

Java en tiempos de IA

Tomando decisiones más allá del código

ING. Yoel David Correa Duke
yoeldcd@uci.cu

Corporación Universitaria
para el Desarrollo de Internet A.C

problema

diseño

criterio

solución

Objetivo

Demostrar el valor de conocer los fundamentos de la **Programación Orientada a Objetos** a la hora de tomar **decisiones de diseño** durante la construcción de soluciones de software en JAVA con asistencia de Inteligencia Artificial.

Preguntas:

- ¿Que es un Software?
- ¿Qué es un lenguaje de Programación?
- ¿Que es JAVA?
- ¿Cómo tomar buenas decisiones al construir Software con JAVA?

¿Qué tarea te gustaría dejar de hacer manualmente?

Tomate unos segundos y piensa:

**“Cada semana
tengo que...”**



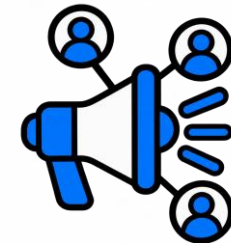
Registrar ...



Clasificar ...



Analizar ...



Comunicar ...

Ya programabas y aún no lo sabías!!!

Una tarea es un conjunto de pasos finitos que se ejecuta durante un intervalo de tiempo finito para alcanzar un objetivo de acuerdo con un conjunto de normas y contratos definidos.



Dos modos de representar la información.

Según la complejidad con que la información es tratada se puede representar:

Mediante **DATOS**

Cantidades



Métricas



Palabras

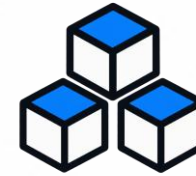


Señales



Mediante **REPRESENTACIONES**

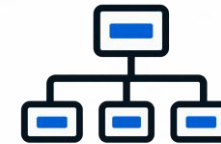
Elementos



Conceptos



Estructuras



Se deben tomar **DECISIONES** que conduzcan a **SOLUCIONES**.

01

Definir Requisitos

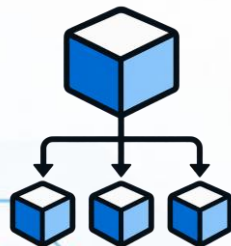
Convertir una necesidad en una **regla verificable**



02

Diseñar

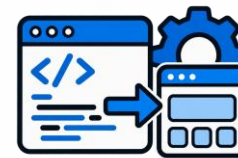
Separar el trabajo en piezas más pequeñas con sus **propia responsabilidades**.



03

Implementar

Construir una **solución funcional** alineada a los **requisitos** por medio de un **lenguaje de programación**.



04

Verificar

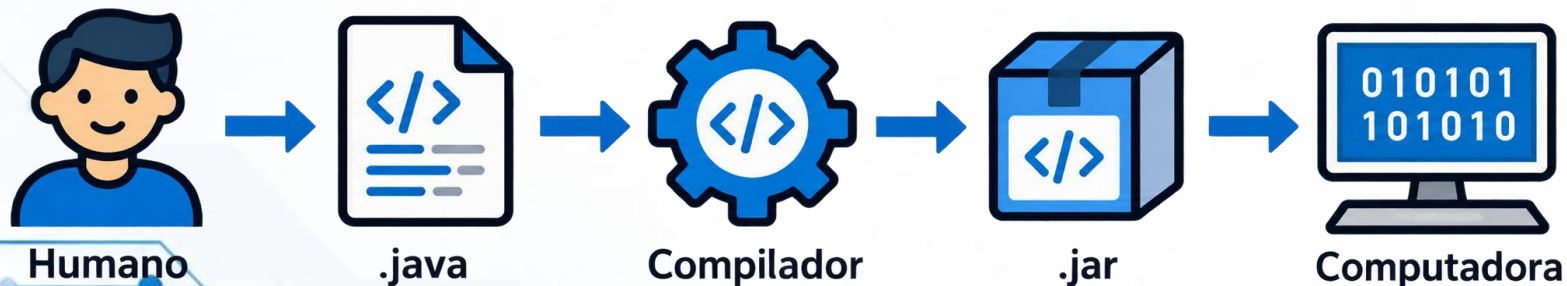
Capturar evidencia que nos **permita medir el éxito** de la solución.
(resiliencia, confiabilidad, seguridad).



Se debe lograr que la **MÁQUINA** se comporte tal como lo se **NECESITA**.

