Nivel 2:
Definiendo Situaciones y Manejando Casos

Creación de Objetos,

Instrucciones Condicionales

Creación de un Objeto

- Para inicializar los valores de un objeto, se usan los métodos CONSTRUCTORES:
  - Son invocados automáticamente en el momento de ejecutar la instrucción de creación

- Reglas:
  - El método constructor se debe llamar igual a la clase
  - No puede tener ningún tipo de retorno

Creación de un Objeto

- Método de creación de un Producto con valores predefinidos (sin parámetros)

```
public Producto ( )
{
   tipo = 0;
   nombre = “”;
   valorUnitario = 0.0;
   cantidadBodega = 0;
   cantidadMinima = 0;
   totalProdVendidos = 0;
}
```

Su nombre siempre es igual al nombre de la clase

Creación de un Objeto

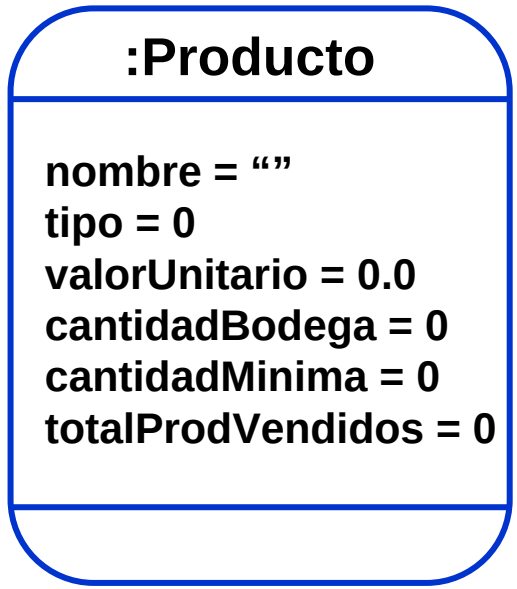
- Método de creación de un Producto con parámetros

```
public Producto (int tip, String nom, double val, int cant, int min)
{
    tipo = tip;
    nombre = nom;
    valorUnitario = val;
    cantidadBodega = cant;
    cantidadMinima = min;
    totalProdVendidos = 0;
}
```

Su nombre siempre es igual al nombre de la clase

Creación de un Objeto

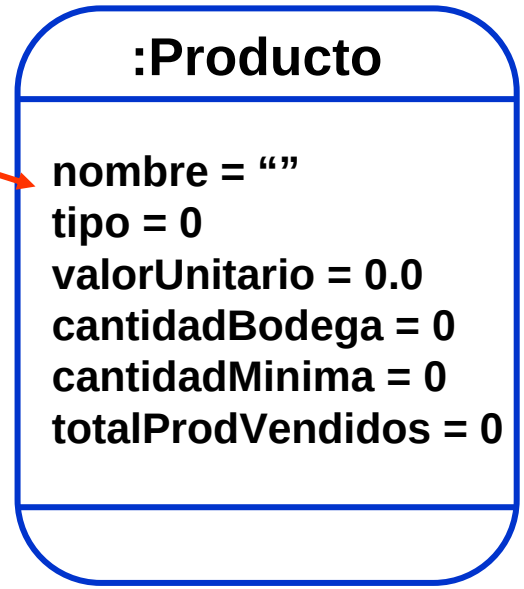
Producto p = new Producto ( );



Creación de un Objeto

Producto p = new Producto ( );

