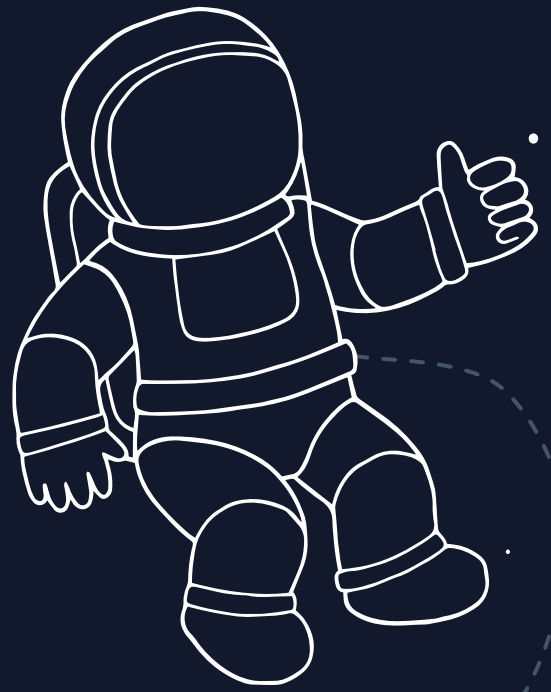
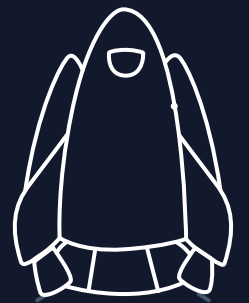


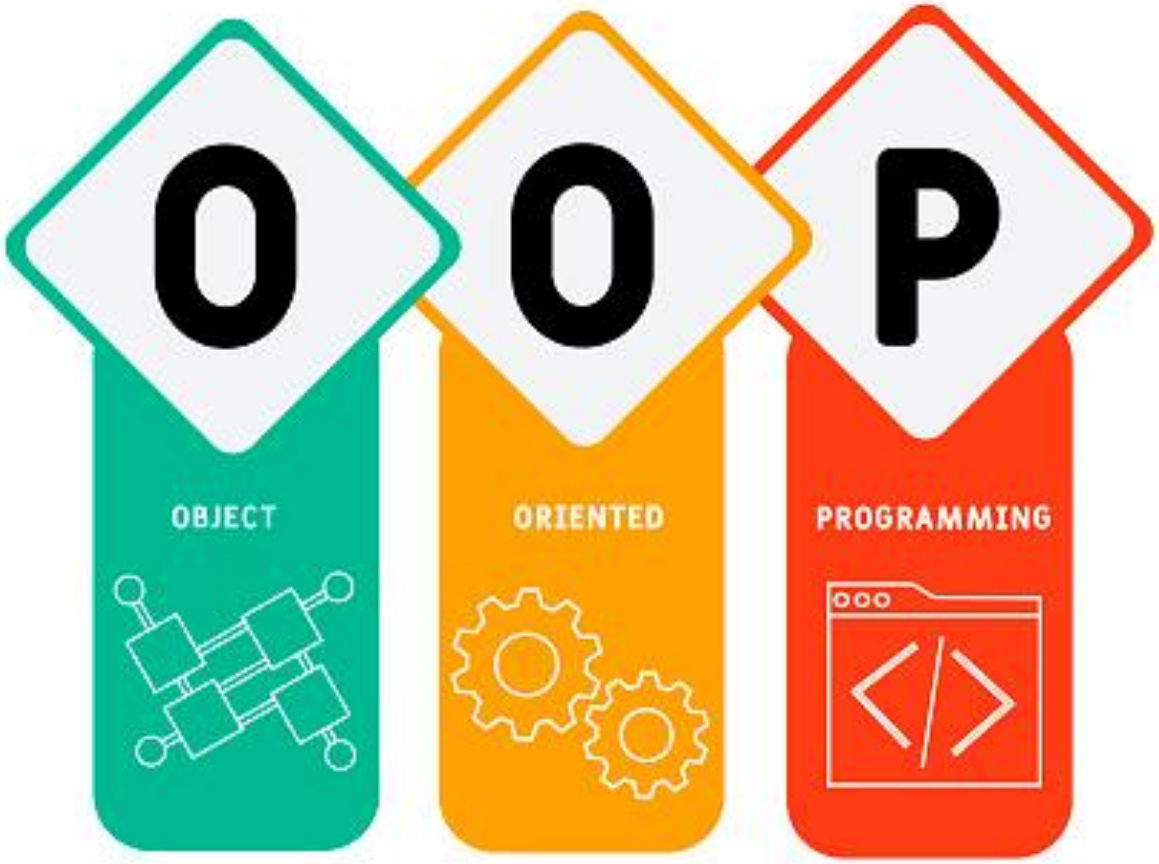
Programa académico CAMPUS

MODULO JAVA
Sesión 5
Introducción a la POO



Trainer Carlos H. Rueda C.

