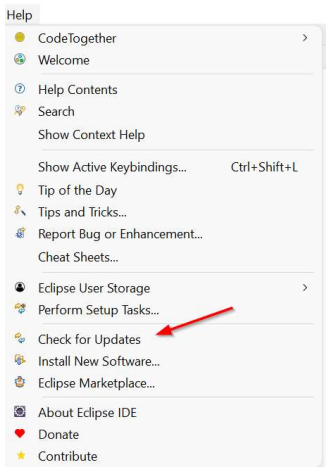


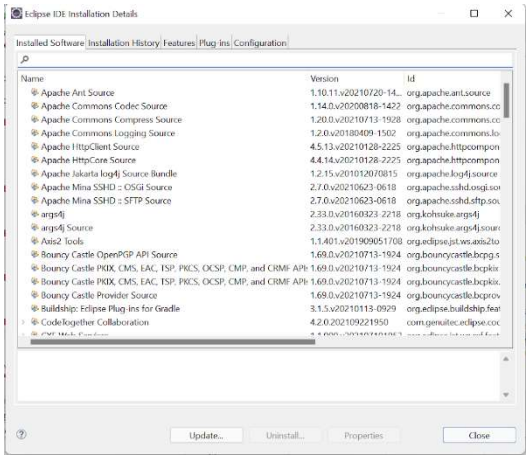
Administración de plugins

Actualizar plugins instalados:

Existen tres opciones:

1. Help > Check for Updates
2. Help > About Eclipse IDE > Installation Details y actualizar individualmente
3. Help > Marketplace y actualizar individualmente





Eliminar plugins instalados:

Exactamente igual que la actualización pero seleccionar Uninstall.

Actividad 5

Instalar los siguientes plugins de la forma indicada:

- A través de Marketplace el plugin “Darkest Dark”.
- A través de URL deinternet el plugin “Egit”.
- Averiguar que utilidad añaden ambos módulos a la plataforma y dónde se muestran situados en la ventana principal del IDE como en las preferencias.

Comprobar si existen actualizaciones de dichos plugins.

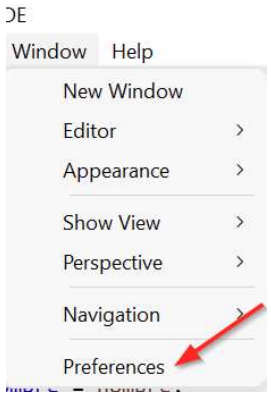
Desinstalar los dos plugins instalados.

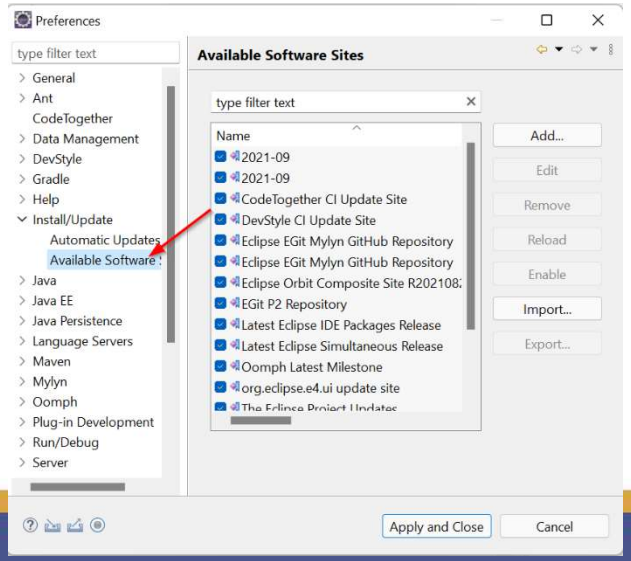
Actualización del entorno

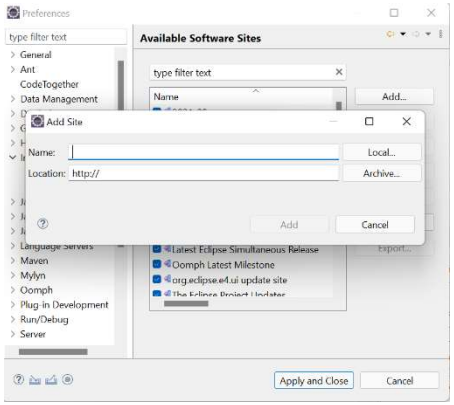
Actualización del entorno

Para **actualizar** a la siguiente versión **completa**: Window > Preferences > Install/Update > Available Software Sites