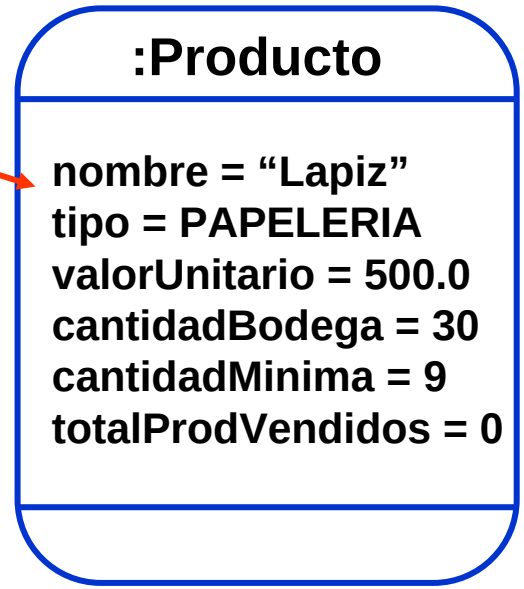
Atributos Inicializados



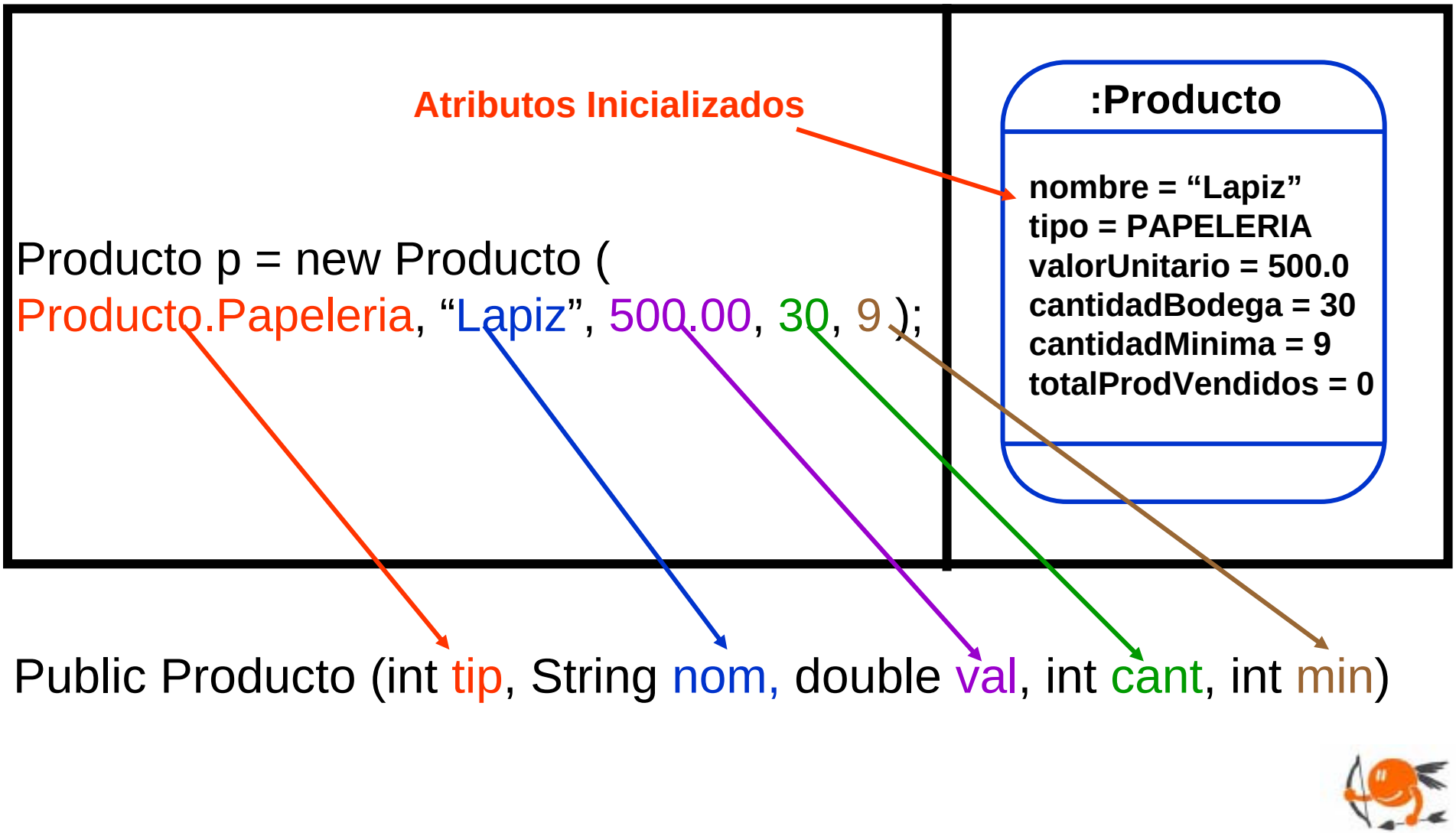
Creación de un Objeto

Atributos Inicializados

```
Producto p = new Producto (
Producto.Papeleria, “Lapiz”, 500.00, 30, 9 );
```



Creación de un Objeto



Quién instancia los productos?

R// La Tienda

Quién crea los productos y la tienda?

