Documentación del Uso del IDE

Documentación del uso del IDE - Eclipse

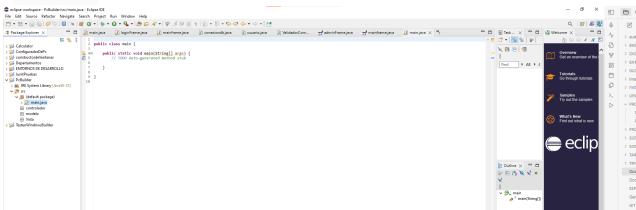
1. CONFIGURACIÓN INICIAL

Instalación de Eclipse

- 1. Descargar Eclipse IDE for Java Developers
- 2. Instalar Java JDK 8 o superior
- 3. Configurar JAVA_HOME en variables de entorno

Creación del proyecto

- 1. File \rightarrow New \rightarrow Java Project
- 2. Nombre: PCBuilder
- 3. Crear estructura de paquetes:
 - modelo
 - vista
 - controlador

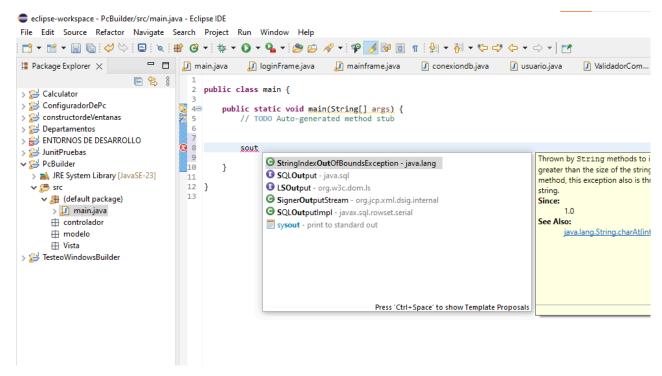


2. FUNCIONALIDADES UTILIZADAS

Edición de código

Autocompletado: Ctrl + Space

-



Formateo automático: Ctrl + Shift + F

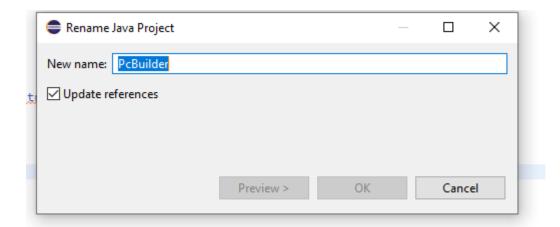
```
package controlador;
 3⊖ import modelo.usuario;
 4 import vista.adminFrame;
 6⊕ /**
    * Controlador para la vista del panel de administración.
 8
    * 
9
   * Este controlador se encarga de inicializar la interfaz de administrador y
10
    * pasarle el usuario autenticado.
11
12
13
    * @author Abdelghaffar EL AKHDAR
14
    * @version 1.6
15
    * @since 2025
16
17
18 public class adminController {
        private adminFrame frame;
19
20
        private usuario admin;
21
22⊝
23
24
         * Constructor que inicializa el frame de administrador.
25
        * @param usuario Usuario autenticado con rol de administrador.
26
27⊝
        public adminController(usuario usuario) {
28
           this.admin = usuario;
this.frame = new adminFrame();
29
30
            frame.setVisible(true);
31
32
33
```

Organizar imports: Ctrl + Shift + O

```
package controlador;

package controlador;
```

Renombrar: Alt + Shift + R

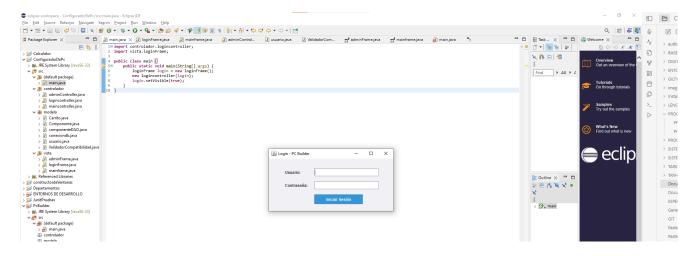


Asistentes de IA

- GitHub Copilot (si está instalado)
- Sugerencias de Eclipse: Quick Fix (Ctrl + 1)
- Generación automática:
 - Getters/Setters: Alt + Shift + S → Generate
 - Constructor: Alt + Shift + S → Generate Constructor