Hacer clic en “Add” y añadir la URL del nuevo repositorio de actualización de Eclipse. (Si quisiéramos actualizar a la última versión tendríamos que poner https://download.eclipse.org/releases/2021-09/)







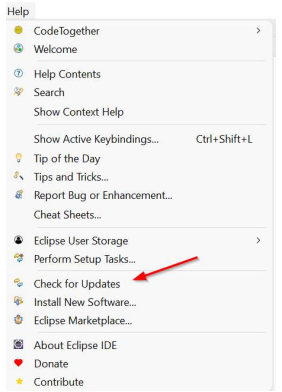
Actualización del entorno

Para **comprobar** si hay **actualizaciones**: Help > Check For updates

Como hemos comentado anteriormente, aquí puedes elegir los elementos que deseas actualizar.

Hay casos en los que será necesario realizar una instalación de Eclipse desde 0 debido a **incompatibilidad** con las tecnologías.

Las **instrucciones** de actualización vienen en el archivo readme_eclipse.html



Generación de ejecutables

Configuración

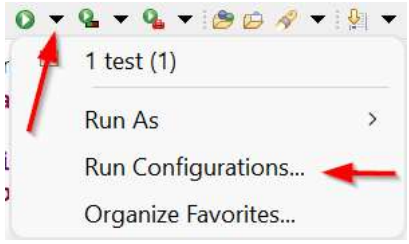
Generación de ejecutables

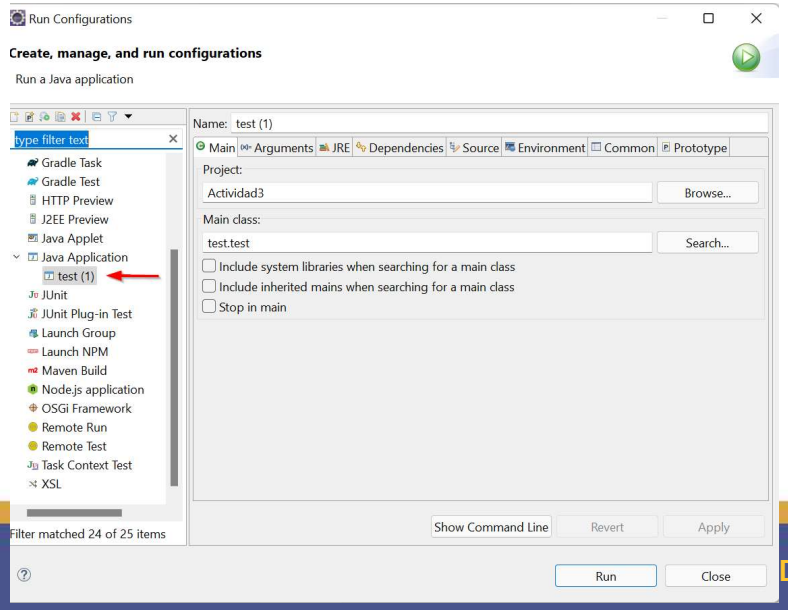
Antes de ejecutar un programa en Java debemos **configurar la ejecución** del mismo, para ello ir a Run > Run Configurations…

También se puede hacer click derecho en el proyecto > Run As > Run Configurations… (De esta forma muchas de las opciones estarán configuradas al proyecto en concreto)

La configuración estará dividida en las siguientes pestañas:

- Main
- Arguments
- JRE
- Dependencies
- Source
- Environment
- Common



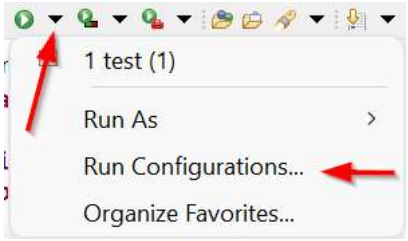


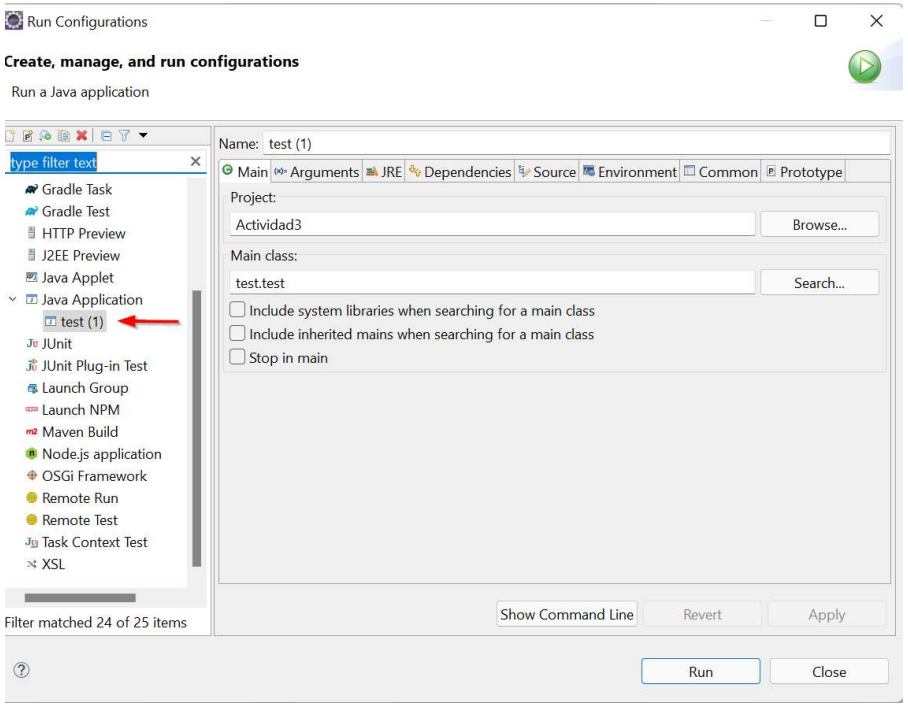
Generación de ejecutables

Antes de ejecutar un programa en Java debemos **configurar la ejecución** del mismo, para ello ir a Run > Run Configurations…

La configuración estará dividida en las siguientes pestañas:

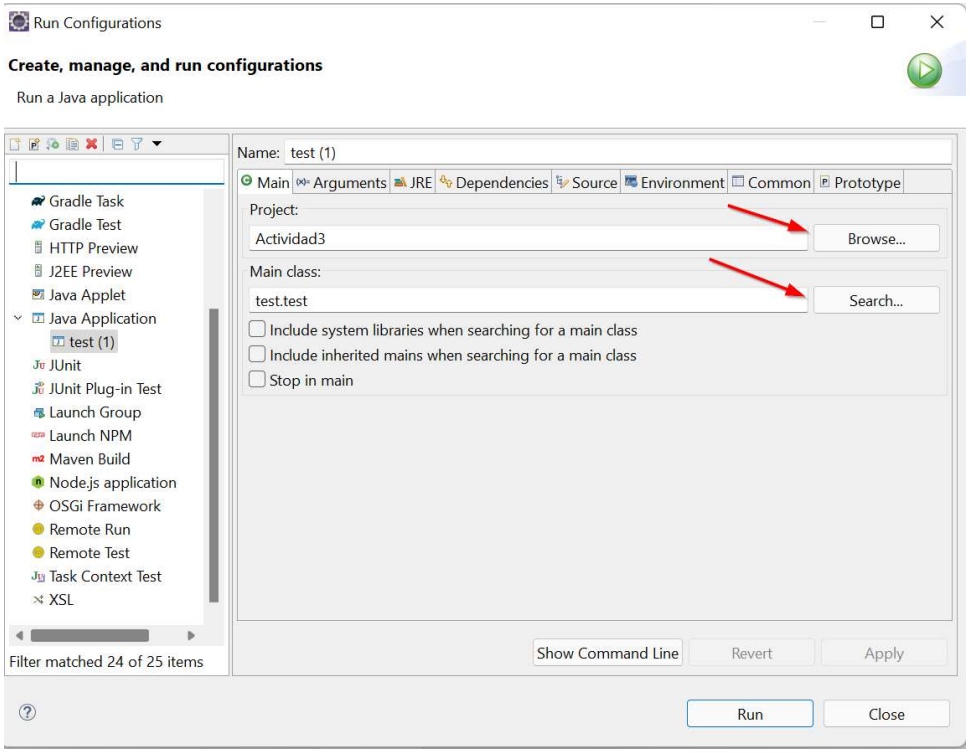
- Main
- Arguments
- JRE
- Dependencies
- Source
- Environment
- Common





