



El futuro digital
es de todos

MinTIC

Ciclo: Programación Básica

**Programación Orientada a Objetos
(POO)**

»

Misión TIC 2022

xxx



**Misión
TIC 2022**



El futuro digital
es de todos

MinTIC



Programación Básica



- Clases y objetos - Encapsulamiento
- Constructores - Visibilidad
- Modificadores de clases



El futuro digital
es de todos

MinTIC



x
x
x

Clases y Objetos - Encapsulamiento



 Definición Clase y Objeto (Atributos – Métodos)

 Encapsulamiento



Programación Orientada a Objetos



Definición de Clase – Objeto (Atributos – Métodos)

x
x
x

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()



Programación Orientada a Objetos



Definición de Clase – Objeto (Atributos – Métodos)

x
x
x

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()



Programación Orientada a Objetos



Representación de la Clase



Nombre de la Clase
Atributos
Métodos()



Programación Orientada a Objetos



Representación de la Clase (Ejemplo de la Silla)

x
x
x

Silla
Color Material Número de patas
Sentar() Golpear() Alcanzar()



El futuro digital
es de todos

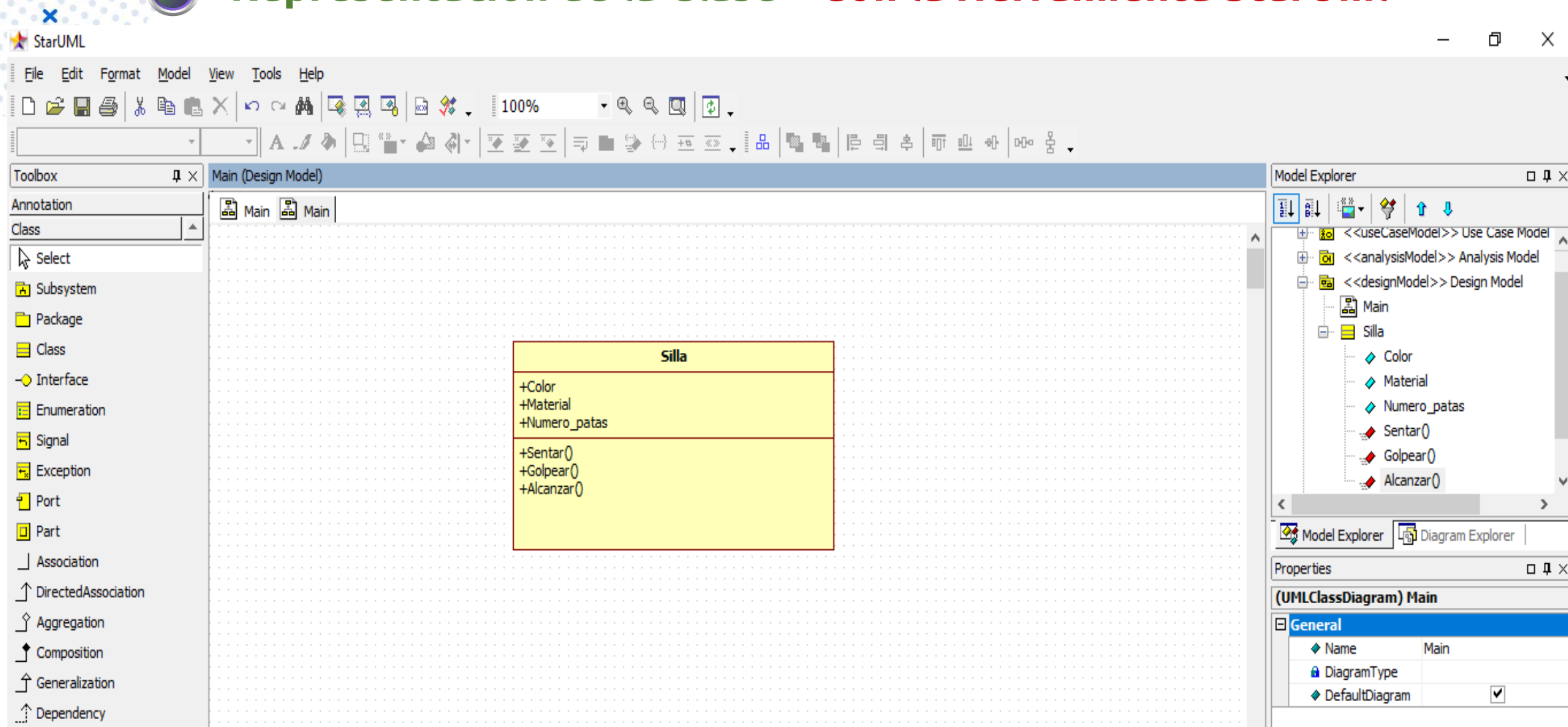
MinTIC