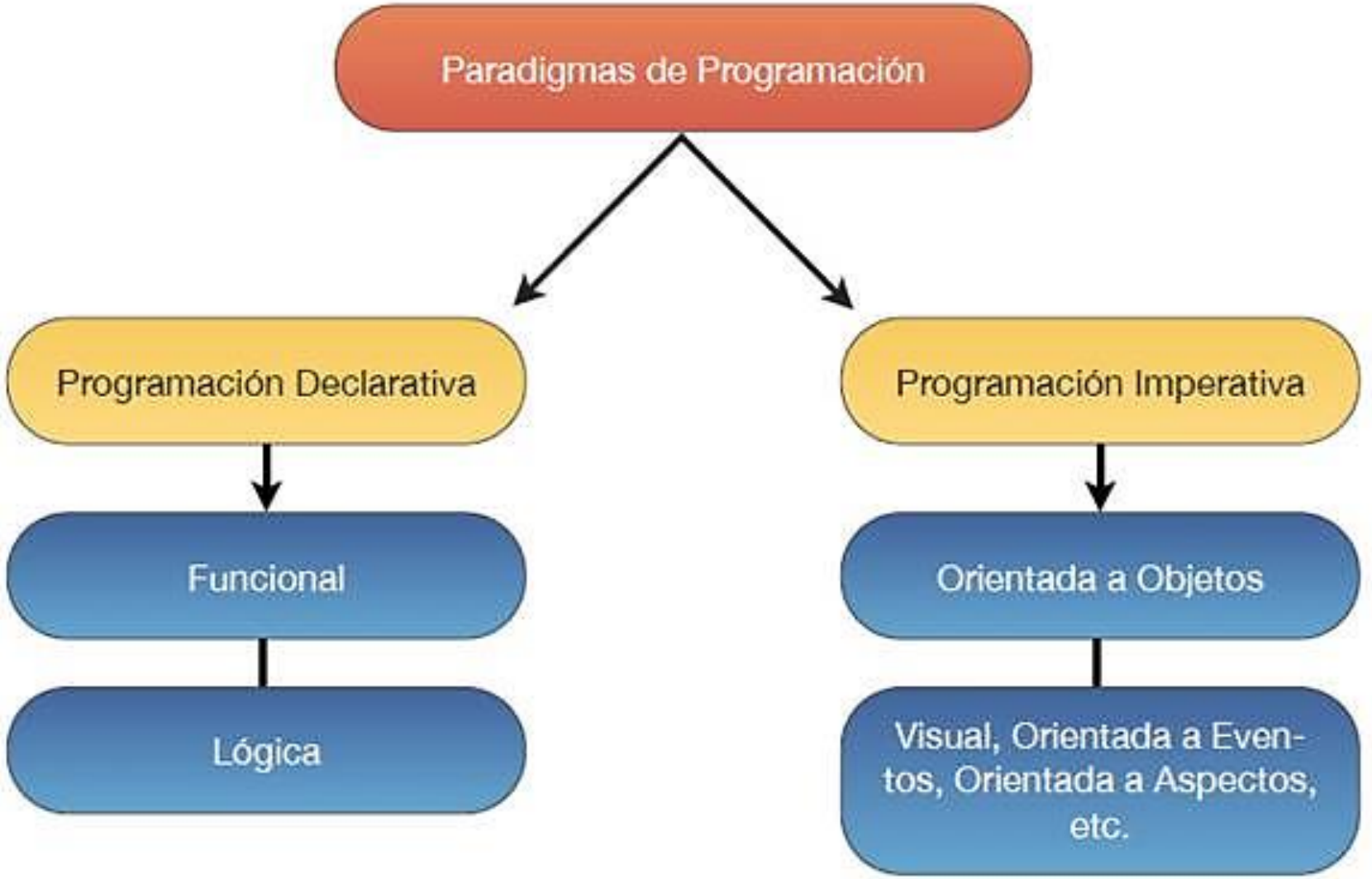

Java™


O
O
P
OBJECT
ORIENTED
PROGRAMMING

¿Qué es la programación orientada a objetos?

La Programación Orientada a Objetos (POO) es un modelo o estilo de programación que se basa en el uso de clases y objetos. Este enfoque permite estructurar un programa de software en componentes simples y reutilizables, representados por clases, de los cuales se crean instancias individuales de objetos.

¿Qué es la programación orientada a objetos?



¿Qué es la programación orientada a objetos?

¿Qué es la programación orientada a objetos?

Con la POO, el objetivo es dejar de lado la lógica pura del programa y comenzar a pensar en términos de objetos, lo cual constituye la base de este paradigma. Esto resulta especialmente útil en sistemas de gran escala, ya que, en lugar de enfocarse en las funciones individuales, se consideran las relaciones e interacciones entre los diferentes componentes del sistema.

¿Qué es la programación orientada a objetos?

En la POO, un programador diseña el programa de software organizando información y comportamientos relacionados en una plantilla llamada clase. Luego, se crean objetos individuales a partir de estas clases. El programa de software se ejecuta cuando varios objetos interactúan entre sí para formar un programa más completo.

¿Por qué la programación orientada a objetos?

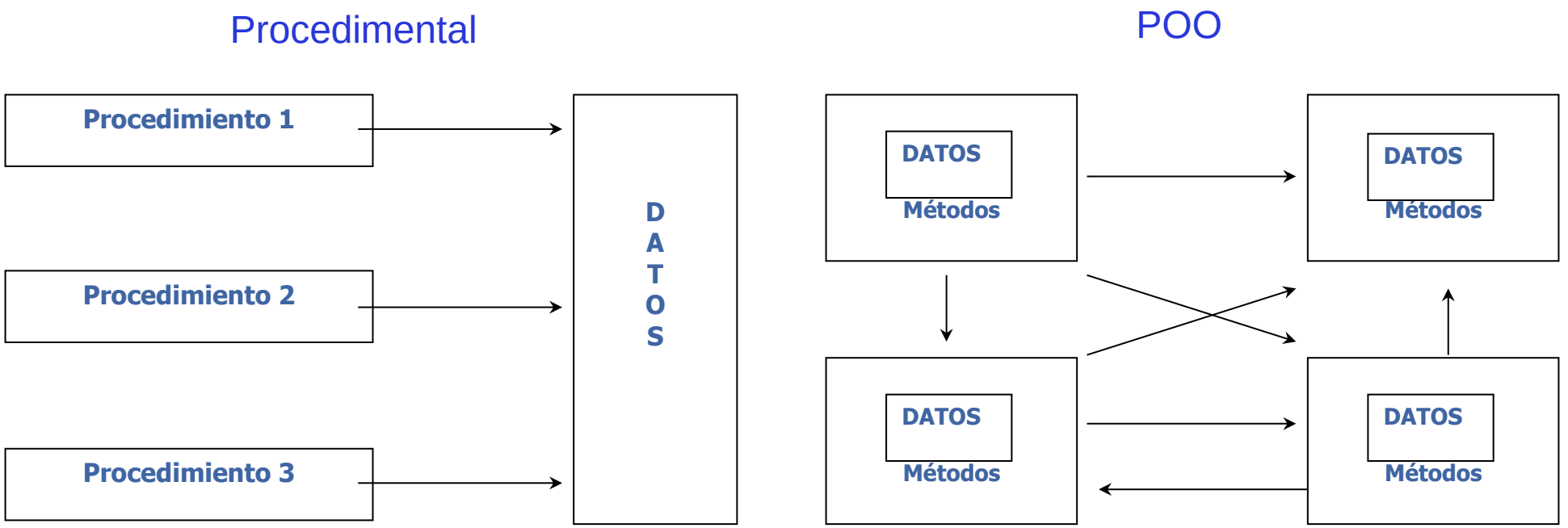
La Programación Orientada a Objetos (POO) toma inspiración de nuestra comprensión del mundo que nos rodea. Si consideramos la creación de un sistema de gestión de bibliotecas como ejemplo, en lugar de enfocarnos en algoritmos y estructuras de datos, la POO nos invita a pensar en entidades presentes en la biblioteca, como libros, bibliotecarios y usuarios. En este enfoque, cada una de estas entidades se convierte en un objeto con sus propias propiedades (datos) y comportamientos (funcionalidades). Por ejemplo, un objeto "Libro" podría tener atributos como título, autor y año de publicación, junto con métodos para obtener información sobre el libro, prestarlo o devolverlo a la biblioteca.

¿Por qué la programación orientada a objetos?

La esencia de la POO reside en la interacción entre estos objetos. Pueden comunicarse enviándose mensajes y colaborando para alcanzar un objetivo común. Por ejemplo, un objeto "Usuario" podría enviar un mensaje al objeto "Libro" para solicitar un préstamo, y el libro respondería actualizando su estado interno. La Programación Orientada a Objetos ofrece beneficios como la reutilización de código, la organización y la facilidad de mantenimiento. Sigue el principio de desarrollo de software conocido como DRY (Don't Repeat Yourself), que busca evitar la duplicación de código y promover la eficiencia en la creación de programas. Además, mediante la encapsulación y la abstracción, se evita el acceso no autorizado a los datos y la exposición del código propietario.