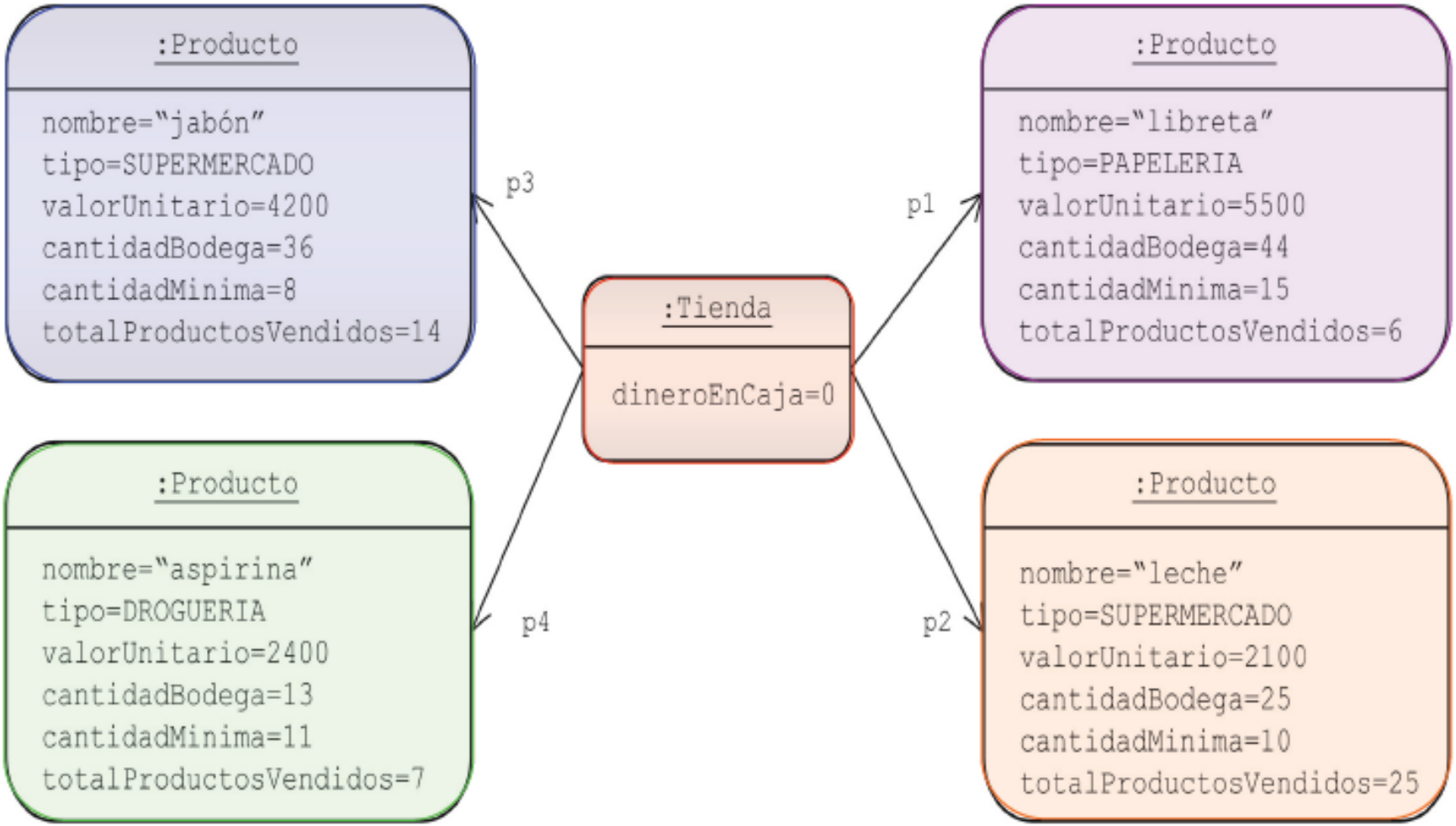
R// La Interfaz

En el constructor de la interfaz

```
public InterfazTienda( )
{
  // Crea los 4 productos de la tienda
  Producto p1 = new Producto( Producto.PAPELERIA, "lápiz", 550.0, 18, 5 );
  Producto p2 = new Producto( Producto.DROGUERIA, "aspirina", 109.5, 25, 8 );
  Producto p3 = new Producto( Producto.PAPELERIA, "borrador", 207.3, 30, 10 );
  Producto p4 = new Producto( Producto.SUPERMERCADO, "pan", 150.0, 15, 20 );

  // Crea la tienda con sus 4 productos
  tienda = new Tienda( p1, p2, p3, p4 );
}
```

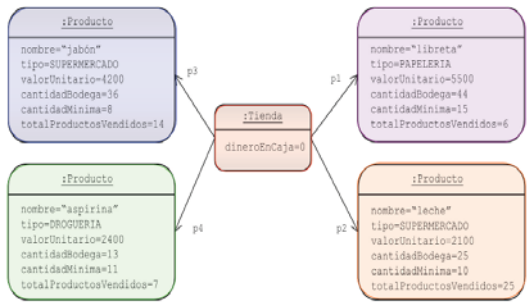
Escenario posible de la tienda



Creación de ese escenario

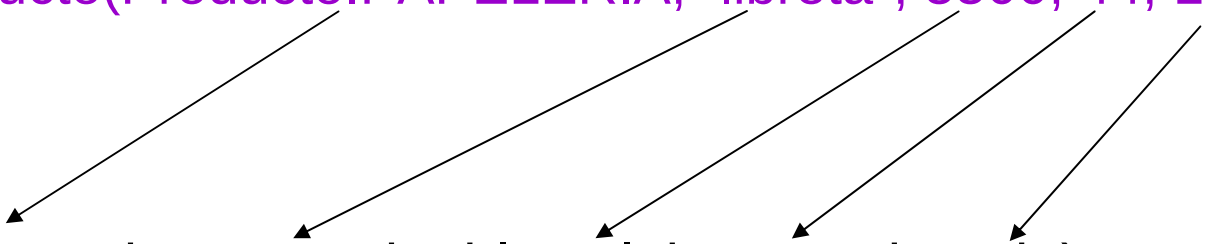
1. Crear los 4 productos como variables locales del método InterfazTienda

```
public InterfazTienda( )
{
  Producto x = new Producto(Producto.PAPELERIA,
                                        “libreta", 5500, 44, 15 );
  Producto y = new Producto(Producto.SUPERMERCADO,
                                        “leche", 2100, 25, 10 );
  Producto z = new Producto(Producto.SUPERMERCADO,
                                        “jabón", 4200, 36, 8 );
  Producto w = new Producto(Producto.DROGUERIA,
                                         “aspirina", 2400, 13, 11 );
   tienda = new Tienda( x, y, z, w);
}
```