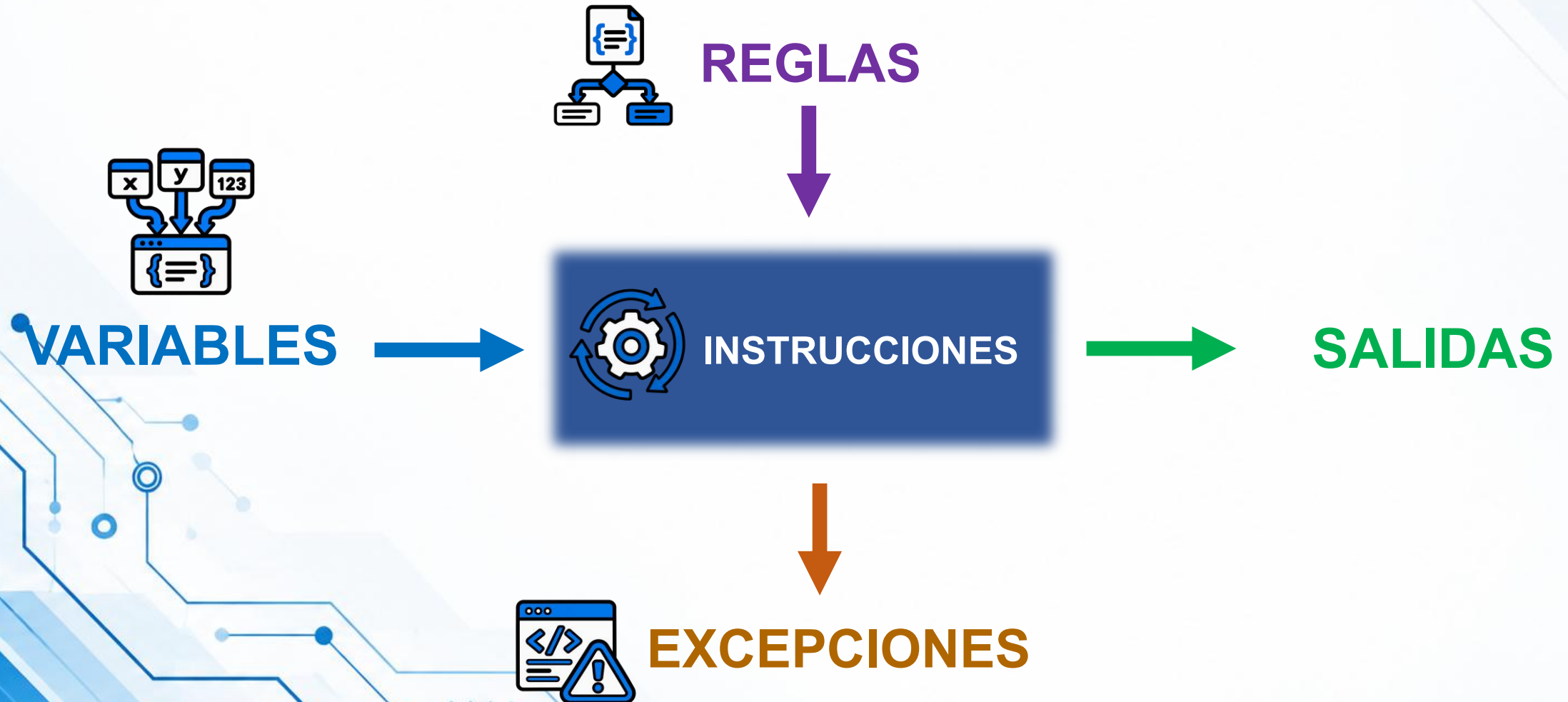
Se requiere un medio de comunicación para **TRANSMITIR** las **INTENCIONES** del **HUMANO** a la **MÁQUINA** que cuente con:

- Un Conjunto de Términos (Símbolos)
- Una Gramática (Sintaxis)
- Un Conjunto de Reglas (Semántica)



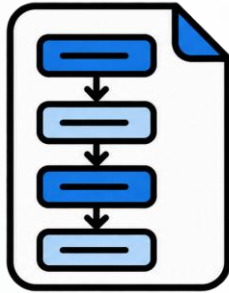
Ya programabas y aún no lo sabías!!!

Un programa es una tarea que se ejecuta algorítmicamente sobre una computadora, dirigida por un lenguaje que esta entiende para alcanzar un resultado.



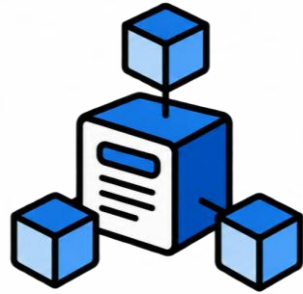
Paradigmas de la Programación.

Al programar se pueden seguir diferentes estilos según los aspectos en que se enfoca el problema.



IMPERATIVA

- Enfocada en **procedimientos**.
- Se ejecuta **paso a paso**.
- Modifica **estado** de los datos.
- Ejerce **control** mediante condiciones y ciclos.



POO

- Enfocada en **entidades**.
- Modela **propiedades & comportamientos**.
- Establece **contratos**.
- Encapsula **responsabilidades**.
- Relaciona **elementos**.



FUNCIONAL

- Enfocada en **transformaciones**.
- Opera sobre **funciones**.
- Prefiere **datos inmutables**.
- Altamente **Paralelizable**



El Lenguaje de Programación JAVA.

JAVA es un lenguaje de programación ampliamente utilizado por instituciones y empresas, gracias a su **versatilidad** y **resiliencia**.

TRAJECTORIA

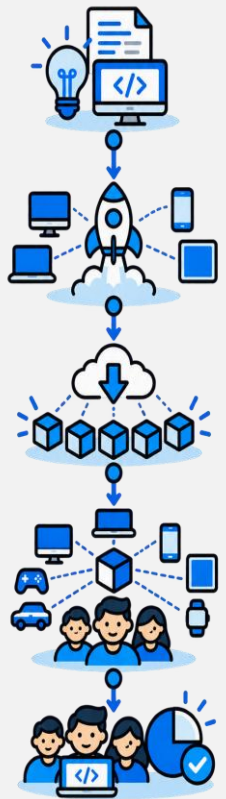
1991: Nace como proyecto experimental en **Sun Microsystems (James Gosling)**.