¿Por qué la programación orientada a objetos?

¿Por qué la programación orientada a objetos?



¿Por qué la programación orientada a objetos?

¿Qué es?	• Es un paradigma de programación que usa objetos y las interacciones entre los mismos
¿Por qué?	• Necesidad de organizar el código fuente • Evitar líneas de código innecesarias(Repetidas)
¿Para qué?	• Diseñar programas informáticos y aplicaciones • Proteger los datos de modificaciones incontroladas

Clases y objetos

El proceso de creación de programas orientados a objetos implica, de manera resumida, la creación de clases y la posterior creación de objetos basados en estas clases. Las clases sirven como el modelo a partir del cual se estructuran los datos y comportamientos del programa.

Clases y objetos

General - Genérico

CLASE: Asociación de

Ejemplo: Silla

Atributos (Características o propiedades) => Información que identifica la clase

Silla=> Color, material, número de patas

Métodos (Operaciones – Acciones) => Usos de la clase

Silla=> sentar(), golpear(), alcanzar()

Clases y objetos

Particular - Específico,
Instancia de una Clase

OBJETO: Asociación de

Ejemplo: La silla donde estamos sentados, la silla del comedor, la silla de espera en un Banco

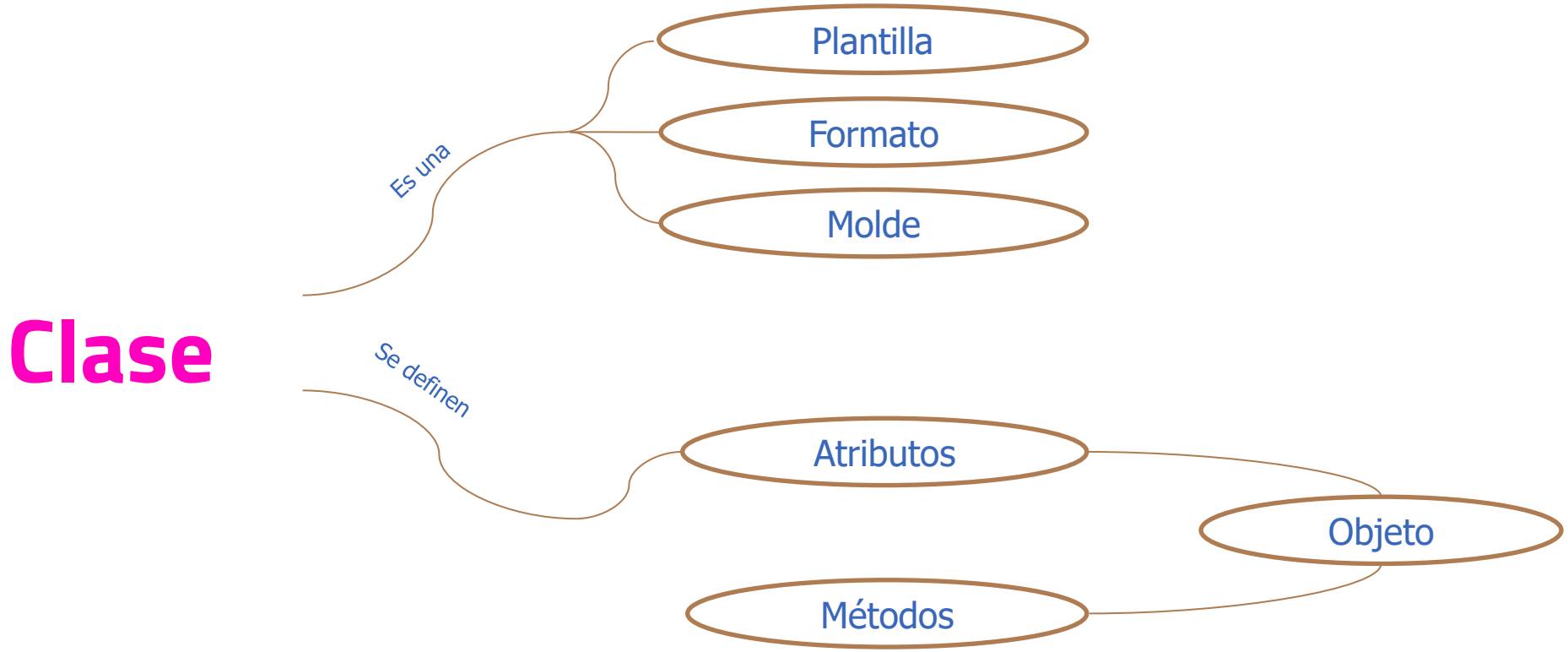
Atributos (Características o propiedades)

Silla=> Color= Negro, material=madera, número de patas=2

Métodos (Operaciones – Acciones)

Silla=> sentar(), golpear(), alcanzar()