Programación Orientada a Objetos



Representación de la Clase – Con la Herramienta StarUml





El futuro digital
es de todos

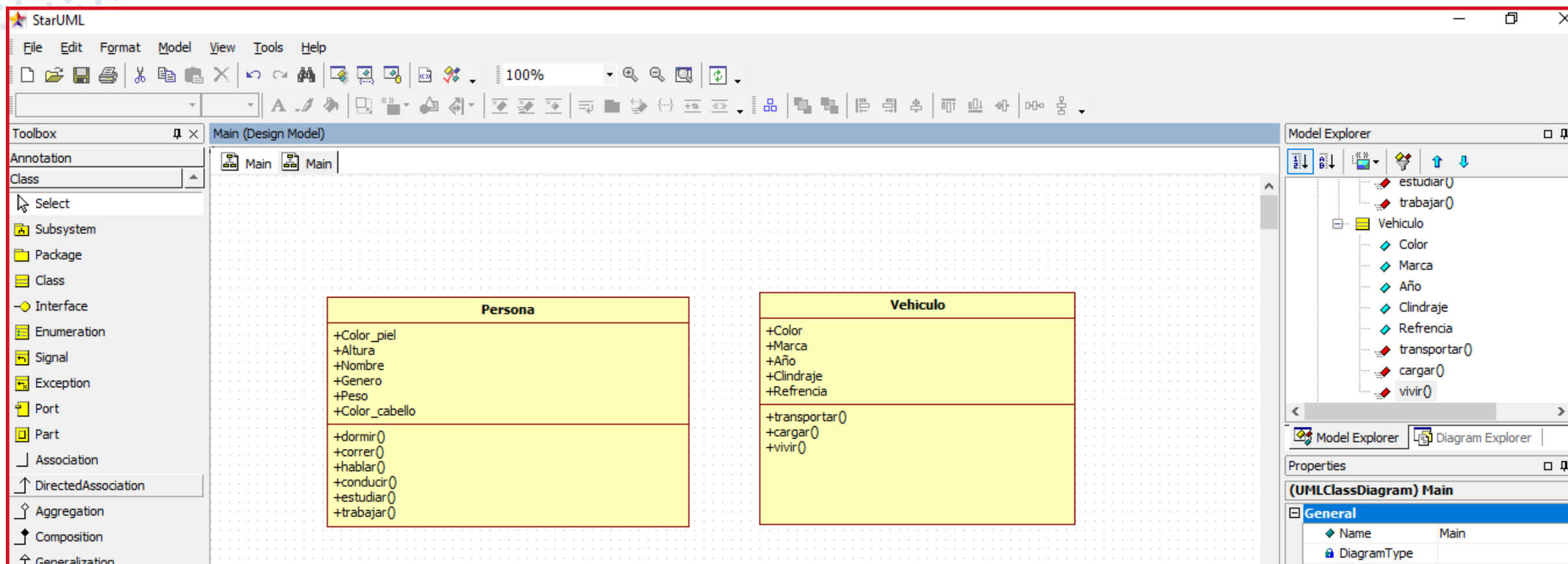
MinTIC



Programación Orientada a Objetos



Representación de la Clase – Con la Herramienta StarUml Otros ejemplos





Programación Orientada a Objetos

Ejemplo: POO

Ejercicio



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 programa en Java que resuelva la situación problema presentada, utilizando el concepto de Clases y Objetos (POO).



Constructor de una Clase



Un **constructor** es un elemento de una **clase** cuyo identificador coincide con el de la **clase** correspondiente y que tiene por objetivo obligar a y controlar cómo se **inicializa una instancia** de una determinada **clase**, ya que el lenguaje **Java** no permite que las variables miembro de una nueva instancia queden sin inicializar.

Un **constructor** es un método perteneciente a la clase que posee unas **características** especiales: Se llama igual que la clase. No devuelve nada, ni siquiera void. Pueden existir varios, pero siguiendo las reglas de la sobrecarga de funciones.