Pruebas

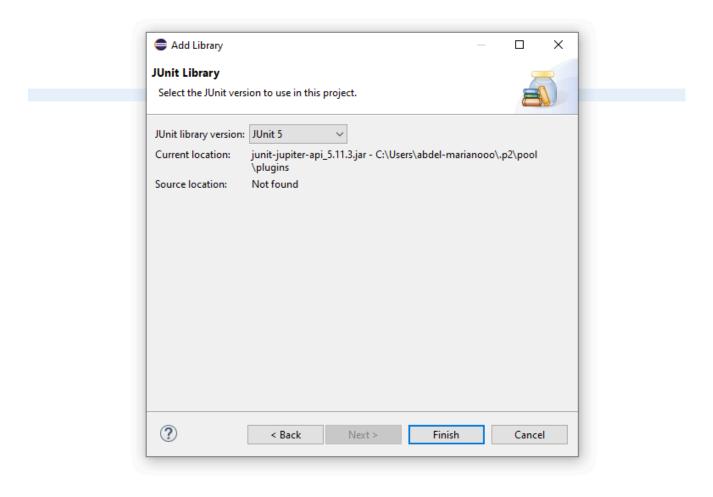
Ejecución: Click derecho → Run As → Java Application



Pruebas unitarias: JUnit (no implementado en este proyecto)

Primero añadimos Junit al Build Path

Haz clic derecho sobre tu proyecto ightarrow Build Path ightarrow Add Libraries...



-Creamos un paquete test y una clase Test

```
🖹 💲 🖰 🗖 🖟 main.java 🔑 loginFrame.java 🔑 mainframe.java 🖟 adminControl... 🖟 Componente.java 🖟 conexiondb.java 🗘 usuario.java
□ Package Explorer ×
> 🔛 Calculator
✔ 😂 ConfiguradorDePc
                                                                  3⊕ import static org.junit.jupiter.api.Assertions.*;[.
  > A JRE System Library [JavaSE-23]
                                                               9 public class ComponenteTest []
10
11 @Test
   > 🛋 JUnit 5
  @Test
public void testGetSetNombre() {
   Componente componente = new Componente();
   componente.setNombre("Intel Core i9");
   assertEquals("Intel Core i9", componente.getNombre());
}
     > 🔠 controlador
     > # modelo
     🗸 🌐 Testuni
     > ① ComponenteTest.java
> ② vista
   > 🛋 Referenced Libraries
                                                                         @Test
public void testGetSetPrecio() {
   Componente componente = new Componente();
   componente.setPrecio(499.99);
   assertEquals(499.99, componente.getPrecio(), 0.01); // mangen de error pana double
> 🛁 constructordeVentanas
  JunitPruebas
```

-Ahora vamos a correr el Test

```
<u>run window meip</u>
▼ 🚺 ▼ 🥦 ▼ 🤔 💪 🔗 ▼ 📪 📝 🕪 📵 🕥 📲 ▼ 🚰 ▼ 🎋 ▼ 🤝 🗘 🗘 ▼ 🗘 ▼ 🖠
                                  mainframe.java
                                                   adminControl...
                                                                        Componente.java
                                                                                             J conexiondb.iava
                                                                                                                J usuario.iava
                                                                                                                                Validado
  2 loginFrame
  Ju 3 CafeteraTest
                              jupiter.api.Assertions.*; ...
  Jv 4 CalculadoraTest
                              st {
  6 main (1)
                              tNombre() {
 □ 7 jfa
                              lente = new Componente();
 3 main
                              bre("Intel Core i9");
                                                           pmbre());
                          > Ju 1 JUnit Test Alt+Shift+X, T
      Run As
      Run Configurations...
      Organize Favorites...
                              tPrecio() {
             Componente componente = new Componente();
             componente.setPrecio(499.99);
             assertEquals(499.99, componente.getPrecio(), 0.01); // margen de error para double
```

-Resultados

y vemos que funcionan pues eh echo esto con casi todo pero da toda la pereza hacer capturas

```
eclipse-workspace - ConfiguradorDePc/src/Testuni/ComponenteTest.java - Eclipse IDE
<u>F</u>ile <u>E</u>dit <u>S</u>ource Refac<u>t</u>or <u>N</u>avigate Se<u>a</u>rch <u>P</u>roject <u>R</u>un <u>W</u>indow <u>H</u>elp
🖁 Package Explorer 🚜 JUnit 🗴 🗀 🗋 🖟 main,java 🖟 loginFrame.java 🖟 mainframe.java 🖟 adminControl... 🖟 Componente.java 🖟 conexiondb.java
               ↑ 🚾 🎝 🔠 🔍 🖟 🔳 🗒 🔻 1 package Testuni;
Finished after 0,16 seconds
                                                        -
3⊕ import static org.junit.jupiter.api.Assertions.*;[]
 Runs: 2/2
             Errors: 0
                               ■ Failures: 0
                                            9 public class ComponenteTest [
ComponenteTest [Runner: JUnit 5] (0,001 s)
                                                              @lest
public void testGetSetNombre() {
   Componente componente = new Componente();
   componente.setNombre("Intel Core i9");
   assertEquals("Intel Core i9", componente.getNombre());
     testGetSetPrecio() (0,000 s)
                                                              WTest
public void testGetSetPrecio() {
   Componente componente = new Componente();
   componente.setPrecio(499.99);
                                                                   assertEquals(499.99, componente.getPrecio(), 0.01); // margen de error para double
```