1995: Lanzamiento oficial de Java como **lenguaje multiplataforma**.

1998: Java supera los **2 millones** de descargas.

2004: **4 millones** de desarrolladores y **1.500 millones** de dispositivos.

2026: **+90 %** de instituciones bancarias utilizan Java en las transacciones.



El Lenguaje de Programación JAVA.

JAVA es un lenguaje de POO que permite desarrollar soluciones sobre los tres paradigmas.

CARÁCTERÍSTICAS

- Sigue el principio de **Escribelo una vez, Ejecútalo donde sea** (WORA: Write Once, Run Anywhere)
- Los programas se ejecutan sobre una **máquina virtual (JVM)**
- El código **.java** se transforma (**se compila**) en un paquete **.jar** que la JVM entiende.
- Gestiona automáticamente el uso de memoria mediante un **recolector de basura GC**.
- Incluye un amplio catalogo de **herramientas de desarrollo** y una **comunidad de colaboradores** activa.



Los datos en JAVA.

Java es opera con tipos definidos y ofrece mecanismos para definir tipos de datos nuevos. Los tipos se agrupan en dos conjuntos:

Tipos **SIMPLES** (PRIMITIVOS)



Números Enteros (byte, int)



Números Decimales (float, double)

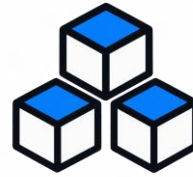


Cadenas de texto (char, String)



Valores de Verdad (bool)

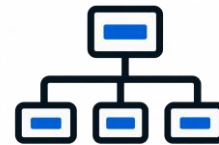
Tipos **COMPUESTOS** (ESTRUCTURALES)



Clases y Objetos (class)



Contratos (Interface)



Colecciones (ArrayList, Stack, Queue)

Estructuras de Control y Ciclos en JAVA.

Java posee varias estructuras que permiten determinar como y cuando ejecutar ciertas partes del programa.

Operaciones Condicionadas

```
double totalCompra = 120.50;
boolean esClienteVip = true;

// Si es VIP y compra más de $100, aplica descuento
if (esClienteVip && totalCompra > 100.0) {
    totalCompra = totalCompra * 0.85; // 15% de descuento
} else {
    totalCompra = totalCompra; // Precio normal
}
```

Operaciones Repetitivas

```
double precioUnidad = 2.50;
double subtotal = 0.0;

// Simulamos pasar 5 productos por el cliente
for (int item = 1; item <= 5; item++) {
    subtotal = subtotal + precioUnidad;
}
```

Manejo de Errores

```
int totalFactura = 100;
int cantidadClientes = 0; // Error: nadie a quién cobrar
int pagoPorPersona = 0;

try {
    // Intentamos hacer una división peligrosa
    pagoPorPersona = totalFactura / cantidadClientes;
} catch (ArithmeticException e) {
    // Si falla, el programa no se cierra; asignamos un valor por defecto
    pagoPorPersona = 0;
}
```

En JAVA las clases son las plantillas que describen un objeto.

```
public class Persona {  
    protected String nombre;  
    protected int codigo;  
  
    public Persona(String nombre, int codigo) {  
        this.nombre = nombre;  
        this.codigo = codigo;  
    }  
}  
  
public class Trabajador extends Persona {  
    private double salario;  
  
    public Trabajador(String nombre, int codigo, double salario) {  
        super(nombre, codigo);  
        this.salario = salario;  
    }  
}
```

Propiedades (Atributos)

- TIPO
- NOMBRE (identificador)
- VISIBILIDAD

Comportamientos (Métodos)

- ENTRADAS (parámetros)
- LÓGICA (implementación)
- SALIDA (tipo de retorno)
- VISIBILIDAD

Método Especial: Constructor

En JAVA la POO se expresa mediante las clases y sus relaciones.

HERENCIA

Un objeto de la clase B hereda las propiedades y comportamientos que tiene un objeto de la clase A

```
public class Persona {
    protected String nombre;
    protected int id;

    public Persona(String nombre, int id) {
        this.nombre = nombre;
        this.id = id;
    }
}

public class Trabajador extends Persona {
    private double salario;

    public Trabajador(String nombre, int id, double salario) {
        super(nombre, id);
        this.salario = salario;
    }
}
```

POLIMORFISMO

Un objeto de la clase B establece como realizar una acción que la clase A solo sabe que puede realizar

```
public abstract class Personal {
    protected String nombre;

    public Personal(String nombre) {
        this.nombre = nombre;
    }

    public abstract double calcularSueldo();
}

public class Asistente extends Personal {
    private double sueldoFijo;

    public Asistente(String nombre, double sueldoFijo) {
        super(nombre);
        this.sueldoFijo = sueldoFijo;
    }

    @Override
    public double calcularSueldo() {
        return this.sueldoFijo;
    }
}
```