El futuro digital
es de todos

MinTIC

Programación Orientada a Objetos

Ejercicios



Identificación de la Clase

Nombre Clase
Atributos
Métodos



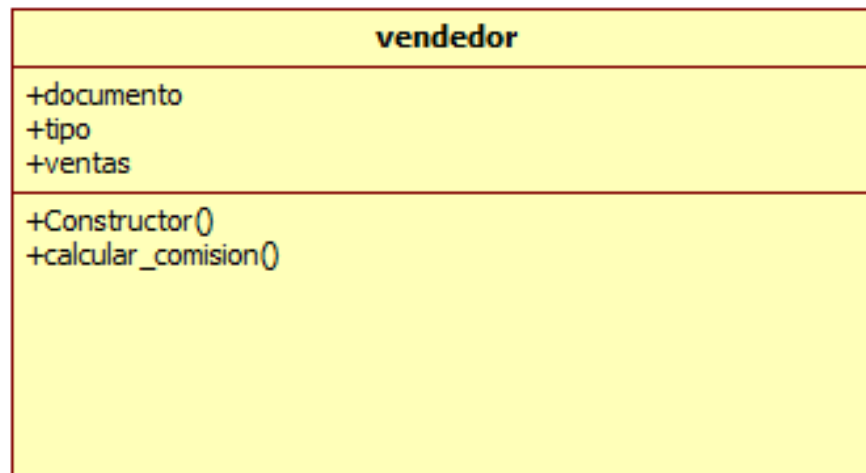
El futuro digital
es de todos

MinTIC

Programación Orientada a Objetos



Clase en StarUML



Ejercicios





Programación Orientada a Objetos

Ejercicios



Método

Parámetros de entrada

Tipo, ventas



Calcular_comisión():
Condicional (tipo de vendedor)



Parámetros de salida

comision

FUNCIÓN retorna o regresa un solo valor



```
public class vendedor {  
    //Atributos  
    long documento;  
    int tipo;  
    double ventas;  
    //Métodos  
    public vendedor(long doc,int tipo,double ven){  
        this.documento=doc;  
        this.tipo=tipo;  
        this.ventas=ven;  
    }  
    public double calcular_comision(){  
        double comision;  
        if (this.tipo==1){  
            comision=this.ventas*0.25;  
        }  
        else{  
            comision=this.ventas*0.20;  
        }  
        return comision;  
    }  
}
```



Ejercicio: (Main)



```
package comisiones_poo_g21;
```

```
/**
```

```
 *
```

```
 * @author SERGIO
```

```
 */
```

```
import java.util.Scanner;
```

```
public class Comisiones_poo_G21 {
```

```
/**
```

```
 * @param args the command line arguments
```

```
 */
```

```
public static void main(String[] args) {
```

```
    // Definición de la consola
```

```
    Scanner consola=new Scanner(System.in);
```

```
    //Definición de variables
```

```
    long documento;
```

```
    int tipo;
```

```
    double ventas,comision;
```

```
    //Definición de la variable para manejar el objeto
```

```
    vendedor objeto_vendedor;
```

```
    //Entrada de Datos
```

```
    System.out.println("Documento: ");
```

```
    documento=consola.nextLong();
```

```
    System.out.println("Tipo(1=Puerta a puerta, 2=Telemercadeo): ");
```

```
    tipo=consola.nextInt();
```

```
    System.out.println("Ventas del mes: ");
```

```
    ventas=consola.nextDouble();
```

```
    //Creación del objeto
```

```
    objeto_vendedor=new vendedor(documento,tipo,ventas);
```

```
    comision=objeto_vendedor.calcular_comision();
```

```
    System.out.println("Comisión: "+comision);
```

```
}
```

```
}
```



El futuro digital
es de todos

MinTIC

Programación Orientada a Objetos

Ejemplo: POO

Ejercicio



Dada un número **entero**, Se pide:

- Conocer si es par o impar
- Conocer si es positivo, negativo o cero
- Conocer si es primo

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



El futuro digital
es de todos

MinTIC

Programación Orientada a Objetos



x
x
x

Clase en StarUML

numero_entero

+numero

+Constructor()

+par_impar()

+positivo_negativo()

+primo()

Ejercicios





Programación Orientada a Objetos

Ejercicios



Método 1

Parámetros de entrada

numero



par_impar()

Condicional (if) (módulo o residuo)



Parámetros de salida

mensaje= Si es par o es
impar

FUNCIÓN retorna o regresa un solo valor



Programación Orientada a Objetos

Ejercicios



Método 2

Parámetros de entrada

numero



Positivo_negativo()