Generación de ejecutables

Main: definiremos la clase **Main** de nuestro programa y la ubicación del **proyecto**.



Arguments: definiremos los **argumentos**:

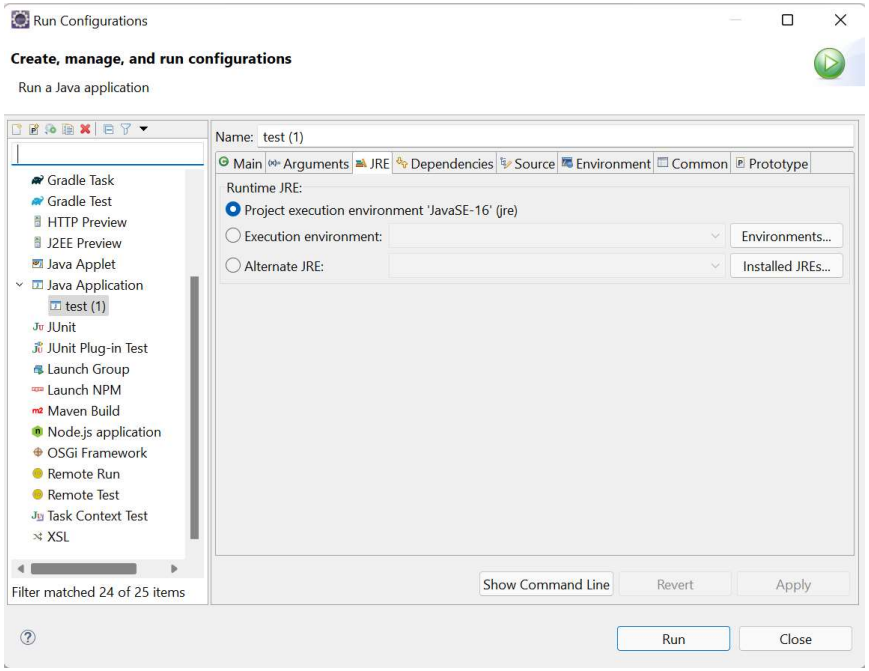
- Argumentos del programa
- Argumentos de la máquina virtual de Java

Ejemplo:
Realizamos un programa que dado dos números, nos calcula su suma. Aquí en argumentos del programa indicaríamos esos dos números separados por espacio: 3 5