ESPECIALIZACION

Un objeto de la clase B perfecciona la forma en que un objeto de la clase A realiza cierta acción.

```
public class Empleado {
    public void trabajar() {
        System.out.println("Realizando tareas generales...");
    }
}

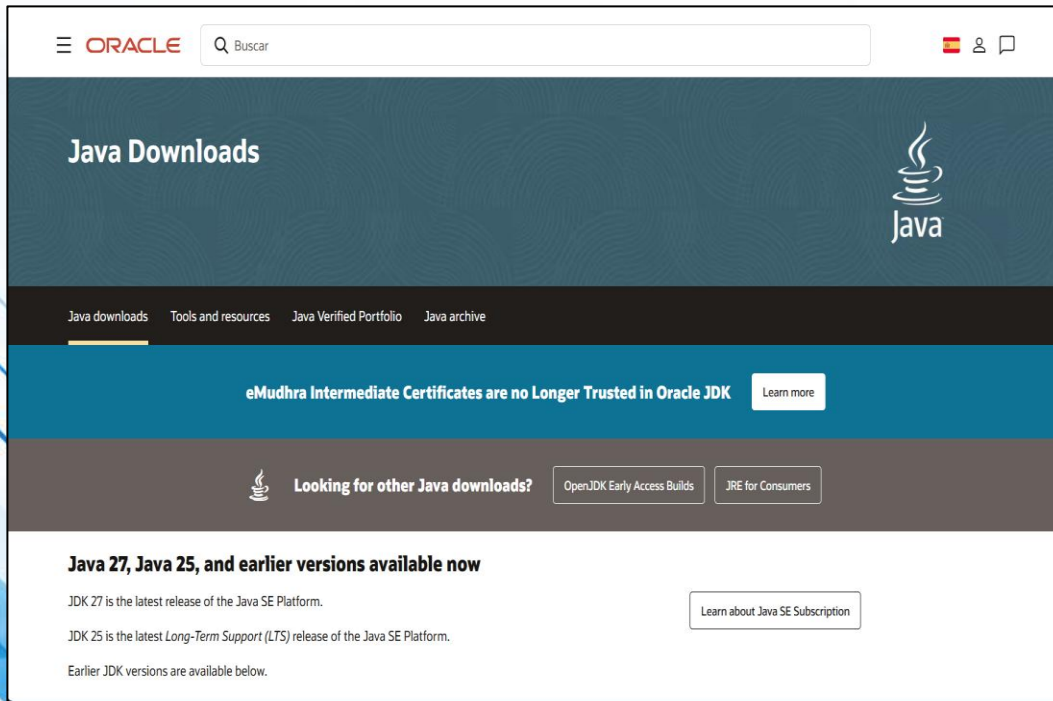
public class Vendedor extends Empleado {
    @Override
    public void trabajar() {
        System.out.println("Atendiendo y vendiendo a clientes...");
    }
}

public class Gerente extends Empleado {
    @Override
    public void trabajar() {
        System.out.println("Supervisando el equipo y metas...");
    }
}
```

Como utilizar JAVA

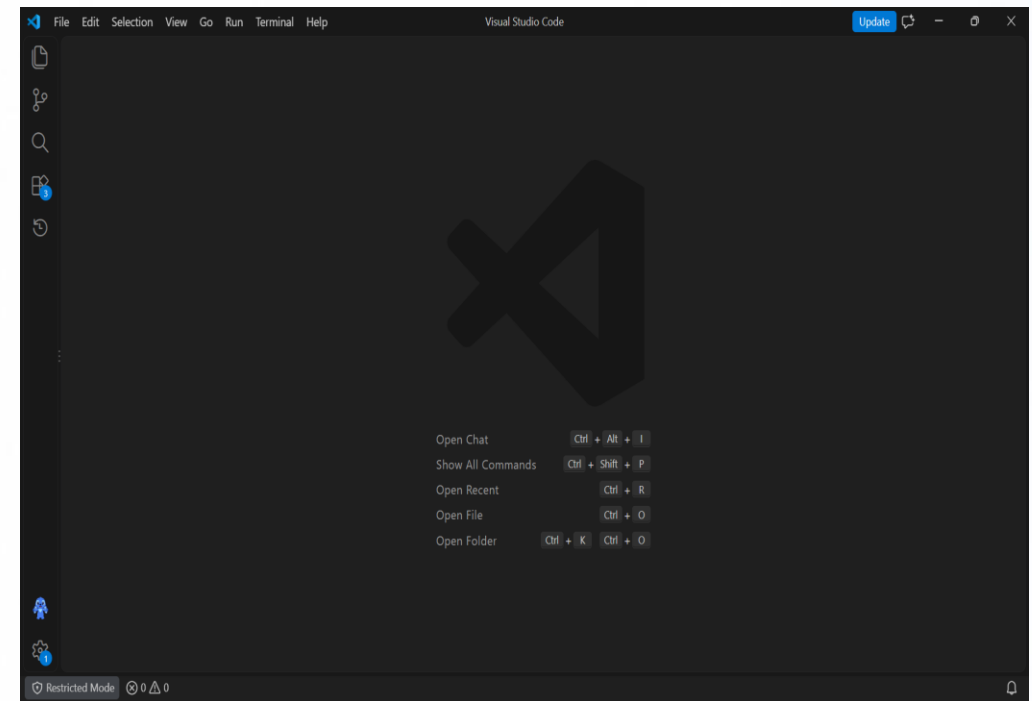
Para desarrollar soluciones con JAVA necesitas disponer de dos herramientas en tu computadora:

JDK (compilador y la JVM)



<https://www.oracle.com/es/java/technologies/downloads/>

Un editor de código (VSCode)



<https://code.visualstudio.com/download>

Nuevas Oportunidades:

El desarrollo de tecnologías emergentes como la **Inteligencia Artificial (IA)** hoy un brinda un nuevo conjunto de aplicaciones enfocadas en la construcción y perfeccionamiento de las soluciones de Software.

Para construir un **PRODUCTO**



Proponer Diseños



Implementar Prototipos



Ejecutar pruebas

Para ofrecer un **SERVICIO**



Analizar contenido



Resumir Información



Asistir la toma decisiones

Pregunta Central

¿Cómo tomar buenas decisiones de **DISEÑO** durante la construcción de **SOLUCIONES** con asistencia de IA mediante el conocimiento de los **PRINCIPIOS** de la **Programación Orientada a Objetos**?

MANOS A LA OBRA...

Conoce el contexto antes de definir los requisitos.

- Analiza en detalle el problema que necesita solución.
- Define los requisitos como resultados esperados.
- Establece criterios de verificación a corto y mediano plazo.
- Distingue los datos estáticos de los derivados.
- Distribuye el trabajo entre componentes pequeños.
- Aplica una estrategia de trabajo incremental



Conoce el contexto antes de definir los requisitos.

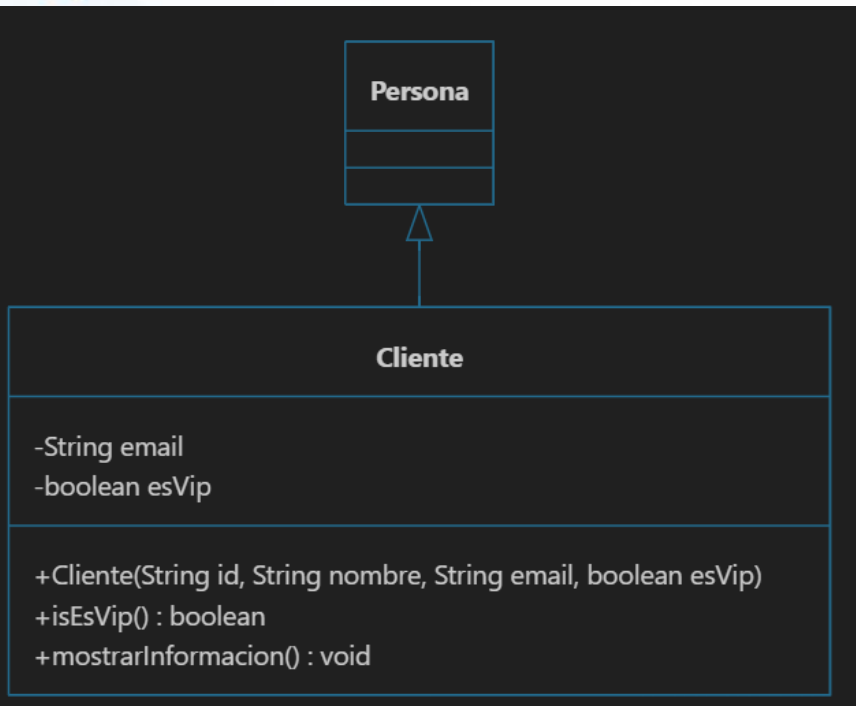
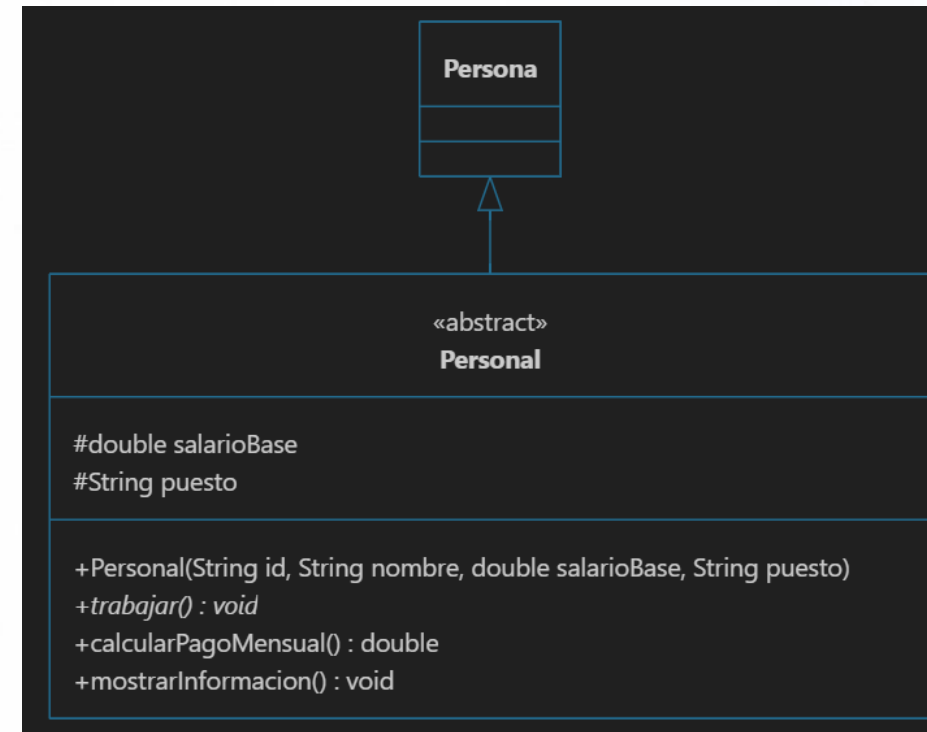
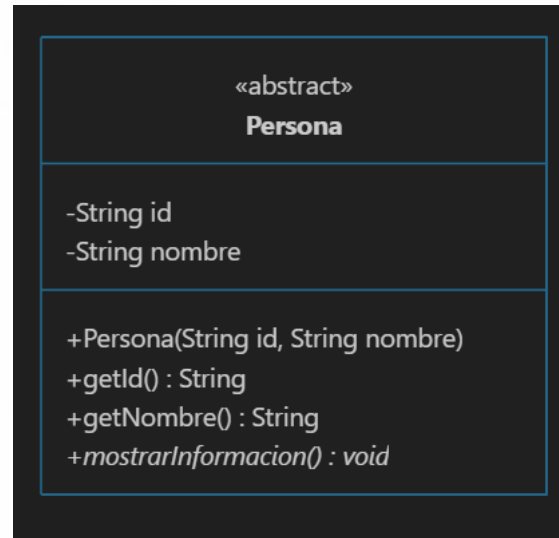