Condicional anidado (Si es >0 o <0
o $=0$)



Parámetros de salida

Mensaje: Es positivo,
negativo o cero

FUNCIÓN retorna o regresa un solo valor



Ejercicio: (Clase)



```
public class numero_entero {  
    //Atributo  
    int numero;  
    //Métodos  
    public numero_entero(int numero){  
        this.numero=numero;  
    }  
    public String par_impar(){  
        String mensaje;  
        if(this.numero%2==0){  
            mensaje="El número "+this.numero+" es PAR";  
        }  
        else{  
            mensaje="El número "+this.numero+" es IMPAR";  
        }  
        return mensaje;  
    }  
}
```

```
public String positivo_negativo(){  
    String mensaje;  
    if (this.numero>0){  
        mensaje="El número "+this.numero+" es POSITIVO";  
    }  
    else if(this.numero<0){  
        mensaje="El número "+this.numero+" es NEGATIVO";  
    }  
    else{  
        mensaje="El número "+this.numero+" es CERO";  
    }  
    return mensaje;  
}
```



Ejercicio: (Main)

```
package positivo_negativo_g21;
```

```
/**  
 *  
 * @author SERGIO  
 */
```

```
import java.util.Scanner;
```

```
public class Positivo_negativo_G21 {
```

```
/**  
 * @param args the command line arguments  
 */
```

```
public static void main(String[] args) {  
    // Definición de la consola  
    Scanner consola=new Scanner(System.in);  
    //Definición variables  
    int numero;  
    String mensaje;  
    //Definición de la variable para el objeto  
    numero_entero objeto_numero;
```

```
    //entrada de datos  
    System.out.println("Número: ");  
    numero=consola.nextInt();  
    // Creación del objeto  
    objeto_numero=new numero_entero(numero);  
    mensaje=objeto_numero.par_impar();  
    System.out.println(mensaje);  
    mensaje=objeto_numero.positivo_negativo();  
    System.out.println(mensaje);
```

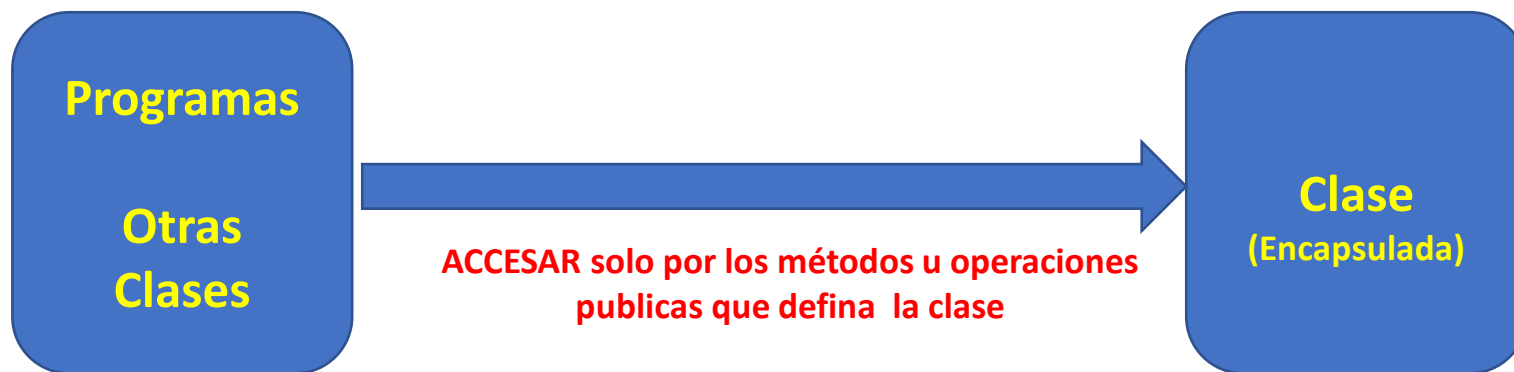
```
}
```

```
}
```



Encapsulamiento

En **programación orientada a objetos**, se denomina **encapsulamiento** al ocultamiento del estado, es decir, de los datos miembro, de un **objeto** de manera que sólo se puede cambiar mediante las operaciones definidas para ese **objeto**.