Generación de ejecutables

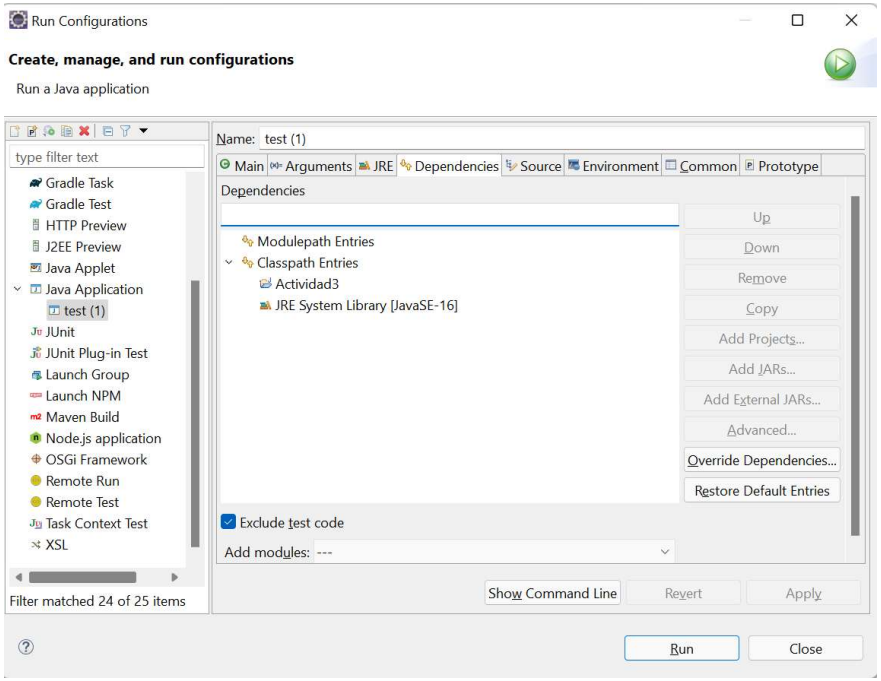
JRE: definiremos el **JRE** con el cual se va a ejecutar, pudiendo elegir:

- El JRE del entorno.
- Uno en concreto que tengamos instalado.
- Instalar uno para usarlo.