Aplica los principios de la P.O.O para dar significado a cada elemento

- Describe solo los **elementos esenciales** del problema.
- Almacena **la información** en la clase que la utiliza.
- Distribuye **las responsabilidades** en capas.
- Utiliza **especialización** en lugar de **ampliación**.
- Fija los **contratos** antes de implementar su **realización**.



Aplica los principios de la P.O.O para dar significado a cada elemento



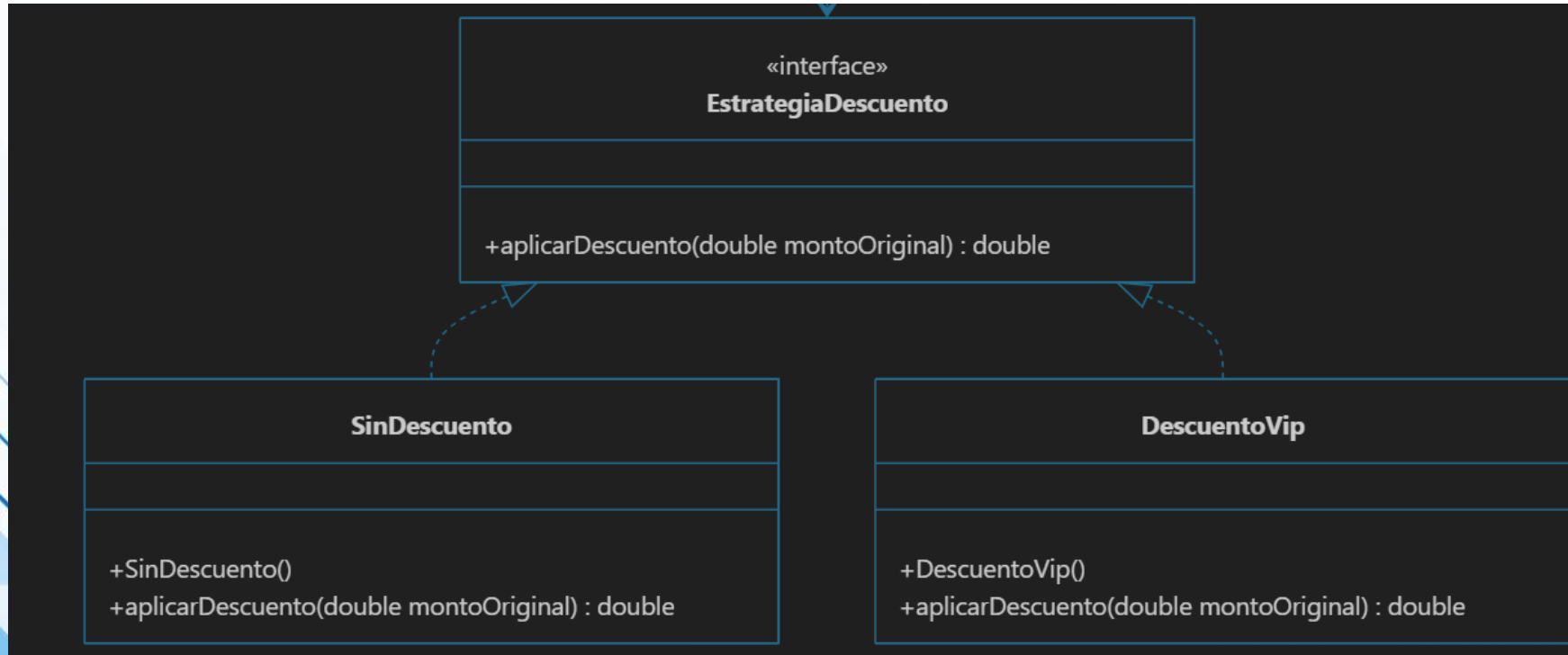
Saber ¿Cuándo se justifica un patrón de diseño?