Métodos Get y Set (Modificadores de las Clases)

En **programación orientada a objetos**, para un correcto **encapsulamiento** de las clases-objetos, se recomienda:

- Atributos definirlos como Privados (Solo para la clase)
- Métodos definirlos como Público (Se puedan utilizar en otros programas u otras clases)



Métodos Get y Set (Modificadores de las Clases)

Los atributos cuando son privados, se deben tener en cuenta crear métodos constructores para cada atributo:

Se debe crear métodos `getAtributo` y `setAtributo` para cada atributo. El método `setAtributo` para asignar al `this.atributo` el parámetro que recibe el constructor y el método `getAtributo` para retornar el atributo y poder obtener su valor.



Programación Orientada a Objetos

Ejemplo: Constructores GET, SET POO

Ejercicio



Utilizando el ejercicio de la liquidación de comisiones a un vendedor.

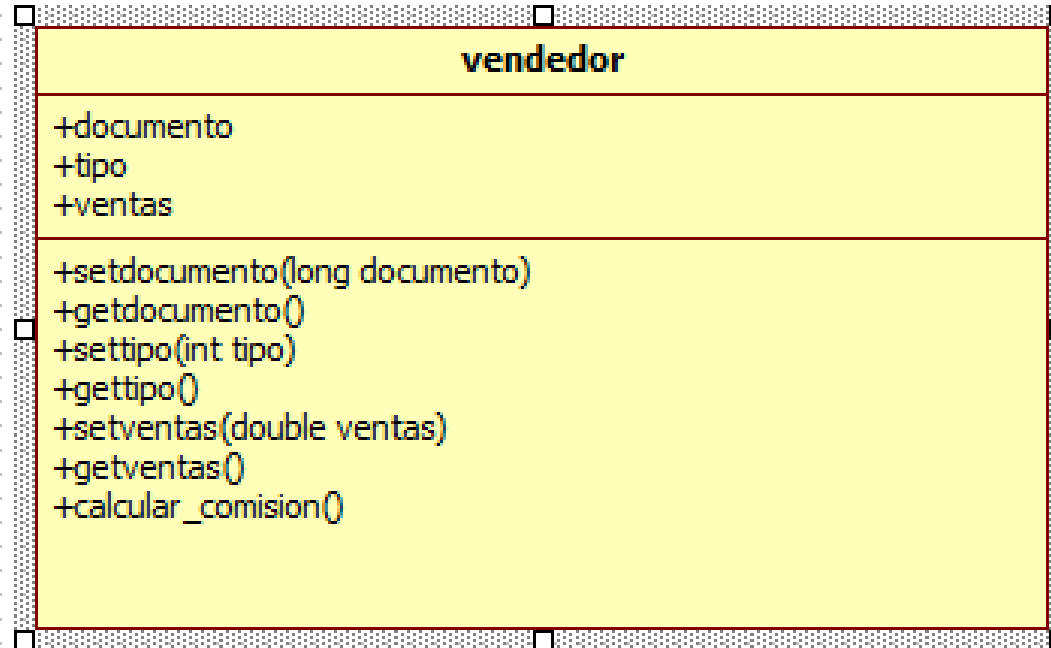
Dada la siguiente información sobre un **vendedor** de una empresa, del cual se conoce:

- 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 programa en Java que resuelva la situación problema presentada, utilizando el concepto de Clases y Objetos (POO).





Ejercicio: (Clase)

```
public class Vendedor {  
    //Atributos privados -> Encapsulamiento  
    private long documento;  
    private int tipo;  
    private double ventas;  
    // Métodos  
    // Constructores  
  
    public Vendedor() {  
    }  
  
    public Vendedor(long documento, int tipo, double ventas) {  
        this.documento = documento;  
        this.tipo = tipo;  
        this.ventas = ventas;  
    }  
}
```

```
//Set- Get  
public void setDocumento(long documento) {  
    this.documento = documento;  
}  
public void setTipo(int tipo) {  
    this.tipo = tipo;  
}  
public void setVentas(double ventas) {  
    this.ventas = ventas;  
}  
public long getDocumento() {  
    return documento;  
}  
public int getTipo() {  
    return tipo;  
}  
public double getVentas() {  
    return ventas;  
}  
public double calcular_comision() {  
    double comision;  
    if(this.tipo==1){  
        comision=this.ventas*0.25;  
    }  
    else{  
        comision=this.ventas*0.20;  
    }  
    return comision;  
}
```




```
public static void main(String[] args) {  
    // Definición de la consola  
    Scanner consola=new Scanner(System.in);  
    //Definición de variables  
    int tipo,N;  
    long documento;  
    double ventas,comision,vtc;  
    //Definición de la variable Objeto  
    Vendedor obj_vendedor;  
    System.out.println("Cantidad de vendedores: ");  
    N=consola.nextInt();  
    vtc=0;  
    for(int i=1;i<=N;i++){  
        System.out.println("Documento: ");  
        documento=consola.nextLong();  
        System.out.println("Tipo(1=Puerta a puerta, 2=Telemercadeo): ");  
        tipo=consola.nextInt();  
        System.out.println("Ventas: ");  
        ventas=consola.nextDouble();  
        //Creación del objeto vendedor  
        obj_vendedor=new Vendedor(documento,tipo,ventas);  
        comision=obj_vendedor.calcular_comision();  
        vtc=vtc+comision;  
        System.out.println("Documento: "+obj_vendedor.getDcoumento());  
        System.out.println("Comision: "+comision);  
    }  
    System.out.println("Valor total Comisiones: "+vtc);  
}
```