Qué pasa cuando se hace new Producto

Producto **X**= new Producto(Producto.PAPELERIA, “libreta", 5500, 44, 15);



```
public Producto (int tip, String nom, double val, int cant, int min)
{
  tipo = tip;
  nombre = nom;
  valorUnitario = val;
  cantidadBodega = cant;
  cantidadMinima = min;
  totalProdVendidos= 0;
}
```

R// Se ejecuta automáticamente el método constructor de la clase Producto con los parámetros en ORDEN correcto

Qué pasa cuando se hace new Producto

Producto **X**= new Producto(Producto.PAPELERIA, “libreta", 5500, 44, 15);

```
public Producto (int tip, String nom, double val, int cant, int min)
{
  tipo = tip;
  nombre = nom;
  valorUnitario = val;
  cantidadBodega = cant;
  cantidadMinima = min;
  totalProdVendidos= 0;
}
```

Parámetros del método, que son los valores que se asignan a los atributos del producto que se está creando

Atributos del producto que se está creando

No todos los valores que se asignan a los atributos en el método constructor entran como parámetros. En la creación del objeto se pueden también dar valores por defecto a algunos atributos

Resultado: se crea un nuevo objeto llamado X de la clase Producto

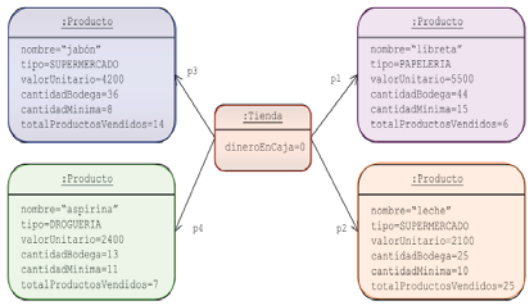
```
Producto X= new Producto(Producto.PAPELERIA, “libreta", 5500, 44, 15);
```

```
public Producto (int tip, String nom, double val, int cant, int min)
{
  tipo = tip;
  nombre = nom;
  valorUnitario = val;
  cantidadBodega = cant;
  cantidadMinima = min;
  totalProdVendidos= 0;
}
```

Volvamos a InterfazTienda…

1. Crear los 4 productos como variables locales del método InterfazTienda

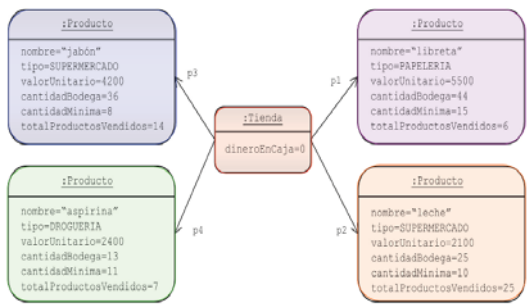
```
public InterfazTienda( )
{
  Producto x = new Producto(Producto.PAPELERIA,
                                    “libreta", 5500, 44, 15 );
  Producto y = new Producto(Producto.SUPERMERCADO,
                                    “leche", 2100, 25, 10 );
  Producto z = new Producto(Producto.SUPERMERCADO,
                                    “jabón", 4200, 36, 8 );
  Producto w = new Producto(Producto.DROGUERIA,
                                    “aspirina", 2400, 13, 11 );
   tienda = new Tienda( x, y, z, w);
}
```



Creación de la tienda

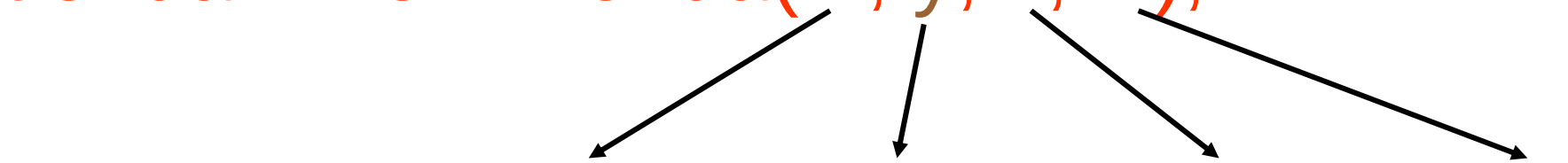
2. Crear la tienda, pasando como parámetros a su método constructor, las variables locales que contienen los productos

```
public InterfazTienda( )
{
  Producto x = new Producto(Producto.PAPELERIA,
                                        “libreta", 5500, 44, 15 );
  Producto y = new Producto(Producto.SUPERMERCADO,
                                        “leche", 2100, 25, 10 );
  Producto z = new Producto(Producto.SUPERMERCADO,
                                        “jabón", 4200, 36, 8 );
  Producto w = new Producto(Producto.DROGUERIA,
                                        “aspirina", 2400, 13, 11 );
  tienda = new Tienda( x, y, z, w);
}
```