- Detecta **problemas recurrentes**.
- Identifica **comportamientos sujetos a cambios**.
- Localiza **dependencias** que dificultan modificaciones.
- Evalúa las **alternativas** aplicables.
- Compara **beneficios** vs **complejidad**.
- Identifica el **origen** de la complejidad (organización, datos, comportamiento)

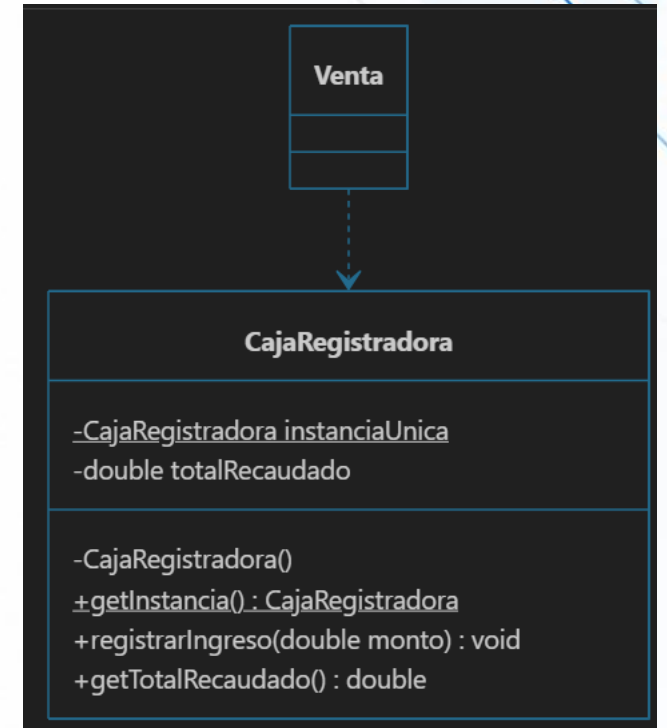


Saber ¿Cuándo se justifica un patrón de diseño?