Clases y objetos



Clases y objetos

Objeto

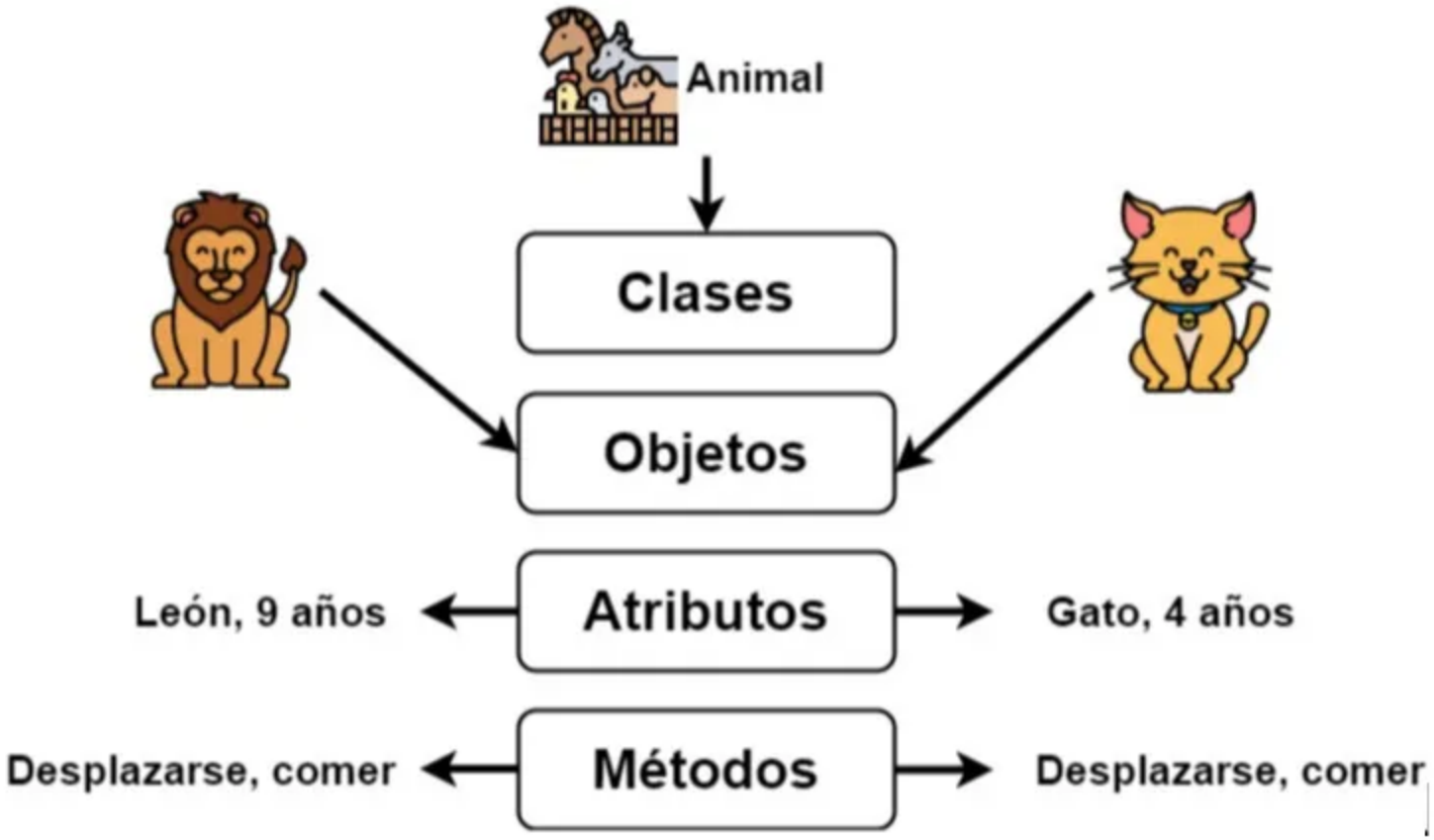
Es una

Entidad

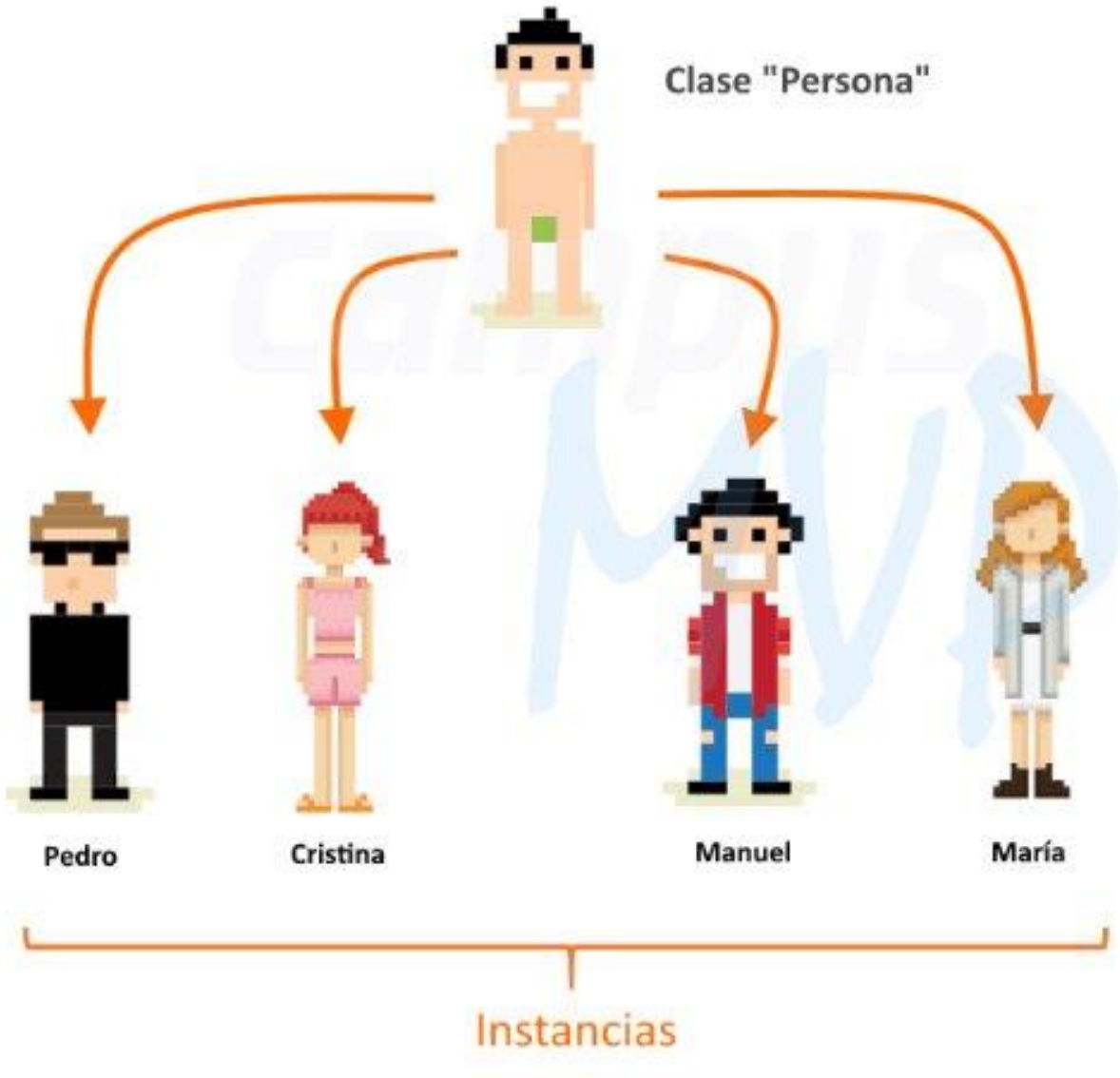
Posee

- **Identidad (Nombre)** — Celular
- **Características (Atributos)** — Tamaño: 20; Color: Negro; Marca: Samsung
- **Comportamiento (Métodos)** — Comunicar