Qué pasa cuando se hace new Tienda…

tienda = new Tienda(x, y, z, w);



```
public Tienda (Producto a1, Producto a2, Producto a3, Producto a4)
{
    p1 = a1;
    p2 = a2;
    p3 = a3;
    p4 = a4;
    dineroEnCaja= 0;
}
```

R// Se ejecuta automáticamente el método constructor de la clase Producto con los parámetros en ORDEN correcto

Qué pasa cuando se hace new Tienda…

```
tienda = new Tienda( x, y, z, w);
```

```
public Tienda (Producto a1, Producto a2, Producto a3, Producto a4)
{
    p1 = a1;
    p2 = a2;
    p3 = a3;
    p4 = a4;
    dineroEnCaja= 0;
}
```

Parámetros del método, que son los valores que se asignan a los atributos del producto que se está creando

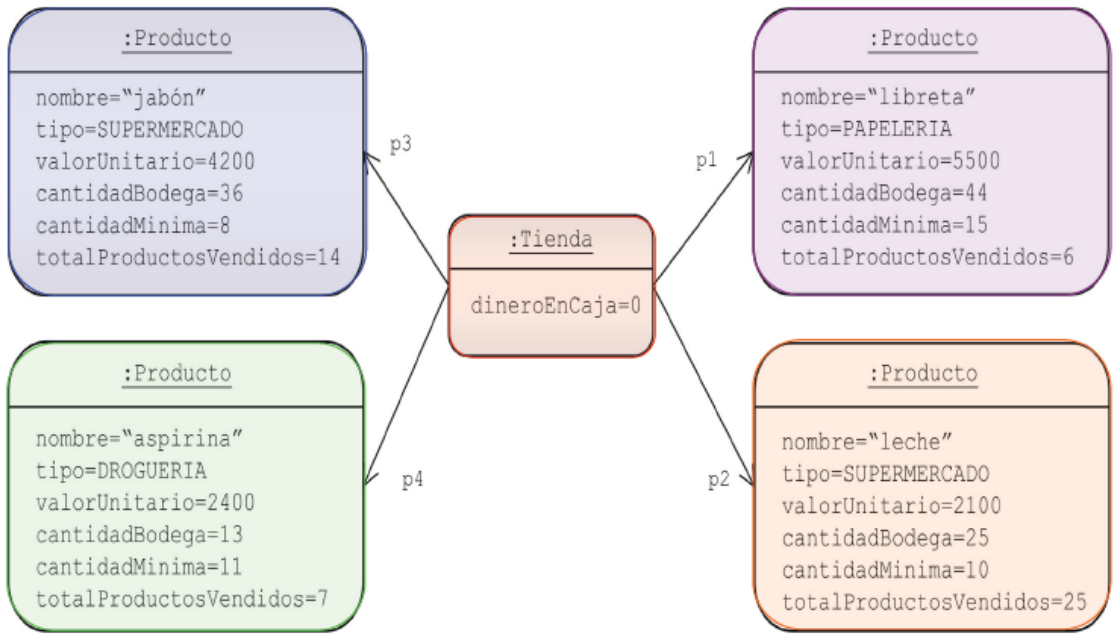
Atributos del producto que se está creando

No todos los valores que se asignan a los atributos en el método constructor entran como parámetros. En la creación del objeto se pueden también dar valores por defecto a algunos atributos

Resultado: se crea un nuevo objeto llamado tienda de la clase Tienda

tienda = new Tienda(x, y, z, w);

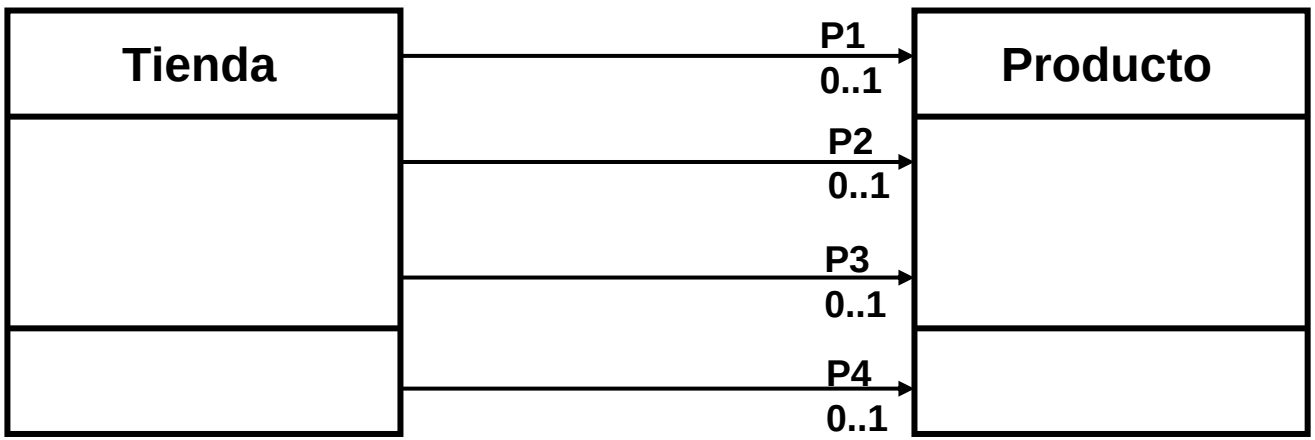
```
public Tienda (Producto a1, Producto a2, Producto a3, Producto a4)
{
    p1 = a1;
    p2 = a2;
    p3 = a3;
    p4 = a4;
    dineroEnCaja= 0;
}
```