Depuración

- 1. Establecer breakpoints: Doble click en el margen izquierdo
- 2. Debug As → Java Application
- 3. Ventanas de depuración:
 - · Variables: Ver valores actuales
 - · Expressions: Evaluar expresiones
 - Step Over (F6): Siguiente línea
 - Step Into (F5): Entrar en método

Haber para hacer la depuración voy a crear un main en el que invocare la clase componente

```
> 📂 Calculator
                                                                                                                                                                                                                                                  package modelo;

→ 

    ConfiguradorDePc
    ConfiguradorDePc

                                                                                                                                                                                                                                                  public class ComponenteDebug {
   public static void main(String[] args) {
           > N JRE System Library [JavaSE-23]
           > M JUnit 5
                                                                                                                                                                                                                                                                                    Componente componente = new Componente();
          🗸 进 src
                                                                                                                                                                                                                                                                                        componente.setNombre("Procesador Ryzen 7");
componente.setDescripcion("8 núcleos, 16 hilos");
                     > # (default package)
                     > A controlador
                                                                                                                                                                                                                                                                                        componente.setPrecio(299.99);

✓ 

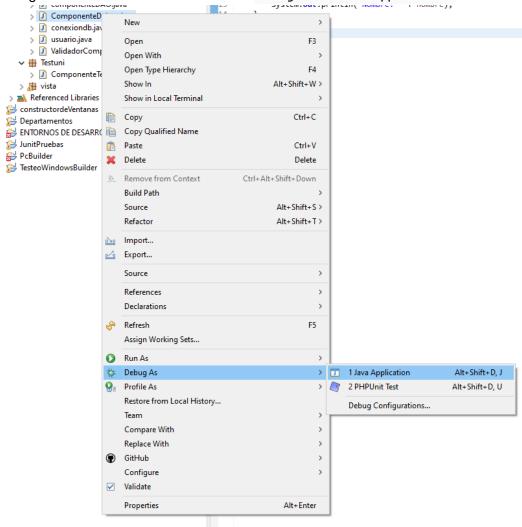
modelo

                                                                                                                                                                                                                                                                                        componente.setStock(10);
                              >   Carrito.java
                                                                                                                                                                                                                                11
                                >  Componente.java
                                                                                                                                                                                                                                                                                        String nombre = componente.getNombre(); // + aqui pon un breakpoint
System.out.println("Nombre: " + nombre);
                                >  omponenteDAO.java
                                                                                                                                                                                                                                13
                                >  ComponenteDebug.java
                                                                                                                                                                                                                                14
                                >  conexiondb.java
                                                                                                                                                                                                                         16
                                > 🚺 usuario.java
                                 > 🚺 ValidadorCompatibilidad.java
                      🗸 🌐 Testuni
                                > D ComponenteTest.java
                      > 🔠 vista
```

• Hay que hacer **doble clic en la línea** String nombre = componente.getNombre(); para poner un **breakpoint** cuando se pone el breakpoint sale una bola azul detras del numero de la linea.

```
String nombre = componente.getNombre(); // 
System.out.println("Nombre: " + nombre);
}
```

Luego hacer clic derecho en la clase → Debug As → Java Application



Eclipse se detendrá ahí → podrás ver el estado completo del objeto componente (nombre, precio, stock, etc.).

Usa F6 y F5 como mencionamos antes para avanzar o entrar en los métodos.

Refactorización

vamos a tomar como ejemplo la clase Componente otra vez

Extraer método: Alt + Shift + M

Mover clases: Arrastrar entre paquetes

Cambiar firma de método: Alt + Shift + C

package modelo;

/**

- Clase que representa un componente de PC en el sistema.
- Almacena toda la información necesaria de cada componente
- · incluyendo características técnicas para validar compatibilidad.
- @author Abdel

private int id;