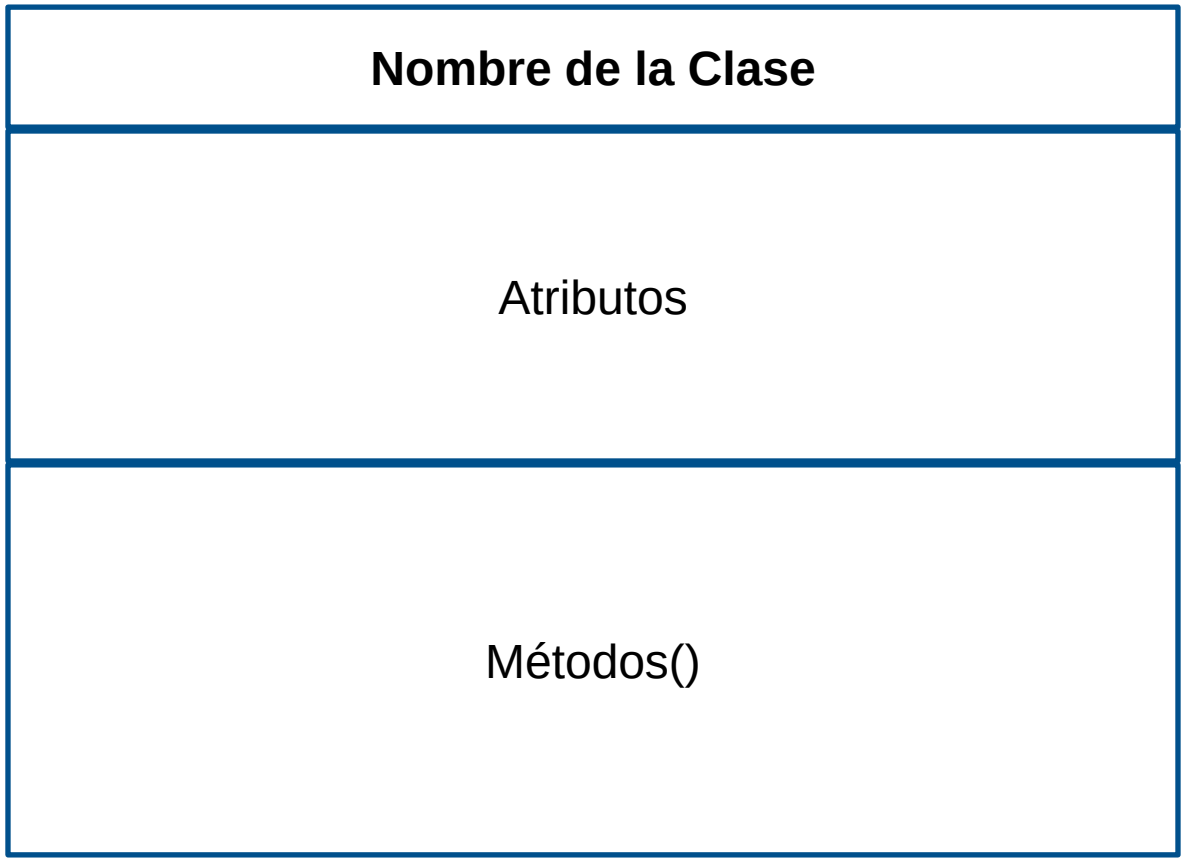
Clases y objetos



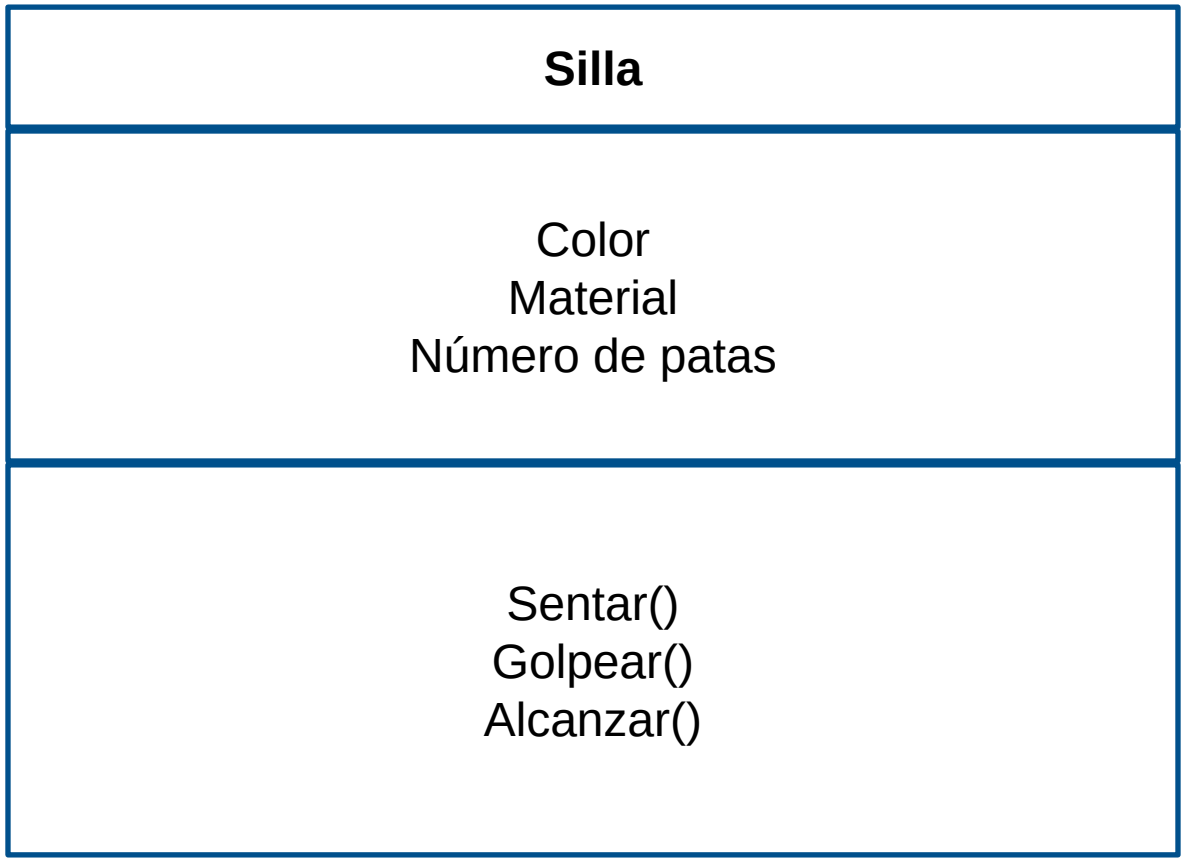
Clases y objetos



Clases y objetos



Clases y objetos



Clases y objetos

Clase: **Cuenta corriente**

- **Atributos:**
  - Número
  - Nombre
  - Saldo
- **Métodos:**
  - Depositar
  - Girar
  - Consultar saldo

Clases y objetos

Clase: **Cuenta corriente**

- Instanciación: **Cuenta corriente A, B**

Clase: **Cuenta corriente**

- **Atributos:**
  - Número
  - Nombre
  - Saldo
- **Métodos:**
  - Depositar
  - Girar
  - Consultar saldo