Sobrecarga de Constructores

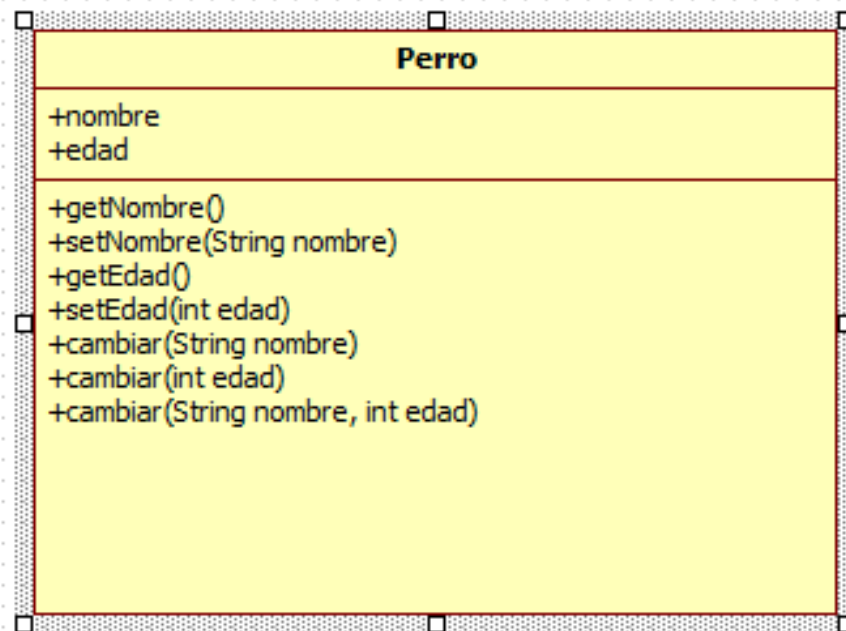
En una clase, la **sobrecarga** (*overloading*) permite definir más de un constructor o método con el mismo nombre, con la condición de que no puede haber dos de ellos con el mismo número y tipo de parámetros.



Programación Orientada a Objetos



Clase



Se definen tres (3) métodos cambiar con diferentes parámetros



Ejercicio: (Clase)



```
public class perro {  
    private String nombre;  
    private int edad;  
  
    public String getNombre()  
    {  
        return nombre;  
    }  
    public void setNombre(String nombre)  
    {  
        this.nombre=nombre;  
    }  
    public int getEdad()  
    {  
        return edad;  
    }  
    public void setEdad(int edad)  
    {  
        this.edad=edad;  
    }  
    public void cambiar(String nombre)  
    {  
        this.nombre = nombre;  
    }  
    public void cambiar(int edad)  
    {  
        this.edad = edad;  
    }  
}
```

```
public void cambiar(String nombre, int edad )  
{  
    this.nombre = nombre;  
    this.edad = edad;  
}
```