@version 1.7*/public class Componente {

```
private String nombre;
private String descripcion;
private double precio;
private int stock;
private String tipo;
private String socket;
private String tipoRam;
private Integer potenciaRecomendada;
private String tamanoPlaca;
```

```
// A Constructor recomendado
public Componente(int id, String nombre, String descripcion, double precio, int
stock, String tipo) {
   this.id = id;
   this.nombre = nombre;
   this.descripcion = descripcion;
   this.precio = precio;
   this.stock = stock;
   this.tipo = tipo;
}
// Getters y setters con nombres corregidos
public int getId() { return id; }
public void setId(int id) { this.id = id; }
public String getNombre() { return nombre; }
public void setNombre(String nombre) { this.nombre = nombre; }
public String getDescripcion() { return descripcion; }
public void setDescripcion(String descripcion) { this.descripcion = descripcion; }
public double getPrecio() { return precio; }
public void setPrecio(double precio) { this.precio = precio; }
public int getStock() { return stock; }
public void setStock(int stock) { this.stock = stock; }
public String getTipo() { return tipo; }
public void setTipo(String tipo) { this.tipo = tipo; }
public String getSocket() { return socket; }
public void setSocket(String socket) { this.socket = socket; }
public String getTipoRam() { return tipoRam; }
public void setTipoRam(String tipoRam) { this.tipoRam = tipoRam; }
public Integer getPotenciaRecomendada() { return potenciaRecomendada; }
public void setPotenciaRecomendada(Integer potenciaRecomendada) {
this.potenciaRecomendada = potenciaRecomendada; }
```

```
public String getTamanoPlaca() { return tamanoPlaca; }
public void setTamanoPlaca(String tamanoPlaca) { this.tamanoPlaca = tamanoPlaca; }
// 🔗 Opcional: método toString() para depuración
@Override
public String toString() {
    return "Componente{" +
            "id=" + id +
            ", nombre='" + nombre + '\'' +
            ", descripcion='" + descripcion + '\'' +
            ", precio=" + precio +
            ", stock=" + stock +
            ", tipo='" + tipo + '\'' +
            ", socket='" + socket + '\'' +
            ", tipoRam='" + tipoRam + '\'' +
            ", potenciaRecomendada=" + potenciaRecomendada +
            ", tamanoPlaca='" + tamanoPlaca + '\'' +
            1}1;
}
```

Refactorización aplicada a la clase Componente:

Se corrigieron los nombres de los métodos getters y setters para seguir la convención estándar de Java (por ejemplo, gettipo() se renombró a getTipo()). Se añadió un constructor para facilitar la creación de instancias y un método toString() útil durante la depuración. Estas mejoras no cambian la funcionalidad del código, pero lo hacen más legible, mantenible y preparado para futuras ampliaciones.

VENTAJAS DEL IDE

}

1. 🔍 Detección de errores en tiempo real

- El IDE subraya errores de sintaxis (rojo) o advertencias (amarillo) mientras escribes el código.
- Por ejemplo, si escribes gettamanoPlaca() sin definir el método, te lo marcará al momento.
- Así evitas compilar con errores y pierdes menos tiempo buscando fallos.

2. Q Autocompletado inteligente

- Mientras escribes código, el IDE te sugiere métodos, atributos y clases automáticamente.
- Por ejemplo: escribes comp. y te sugiere getId(), getTipo(), setPrecio(), etc.
- Ahorra tiempo, evita errores y ayuda cuando no recuerdas exactamente el nombre de algo.

3. 💋 Navegación rápida entre clases y métodos

- Puedes hacer Ctrl + click en cualquier clase, método o variable para ir a su definición.
- Útil cuando trabajas con muchas clases como Componente, ValidadorCompatibilidad, etc.
- También puedes buscar archivos o clases con Ctrl + Shift + R (Eclipse) o Double Shift (IntelliJ).

4. Tintegración con Git

- Puedes conectar el proyecto a Git directamente desde el IDE.
- Permite hacer commits, ver el historial, comparar versiones y hacer push/pull sin usar la terminal.
- Ideal para control de versiones en equipo o tener copias de seguridad de tus avances.

5. Gestión de librerías (como MySQL Connector)

- Puedes añadir fácilmente bibliotecas externas (JAR) como el MySQL Connector.
- El IDE te permite añadirlas al classpath del proyecto sin complicaciones.
- Por ejemplo: conectar con la base de datos para cargar los Componente desde MySQL.

¿Cómo mostrarlo en clase?

- Haz capturas de pantalla reales mostrando cada ventaja en acción con tu proyecto.
- Por ejemplo:
 - Un error subrayado en rojo mientras programas.
 - El autocompletado mostrando métodos de Componente.
 - El explorador de clases con Componente.java, adminFrame.java, etc.
 - El panel de Git mostrando los cambios.
 - El proyecto mostrando que incluye mysql-connector-java.jar.

4. CONFIGURACIÓN ESPECÍFICA DEL PROYECTO

Agregar MySQL Connector

- 1. Click derecho en proyecto → Build Path → Configure Build Path
- 2. Libraries → Add External JARs

3. Seleccionar mysql-connector-java-9.3.0.jar

```
    ConfiguradorDePc
    IRE System Library [JavaSE-23]
    IUnit 5
    src
    (default package)
    modelo
    modelo
    Testuni
    introduction
    Referenced Libraries
    mysql-connector-j-9.3.0.jar - C:\Users\abdel-marianooo\Nextcloud
    bin
```

Configuración de codificación

- $\bullet \quad \text{Window} \to \text{Preferences} \to \text{General} \to \text{Workspace}$
- Text file encoding: UTF-8