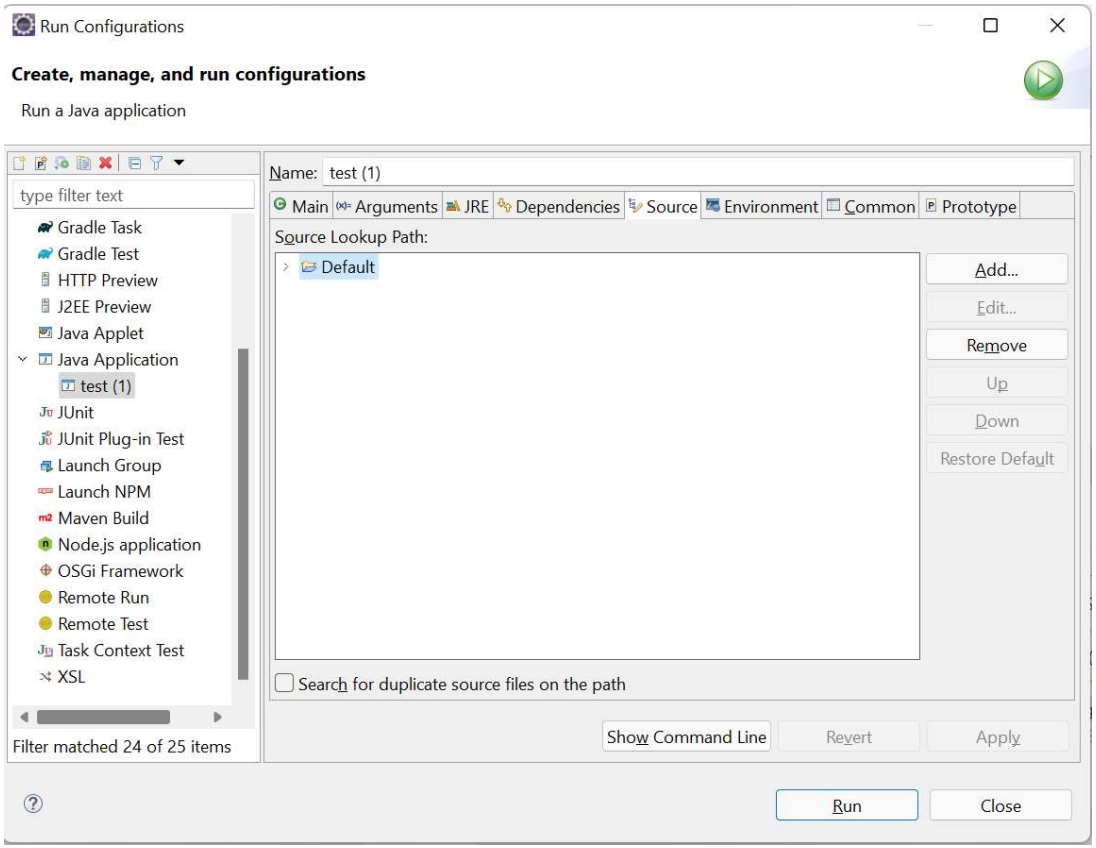
Generación de ejecutables

Dependencies: definiremos todo lo relacionado con las **librerías**, **módulos** y **dependencias** del proyecto.



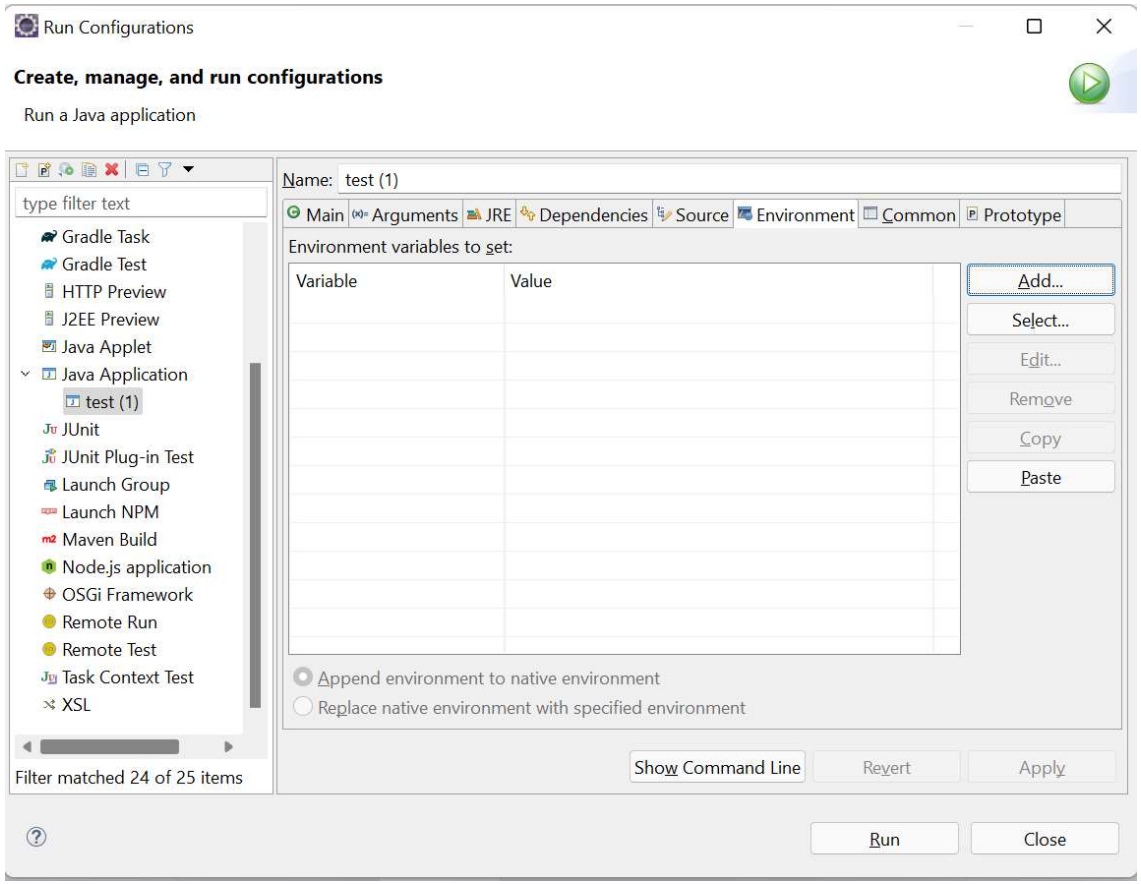
Generación de ejecutables

Source: definiremos la ruta de cierto **código** que queramos **añadir** a nuestro proyecto a través de una ruta.