Manejo de asociaciones opcionales

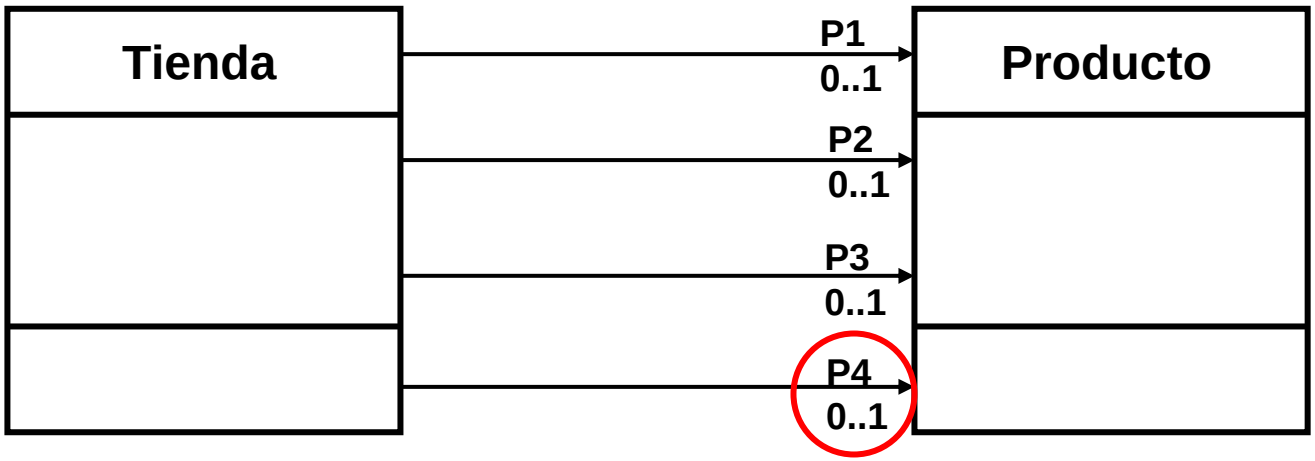
Asociaciones opcionales

- En algunos problemas, las asociaciones pueden o no existir.
- Ejemplo:
  - La tienda puede tener 1, 2, 3 ó4 productos.
  - No todos tienen que existir obligatoriamente.