Ejercicio: (Main)

```
import java.util.Scanner;
```

```
public class Sobrecarga_ejercicio {
```

```
/**
```

```
 * @param args the command line arguments
```

```
 */
```

```
public static void main(String[] args) {
```

```
    // Definición consola
```

```
    Scanner consola=new Scanner(System.in);
```

```
    //perro perro1 = new perro("Chispas", 5)
```

```
    perro perro1;
```

```
    perro perro2;
```

```
    perro perro3;
```

```
    perro1=new perro();
```

```
    perro1.setNombre("Chispas");
```

```
    perro1.setEdad(5);
```

```
    perro2 = new perro();
```

```
    perro2.setNombre("Sombra");
```

```
    perro2.setEdad(3);
```

```
    perro3 = new perro();
```

```
    perro3.setNombre("Zeus");
```

```
    perro3.setEdad(7);
```

```
    System.out.println(perro1.getNombre() + " tiene " + perro1.getEdad() + " años.");  
    System.out.println(perro2.getNombre() + " tiene " + perro2.getEdad() + " años.");  
    System.out.println(perro3.getNombre() + " tiene " + perro3.getEdad() + " años.");
```

```
    perro1.cambiar("Jaque");
```

```
    perro2.cambiar(4);
```

```
    perro3.cambiar("Goku", 8);
```

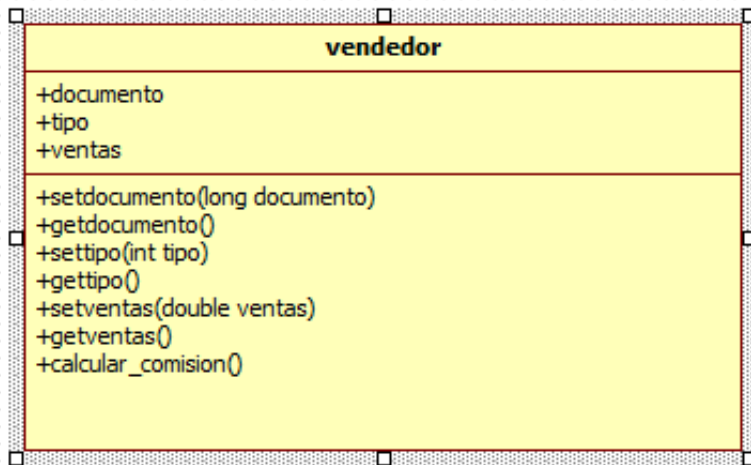
```
    System.out.println("\nDespués de los cambios:");
```

```
    System.out.println(perro1.getNombre() + " tiene " + perro1.getEdad() + " años.");  
    System.out.println(perro2.getNombre() + " tiene " + perro2.getEdad() + " años.");  
    System.out.println(perro3.getNombre() + " tiene " + perro3.getEdad() + " años.");
```

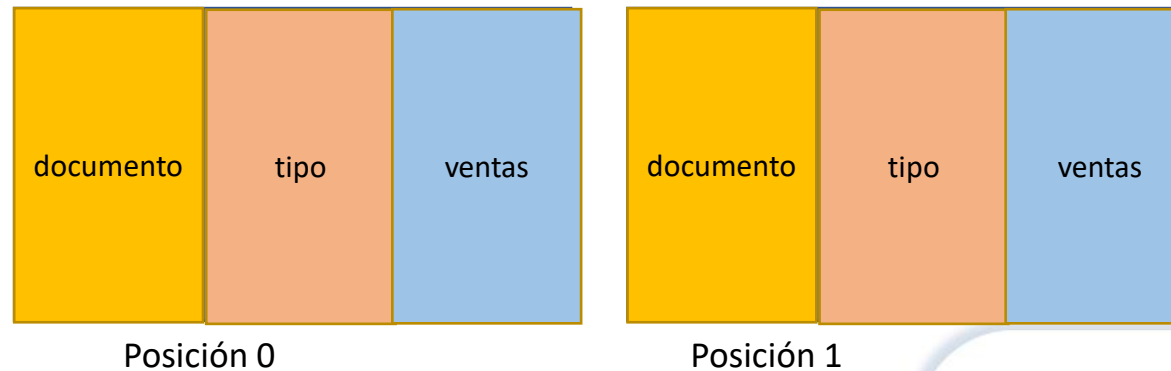



Array de Objetos (Clase ArrayList)

Es una clase, que permite la creación de una lista o array donde se almacenan objetos de una clase específica, que facilita el procesamiento de los objetos en un programa en Java.



ArrayList de la clase vendedor





Programación Orientada a Objetos

Ejercicio



Ejemplo: ArrayList de objetos

Dada la siguiente información sobre los N **suscriptores** del servicio de agua, de los cuales se conoce:

- Código
- Nombre
- Estrato(1,2,3,4,5)
- Consumo

Se pide calcular el valor a pagar por concepto de servicio de agua de cada suscriptor y el TOTAL (Todos), de acuerdo con la siguiente indicación:

El valor del servicio de agua es la suma del valor de la tarifa básica y el valor del consumo. La tarifa básica depende del estrato, así:

Estrato	Tarifa Básica
1	10.000
2	15.000
3	20.000
4	25.000
5	30.000

El valor del consumo es igual al consumo por \$100.