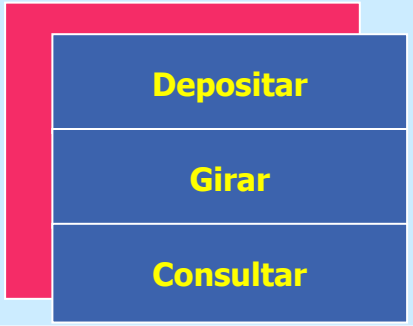
Objeto **A**

Numero 1234

Nombre: Juan

Saldo: 350.000

Métodos



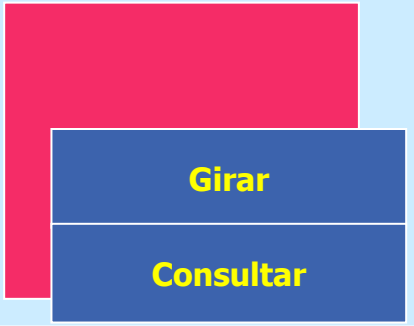
Objeto **B**

Numero: 9876

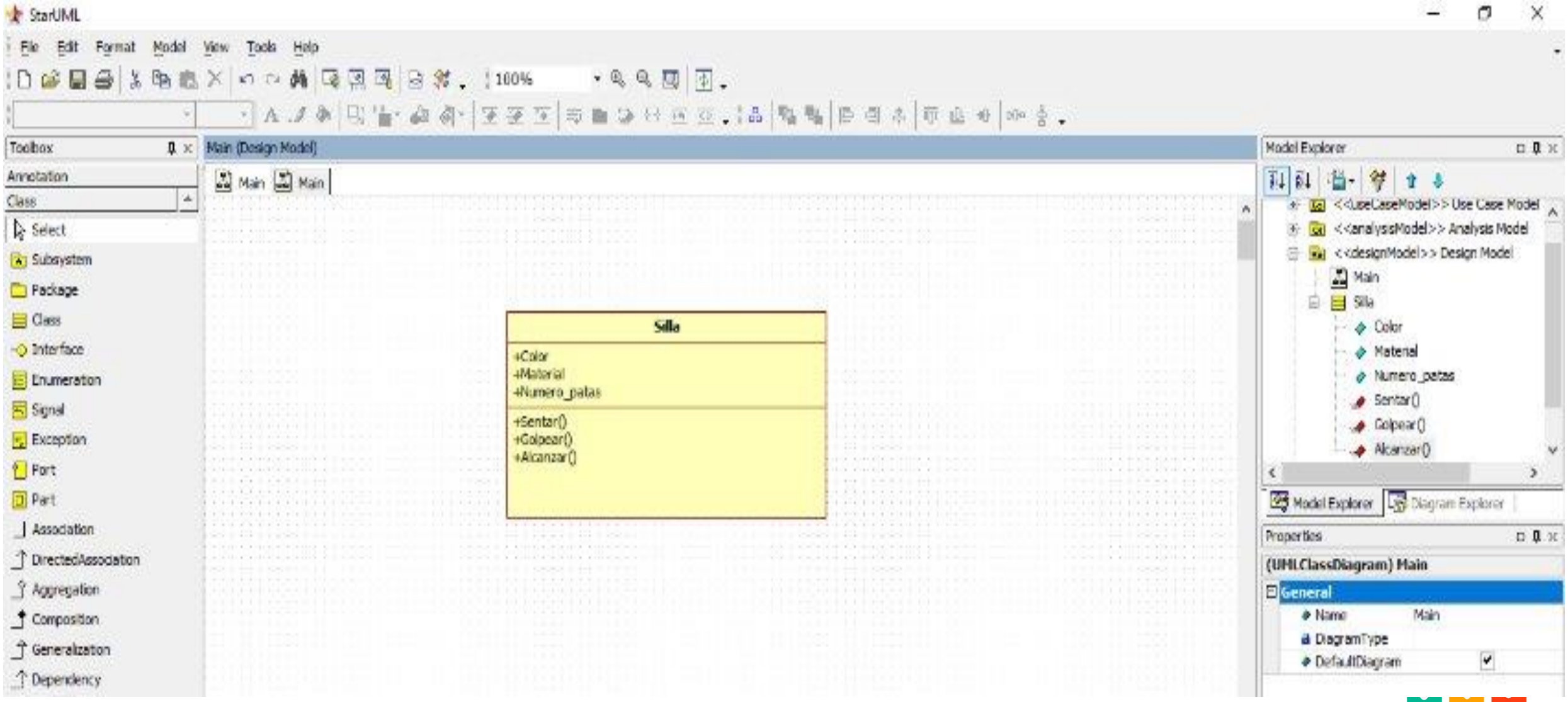
Nombre: María

Saldo: 450.600

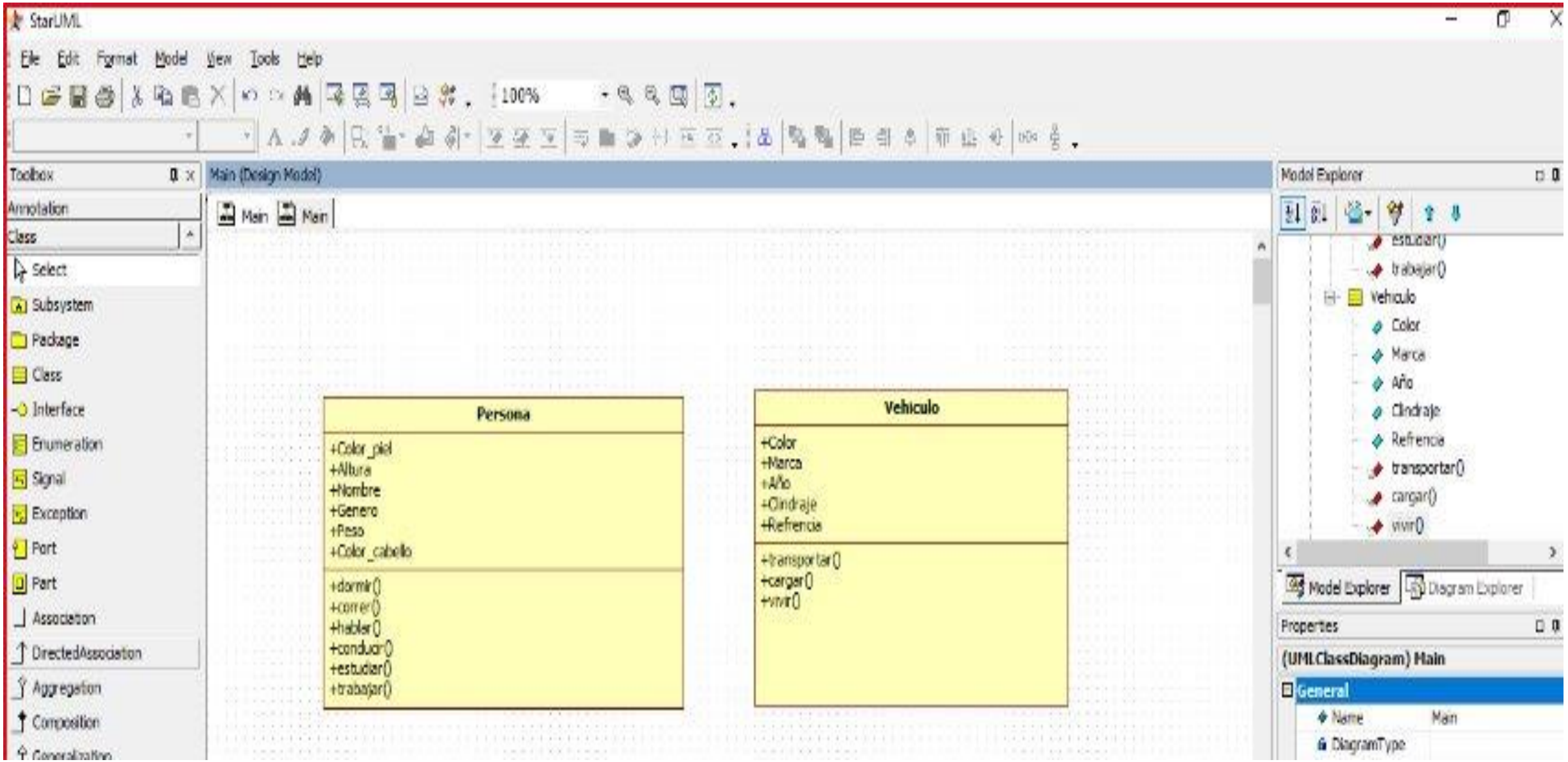
Métodos



Clases y objetos



Clases y objetos



Clases y objetos

Dada la siguiente información sobre un vendedor de una empresa, del cual se conoce:

- Documento de identidad
- Tipo Vendedor(1=Puerta a Puerta, 2=Telemercadeo)
- Valor ventas del mes

Se pide calcular el valor a pagar por concepto de comisión al vendedor, de acuerdo con la siguiente indicación:

Para el vendedor de tipo 1(Puerta a Puerta) se le paga por concepto de comisión el 25% del valor de las ventas del mes. Cuando el vendedor es tipo 2(Telemercadeo) se le paga por concepto de comisión el 20% del valor de las ventas del mes.

Realizar el diseño que resuelva la situación problema presentada, utilizando el concepto de Clases y Objetos (POO).

Clases en Java

```
Class MiClase {
    // Atributos de la clase

    // Constructor de la Clase

    // Método de la clase
}
```

Clases en Java

```java
//Le damos un nombre "MiClase" a la clase
public class MiClase
{
    //Atributos de la clase
    String atributo1;
    int atributo2;
    float atributo3;

    //Constructor con el mismo nombre de la clase
    public MiClase(){}

    //Métodos de la clase
    public void metodo1()
    {
        //Método vacío
    }

    public String metodo2()
    {
        return "metodo2" ;
    }
}
```

Clases en Java

Los métodos de una clase describen la forma en la cual ésta interactúa con su entorno. Pueden ser **públicos, privados o protegidos**.

- ✓**public (+):** indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.

- ✓**private (-):** indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la misma clase lo pueden acceder).

- ✓**protected (#):** indica que el método no será accesible desde fuera de la clase, pero si podrá ser utilizado por métodos de la clase además de las subclases que se deriven (herencia).

Las funciones (métodos) en Java

En Java, las funciones se definen como métodos dentro de una clase. Los métodos permiten encapsular un conjunto de instrucciones que realizan una tarea específica. Un método en Java se declara dentro de una clase utilizando la siguiente sintaxis:

```
modificador tipoDeRetorno nombreDelMetodo ([par+ametros]){

  //cuerpo del método

}
```

Las funciones (métodos) en Java

```
modificador tipoDeRetorno nombreDelMetodo ([par+ametros]){

  //cuerpo del método

}
```

- **Modificadores**: Son opcionales y pueden especificar el acceso y otros atributos del método (por ejemplo, `public`, `private`, `static`, etc.).
- **Tipo de retorno:** Especifica el tipo de dato que el método devuelve como resultado, o se utiliza la palabra reservada `void` si el método no devuelve ningún valor.
- **Nombre del método:** Es el identificador del método que se utiliza para invocarlo posteriormente.
- **Parámetros:** Son valores de entrada que el método puede recibir. Pueden ser de cualquier tipo de dato y se separan por comas si hay más de uno.
- **Cuerpo del método:** El cuerpo del método es el bloque de código que contiene las instrucciones que se ejecutan cuando se invoca el método. Puede contener declaraciones de variables, estructuras de control como bucles y condicionales, y cualquier otra instrucción válida en Java. Dentro del método también se pueden utilizar los parámetros pasados y devolver con `return` el valor del tipo de retorno a devolver.

Invocación de un método

```
tipoDeRetorno resultado = nombreDelMetodo(argumentos);
```

- **Tipo de retorno:** Si el método devuelve un valor, se puede asignar a una variable del tipo correspondiente.
- **Nombre del método:** Es el identificador del método que se desea invocar.
- **Argumentos:** Son los valores que se pasan al método, en el mismo orden que se declararon los parámetros.

Métodos estáticos

Es importante recordar que, en Java, los métodos pueden ser llamados desde otros métodos dentro de la misma clase o desde instancias de la clase si el método es público o accesible. También existen métodos estáticos que se pueden invocar directamente desde la clase sin necesidad de crear una instancia. La estructura de un método estático es:

```
modificador static tipoDeRetorno nombreDelMetodo ([parámetros]){
  //cuerpo del método
}
```

Métodos estáticos

La principal diferencia entre un método estático y un método de instancia es que un método estático se puede invocar directamente desde la clase, sin necesidad de crear una instancia de la clase.

```
public class MiClase {
     [modificadores] static tipoDeRetorno nombreDelMetodo ([parámetros]) {
          // Código del método estático
      }
 }
```

```
MiClase.metodoEstatico();
```

Clases y Objetos Java

Ejemplo de clase Java

```java
public class Animal {
    String tipo;
    int edad;

    public Animal(String nuevoTipo) {
        tipo = nuevoTipo; //Se le da un tipo al animal
    }

    //Método para obtener la edad del animal
    public int getEdad() {
        return edad;
    }

    //Método para establecer la edad del animal
    public void setEdad(int nuevoEdad) {
        edad = nuevoEdad;
    }

    //Método para obtener el tipo del animal
    public String getTipo() {
        return tipo;
    }
}
```

Objetos en Java

Al crear objetos en Java, hay dos aspectos fundamentales que se deben tener claros. Primero, conocer el nombre de la clase para la cual se quiere crear el objeto, y segundo, entender el constructor asociado a dicha clase, es decir, si el constructor requiere parámetros o no. Para crear objetos en Java, el lenguaje proporciona la palabra clave "`new`". Con esta palabra clave, se indica a Java que se quiere crear un nuevo objeto de una clase específica y se le proporcionan los parámetros necesarios según el constructor correspondiente.

```
MiClase miObjeto; //Declaración de una variable del tipo de la clase
miObjeto = new MiClase(); //Creación del objeto y asignación a la variable
```

```
MiClase miObjeto = new MiClase();
```