En UML

Asociaciones opcionales

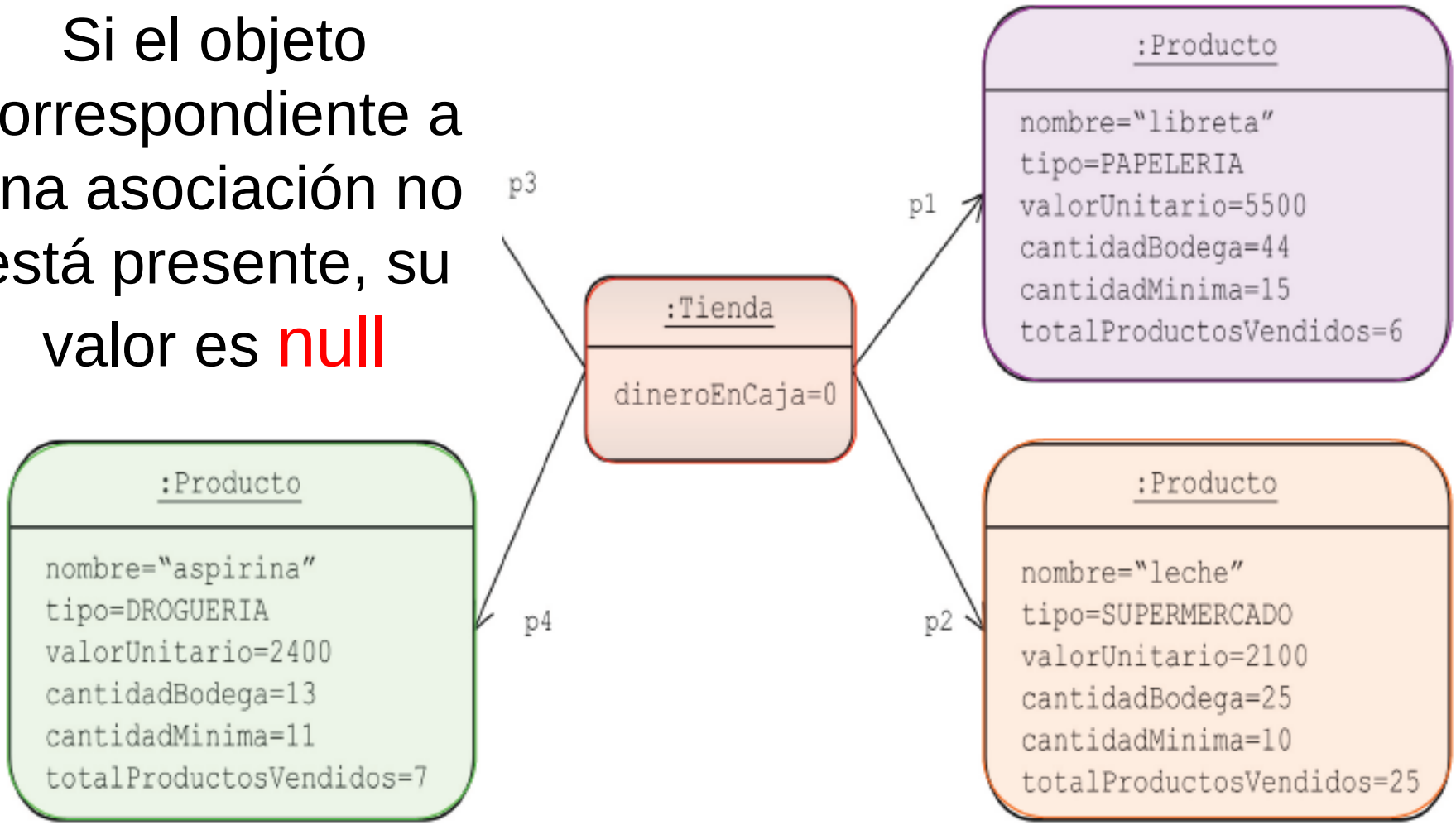


La cardinalidad de la asociación es cero o uno (0..1) para indicar que puede o no existir el objeto

Si en el diagrama no aparece ninguna cardinalidad, se interpreta como 1 (existe exactamente un objeto de la otra clase)

Asociaciones opcionales

Si el objeto correspondiente a una asociación no está presente, su valor es null



Instrucciones Condicionales

Cuándo usar instrucciones condicionales?

- Cuando necesitamos dar una solución a un problema considerando distintos CASOS que se pueden presentar

- Dependiendo del CASO (que se expresa con una CONDICION) se ejecuta una acción diferente.

Cuándo usar instrucciones condicionales?

- El siguiente ejemplo nos muestra la clase “Producto”, que se encarga de vender cierta cantidad de unidades presentes en la bodega:

Producto
String nombre int tipo double valorUnitario Int cantidadBodega Int cantidadMinima Int totalProductosVendidos

Pueden existir dos casos posibles en el caso de una venta:

a. **La cantidad a vender es mayor que la disponible en bodega.**

b. **Hay suficientes unidades del producto para vender.**

Cual seria la solución?

Instrucciones Condicionales

- Cuando piden mas de lo que hay…

  cantidadPedida > cantidadBodega

  Si esto se cumple → Se venden todas las unidades disponibles.

  ```
  totalProductosVendidos += cantidadBodega;
  cantidadBodega = 0;
  ```

- Cuando alcanza lo que piden con lo que hay en bodega.

  cantidadPedida <= cantidadBodega

  Si esto se cumple → Se vende la cantidad pedida por el usuario.

  ```
  totalProductosVendidos += cantidad;
  cantidadBodega -= cantidad;
  ```

Instrucción if-else

If (condicion)
{
    instrucciones que se deben ejecutar si se cumple la condición
}
else
{
    instrucciones que se deben ejecutar si NO se cumple la condición
}

Instrucción if - else (Ejemplo 1)

Permite expresar los casos dentro de un método

```
public class Producto
{
 …
    public void vender ( int cantidad )

    if ( cantidad > cantidadBodega )
    {
            totalProductosVendidos += cantidadBodega;
            cantidadBodega = 0;
    else
    {
            totalProductosVendidos += cantidad;
            cantidadBodega -= cantidad;
    }
}
```

Condición: expresión lógica

Solución al primer caso

Solución al segundo caso

Instrucción if - else (Ejemplo 2)

```
Class Producto
{
    public boolean haySuficienteParaVender( )
    {
      boolean suficiente;
      if ( cantidadBodega > cantidadMinima)
      {
            suficiente = true;
       }
      else
       {
            suficiente = false;
       }
      return suficiente;
  }
}
```

Instrucción if - else (Ejemplo 2 otra opción)

```
ClassProducto
{
    public boolean haySuficienteParaVender( )
    {
        return ( cantidadBodega> cantidadMinima);
    }
}
```

Instrucción if - else (Ejemplo 3)

Dar el precio final de un producto de papelería con o sin IVA dependiendo del parámetro que lo indica

```
ClassProducto
{
   public double darPrecioFinalPapeleria( boolean conIVA)
   {
          double precioFinal= valorUnitario;
          if ( conIVA== true)
          {
              precioFinal= valorUnitario+ (valorUnitario* IVA_PAPEL);
          }
          return precioFinal;
     }
}
```