Realizar el programa en Java que resuelva la situación problema presentada, utilizando el concepto de Clases y Objetos (POO) y ArrayList.



El futuro digital
es de todos

MinTIC

Programación Orientada a Objetos



Clase

Suscriptor

+codigo
+nombre
+estrato
+consumo

+setCodigo(codigo)
+getCodigo()
+setNombre(nombre)
+getNombre()
+setEstrato(estrato)
+getEstrato()
+setConsumo(consumo)
+getConsumo()
+factura_suscriptor()

Ejercicios





Programación Orientada a Objetos

Ejercicios



Método

Parámetros de entrada

Estrato, consumo



Factura_suscriptor()

Selección múltiple (switch) para
calcular tarifa básica

Calcular valor del consumo

Valor_pagar=tarifa básica + valor
consumo



Parámetros de salida

vp

FUNCIÓN retorna o regresa un solo valor



Ejercicio: (Clase)



```
public class suscriptor {  
    private Long codigo;  
    private String nombre;  
    private int estrato;  
    private Long consumo;  
  
    public suscriptor() {  
    }  
    public suscriptor(Long codigo, String nombre, int estrato, Long consumo) {  
        this.codigo = codigo;  
        this.nombre = nombre;  
        this.estrato = estrato;  
        this.consumo = consumo;  
    }  
    public void setcodigo(long codigo) {  
        this.codigo=codigo;  
    }  
    public void setnombre(String nombre) {  
        this.nombre=nombre;  
    }  
    public void setestrato(int estrato) {  
        this.estrato=estrato;  
    }  
    public void setconsumo(long consumo) {  
        this.consumo=consumo;  
    }  
    public Long getcodigo() {  
        return codigo;  
    }  
}
```

```
    public String getnombre() {  
        return nombre;  
    }  
    public int getestrato() {  
        return estrato;  
    }  
    public Long getconsumo() {  
        return consumo;  
    }  
    public double factura_suscriptor() {  
        double tb=0, vp;  
        switch (this.estrato) {  
            case 1: tb=10000;break;  
            case 2: tb=15000;break;  
            case 3: tb=20000;break;  
            case 4: tb=25000;break;  
            case 5: tb=30000;break;  
        }  
        vp=tb+this.consumo*100;  
        return vp;  
    }  
}
```




```
import java.util.ArrayList;
import java.util.Scanner;
public class ArrayList_G91 {

    public static void main(String[] args) {
        // Definición de la consola
        Scanner consola=new Scanner(System.in);
        consola.useDelimiter("\n");
        //Definición de las variables
        long cedula;
        String nombre;
        int N,clase;
        double pasajes,encomiendas,comision,total;
        //Definición de la variable Objeto
        Conductor obj_conductor;
        //definición del arreglo de objetos
        ArrayList<Conductor> lista=new ArrayList();
        //Llenar el arreglo de objetos
        System.out.println("Cantidad de conductores: ");
        N=consola.nextInt();
```



```
for(int i=1;i<=N;i++){
    System.out.println("Cédula: ");
    cedula=consola.nextLong();
    System.out.println("Nombre: ");
    nombre=consola.next();
    consola.nextLine();
    System.out.println("Clase (1=Experto,2=Novato): ");
    clase=consola.nextInt();
    System.out.println("Valor Pasajes del mes: ");
    pasajes=consola.nextDouble();
    System.out.println("Valor encomiendas del mes: ");
    encomiendas=consola.nextDouble();
    //Crear el objeto
    obj_conductor=new Conductor(cedula,nombre,clase,pasajes,encomiendas);
    //Guardar el objeto en el ArrayList de objetos
    lista.add(obj_conductor);
}
//Procesar el Arreglo de conductores
total=0;
for(int i=0;i<lista.size();i++){
    comision=lista.get(i).liquidar_comision();
    total=total+comision;
    System.out.println("Nombre del conductor: "+lista.get(i).getNombre());
    System.out.println("Vaalor de la Comisión: "+comision);
}
System.out.println("Valor total comisiones: "+total);
}
```



El futuro digital
es de todos

MinTIC

Ciclo: Programación Básica

**Programación Orientada a Objetos
(POO)**

»

Misión TIC 2022

xxx



**Misión
TIC 2022**