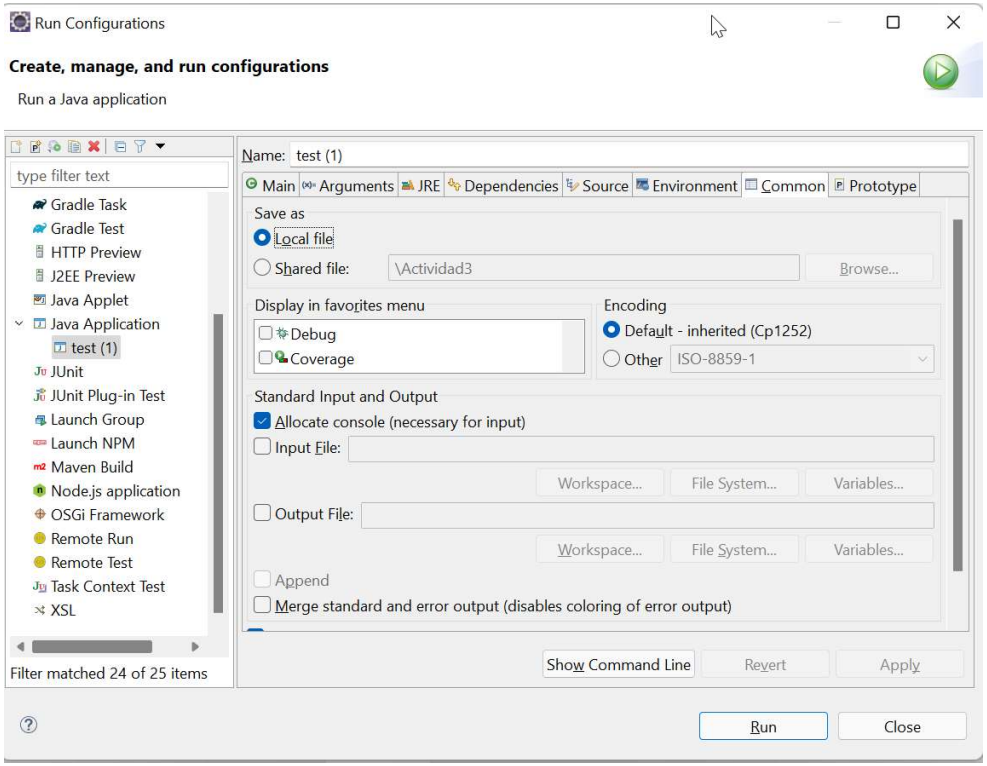


Generación de ejecutables

Environment: definiremos **variables** de nuestro **entorno**. Indicando su nombre y valor.



Common: definiremos **opciones comunes**, como la capacidad de asignar una consola para entrada y salida estándar.



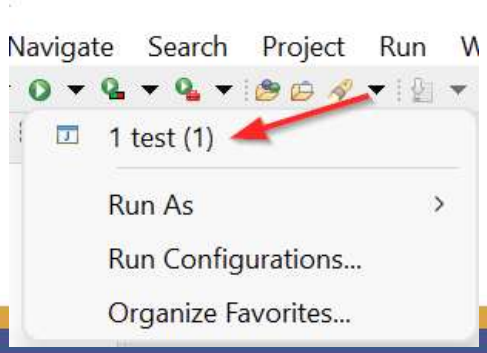
Generación y ejecución

Generación de ejecutables

Una vez tengamos nuestra configuración realizada, ya podremos **compilar** y **ejecutar** nuestro proyecto.

Para ello, podremos hacerlo tanto desde el icono de ejecución y seleccionando la configuración o Run As > Java Application, como haciendo click derecho en el proyecto > Run As > Java Application.

Si tenemos dos configuraciones diferentes para el mismo proyecto tendremos que usar la primera opción para indicarle expresamente cual es.



Actividad 6

Compilar y ejecutar la actividad 03 después de corregir el error.

Para corregir el error la línea deberá quedar así: “`this.dia = dia;`”

FIN
UNIDAD