Condicionales en Cascada = Varios CASOS

```
If (condicion1)
{
    instrucciones que se deben ejecutar si se cumple la condición1
}
else if (condicion2)
    {
      instrucciones que se deben ejecutar si se cumple la condición2
    }
    else if (condicion3)
          {
            instrucciones que se deben ejecutar si se cumple la
            condición3
          }
          else
          {
             instrucciones que se deben ejecutar si no se cumple
             ninguna de  las condiciones anteriores
          }
```

Ejemplo – En la Clase Producto

```
public double darIVA( )
{
   if ( tipo == PAPELERIA)
   {
       returnIVA_PAPEL;
    }
    else if (tipo == SUPERMERCADO)
            {
                return IVA_MERCADO;
            }
            else
            {
               return IVA_FARMACIA;
            }
}
```

Ejemplo – Otra opción

```
public double darIVA( )
{
  doubleresp= 0.0;
  if( tipo == PAPELERIA)
  {
    resp= IVA_PAPEL;
   }
   else if (tipo == SUPERMERCADO)
   {
      resp= IVA_MERCADO;
   }
  else
  {
     resp= IVA_FARMACIA;
  }
 return resp;
}
```

Tarea

Objetivo: Practicar el uso de instrucciones condicionales para expresar el cambio de estado que debe hacerse en un objeto, en cada uno de los casos indicados.

Escriba el código de cada uno de los métodos descritos a continuación. Tenga en cuenta la clase en la cual está el método y la información que se entrega como parámetro.

Ejercicio – En la Clase Producto

```
public void hacerPedido(int cantidadPedido)
{




}
```

Recibir un pedido, solo si en bodega se tienen menos unidades de las indicadas en el tope mínimo. En caso contrario, el método no debe hacer nada

Ejercicio – En la Clase Producto

```
public void cambiarValorUnitario( int cantidadPedido)
{




}
```

Modificar el precio del producto, utilizando la siguiente política: si el producto es de droguería o papelería debe disminuir un 10%. Si es de supermercado, debe aumentar un 5%.

Ejercicio – En la Clase Tienda

```
public int venderProducto( String nombreProducto, int cantidad)

{



}
```

Vender una cierta cantidad del producto cuyo nombre es igual al recibido como parámetro. El método retorna el número de unidades efectivamente vendidas. Utilice el método vender de la clase Producto como parte de su solución.

Ejercicio – En la Clase Tienda

```
public int cuantosPapeleria ()
{



}
```

Calcular el número de productos de papelería que hay en la tienda.

Condicionales Compuestas

- SWITCH:
  Es una manera alterna de expresar la solución de un problema, para el cual existe un conjunto de casos. Esta instrucción tiene la restricción de que cada caso debe estar identificado con un valor entero.

```
Switch ( expresion)
{
    case valor1 : instrucciones que se deben ejecutar si la
                  expresión tiene el valor1
                   break;
    case valor2 : instrucciones que se deben ejecutar si la
                  expresión tiene el valor2
                  break;
    case valor3 : instrucciones que se deben ejecutar si la
                  expresión tiene el valor1
                  break;
}
```

Ejemplo sin switch

```
public double darIVA( )
{
   doubleiva= 0.0;
   if ( tipo == PAPELERIA)
   {
        iva=  IVA_PAPEL;
   }
   else if (tipo == SUPERMERCADO)
   {
      iva= IVA_MERCADO;
    }
   else
   {
     iva= IVA_FARMACIA;
   }
   return iva;
}
```

Ejemplo con switch

```
public double darIVA( )
{
   double iva= 0.0;
   switch ( tipo)
   {
       case PAPELERIA :      iva= IVA_PAPEL;
                             break;
      case SUPERMERCADO: iva= IVA_MERCADO;
                             break;
      case DROGUERIA :       iva= IVA_FARMACIA;
                             break;
   }
   return iva;
}
```

Responsabilidades de una Clase

Métodos Constructores

Inicializan los valores de los atributos de un objeto durante su proceso de creación. “Crean”.

- Método de creación de un Producto con valores predefinidos (sin parámetros)

```
public Producto ( )
{
        tipo = 0;
        nombre = “”;
        valorUnitario = 0.0;
        cantidadBodega = 0;
        cantidadMinima = 0;
        totalProductosVendidos = 0;
}
```

Métodos Constructores

- Método de creación de un Producto con parámetros

```
public Producto(int tip, String nom, double val, int cant, int min )
{
    tipo = tip;
    nombre = nom;
    valorUnitario = val;
    cantidadBodega = cant;
    cantidadMinima = min;
    totalProductosVendidos = 0;
}
```

Métodos Modificadores

- Cambian el estado de los objetos de una clase. “Hacen”.

Método de cambio de valor de un Producto

```
public void cambiarValorUnitario ( double nuevoValor )
{
    valorUnitario = nuevoValor;
}
```

Métodos Analizadores

- Calculan información a partir del estado de los objetos. “Saben”.

Método de retorno del valor de un Producto

```
public double darValorUnitario ( )
{
    return valorUnitario;
}
```