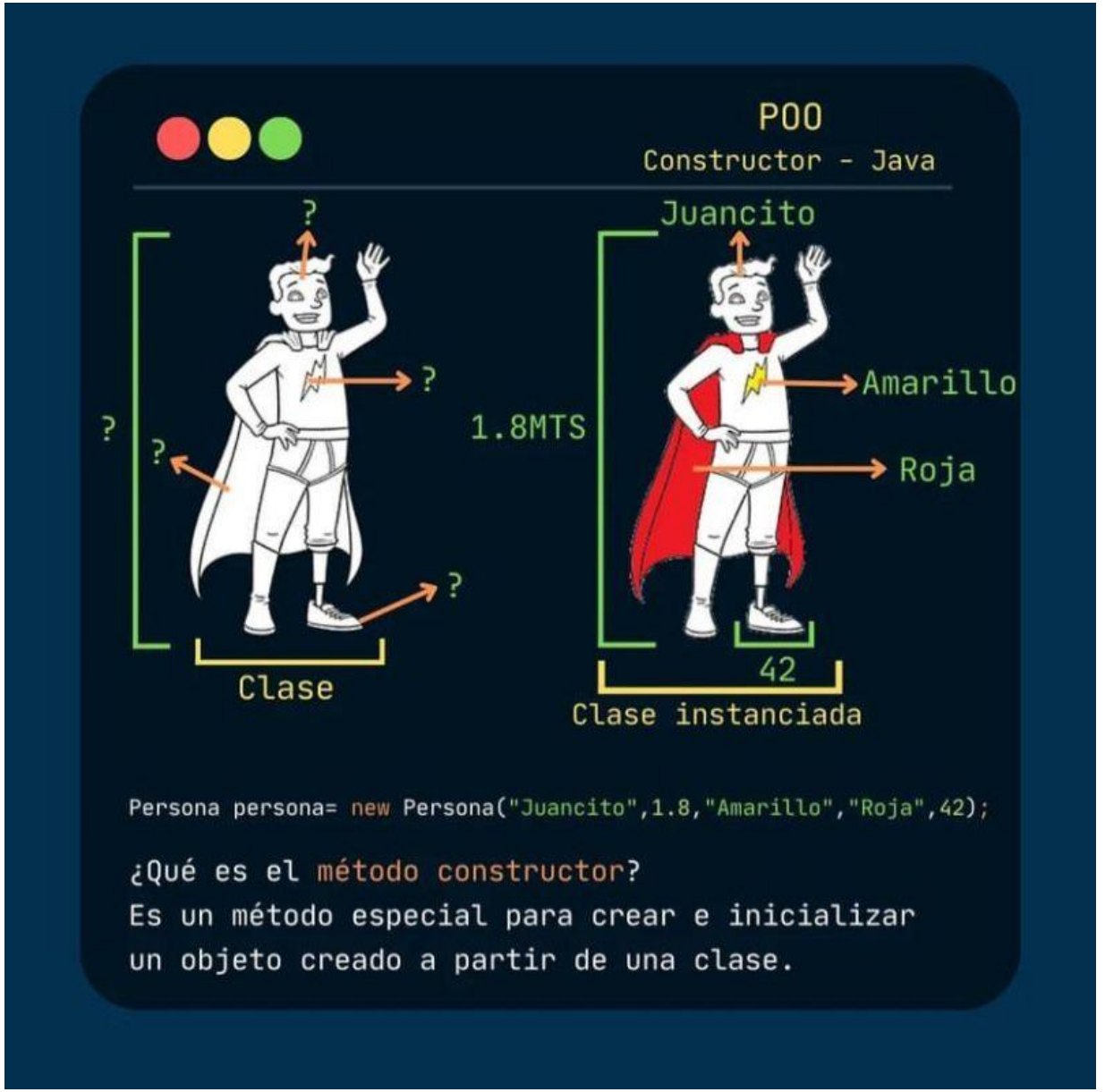
Objetos en Java

```java
//Creación de un animal cuyo tipo será gato
Animal miAnimal = new Animal("gato");
//Se establecen 3 años de edad al gato.
miAnimal.setEdad(3);
//Se muestra el tipo del animal por pantalla
System.out.print("El tipo es: " + miAnimal.getTipo());
//Se muestra la edad del animal por pantalla
System.out.println(" y tiene " + miAnimal.getEdad() + " años");
//Este código debería imprimir "El tipo es: gato y tiene 3 años"
```

Método constructor

En la programación orientada a objetos, incluyendo Java, un constructor es una función especial que se emplea para iniciar un objeto recién creado y asignar valores iniciales a sus variables de instancia. Su propósito principal es establecer el estado inicial del objeto antes de que se utilice en el programa. En resumen, un constructor es un método que se invoca automáticamente al crear un objeto de una clase determinada.

Método constructor



Constructor predeterminado

Java automáticamente proporciona un constructor predeterminado si no se define ningún constructor explícito en la clase. Este constructor predeterminado no recibe parámetros y su cuerpo está vacío. El objetivo principal del constructor predeterminado es inicializar los atributos de la clase con sus valores por defecto. Si se desea proporcionar una implementación personalizada para el constructor predeterminado, debe definir explícitamente un constructor en la clase y proporcionar el cuerpo del mismo.

```
public class Ejemplo {
      int num;
      String cadena;
      public Ejemplo() {
            num = 0;
            cadena = "mi cadena";
      }
}
```

Constructor parametrizado

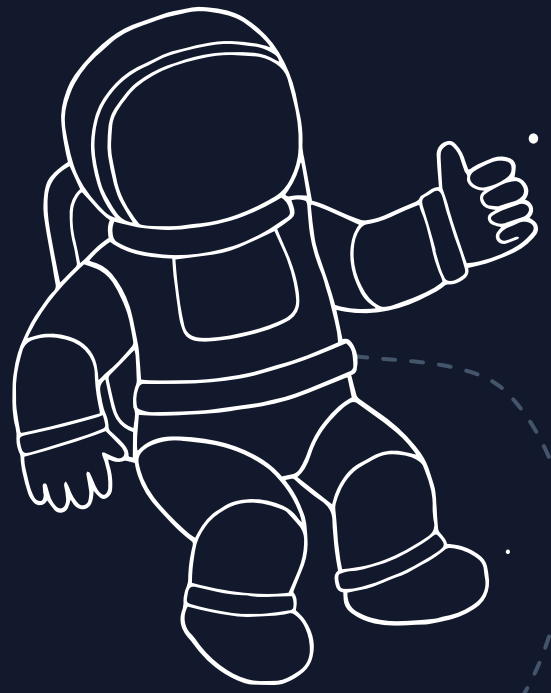
El constructor parametrizado es aquel que recibe uno o más parámetros y se utiliza para inicializar los atributos de la clase con valores específicos proporcionados por el usuario. Es necesario definir explícitamente este tipo de constructor en la clase.

```java
public class Ejemplo {
	int num;
	String cadena;
	public Ejemplo() {
		num = 0;
		cadena = "mi cadena";
	}
}
```

Clases en Java

Enunciado: construya una clase denominada estudiante, con los siguientes atributos nombre, nota1, nota2, nota3, y definitiva. Solicite al usuario los valores para un estudiante y almacénelos en una instancia de la clase estudiante. Calcule la nota definitiva como el promedio simple de las tres notas (nota1+nota2+nota3)/3.

- ✓Diseñar diagrama de clase UML
- ✓Construir clase en Java



Programa académico CAMPUS

Módulo JAVA