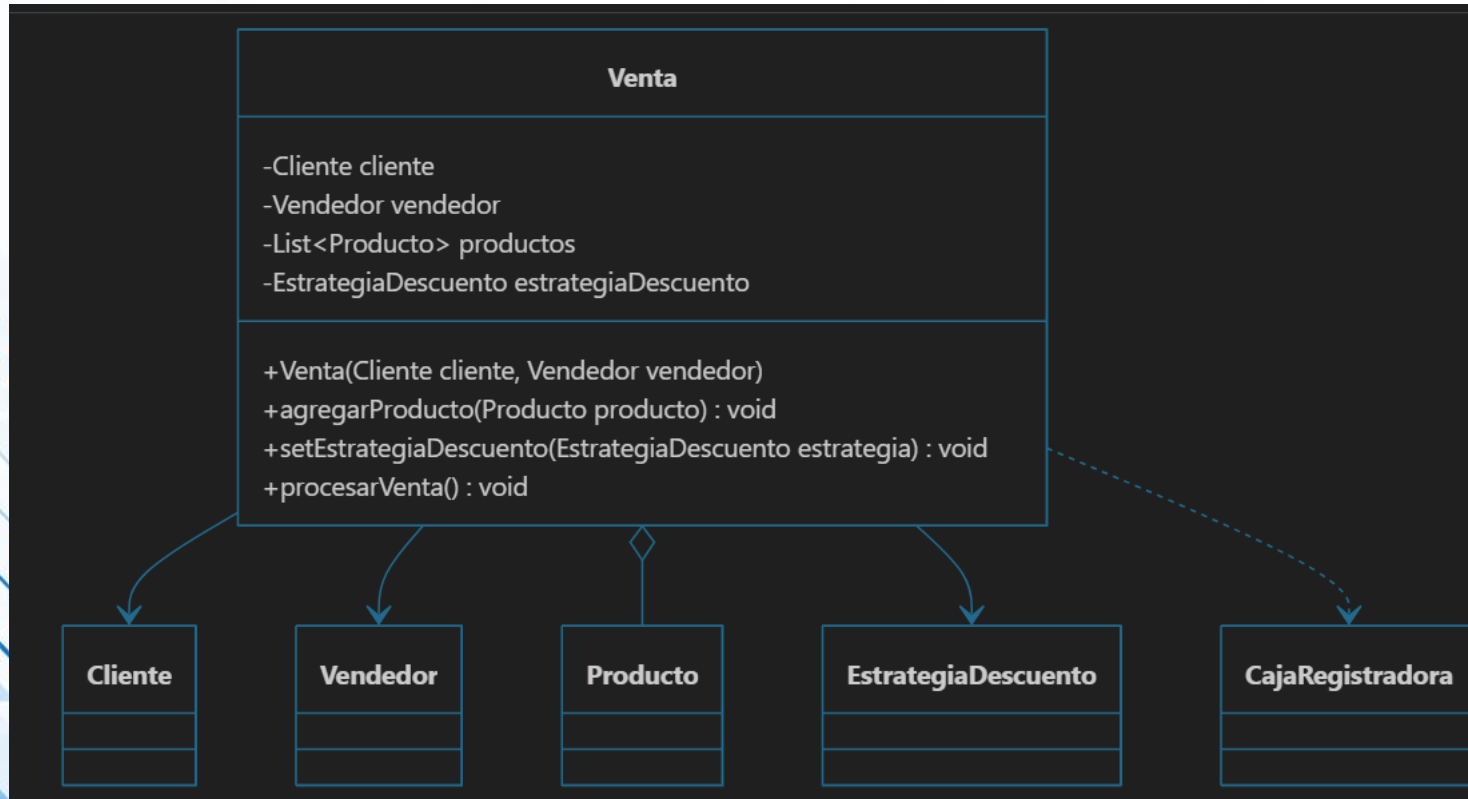
Patron Estrategia



Patron Singleton



Saber ¿Cuándo se justifica un patrón de diseño?



La **Venta** no necesita conocer la fórmula de cada promoción: llama al mismo método y la estrategia que recibe calcula la propia implementación.

La **Venta** se registra en una única caja registradora.

Al Utilizar IA en tus soluciones.

- Explota sus **capacidades** siendo consciente de sus **limitaciones**.
- Especifica tareas **pequeñas y detalladas**, **sin ambigüedades**.
- Establece **metas** realizables **a corto plazo**.
- Ejerce tu **criterio** sobre las **decisiones críticas**.
- Verifica la **alineación** de los resultados con tus **requerimientos**.



Al Utilizar IA en tus soluciones

Archivo de Configuración (AGENTS.md)

```
AGENTS.md M, ●, ⓘ ●
6
7  ## What's Policies you MUST to follow?
8
9  You MUST respect next rules when work:
10
11  - **FOLLOW RECEIVED INSTRUCTIONS**: Align your actions to received instructions contrains &
    boundaries.
12  - **Never execute dangerous commands**: Avoid destructive, regresive or or deep restructurings
    operation without explicit admin permission.
13  - **Stop when not safe**: Stop your work when reached a critical execution exception, work is
    unsafe or destructive, or tools internal failures (non operative). Report stoping causes on
    details.
14  - **Avoid Assumptions working on facts**: Work with provided & existing evidence, reading
    accesible sources & authorized records.
15  - **Avoid Write Transient Files**: Is **STRICTLY PROHIBITED** write transient scripts or garbage
    in workspace root directory. All command invocation or designed test MUST be doing **in memory**.
    Only authorized temporal dir are .tmp/ or .tasks/t{N}/.tmp/ under explicit user authorization.
```

Control de Cambios GIT

```
Changes
Message (Ctrl+Enter to commit on "...
[✓] Commit
> Staged Changes 37
Changes 44
AGENTS.md Demo M, ●, ⓘ
index.html Demo\data U, ●, 1K
productos.csv Demo\data M, ●, ⓘ
JS app.js Demo\data\web U, ●, 2K
# styles.css Demo\data\web U, ●, 5K
JS tienda-api.js Demo\data\web\api U, ●, ⓘ
JS layout-vista.js Demo\data\web\c... U, ●, ⓘ
JS panel-consulta.js Demo\data\we... U, ●, 2K
JS tabla-datos.js Demo\data\web\c... U, ●, ⓘ
JS administrativos.js Demo\data\we... U, ●, ⓘ
JS caja.js Demo\data\web\views U, ●, ⓘ
```

Resultado...

m.

Mi Mercadito

Gestión de la tienda

ESPACIO DE TRABAJO

Productos

Cientes

Vendedores

Administrativos

Caja

Mi Mercadito / Productos

Panel de gestión

MI MERCADITO / GESTIÓN

Productos

El catálogo de tu tienda, organizado en un solo lugar.

Información de la tienda

Actualizar

Información actualizada.

P01

Café Molido Premium

\$10.00

P02

Caja de Galletas de Chocolate

\$5.00

P03

Aceite de Oliva

\$15.00

P04

Arroz Integral 1 kg

\$2.50

P05

Leche Deslactosada 1 litro

\$1.50

Conclusiones



Toda tarea de que depende de la **representación** del mundo real mediante datos **puede ser automatizada**.



Java es una poderosa herramienta, **ampliamente utilizada** para la implementación de soluciones de software en la industria.



La ampliación de **las capacidades de la IA** brinda nuevas oportunidades para solucionar problemas reales. Pero siempre **asumiendo sus limitaciones** con **criterio**.



Lograr **el éxito de una solución** de software requiere **comprender el problema en profundidad** para tomar **buenas decisiones** durante su diseño.

Java en tiempos de IA

Tomando decisiones más allá del código

ING. Yoel David Correa Duke
yoeldcd@uci.cu

Corporación Universitaria
para el Desarrollo de Internet A.C

problema

diseño

criterio

solución