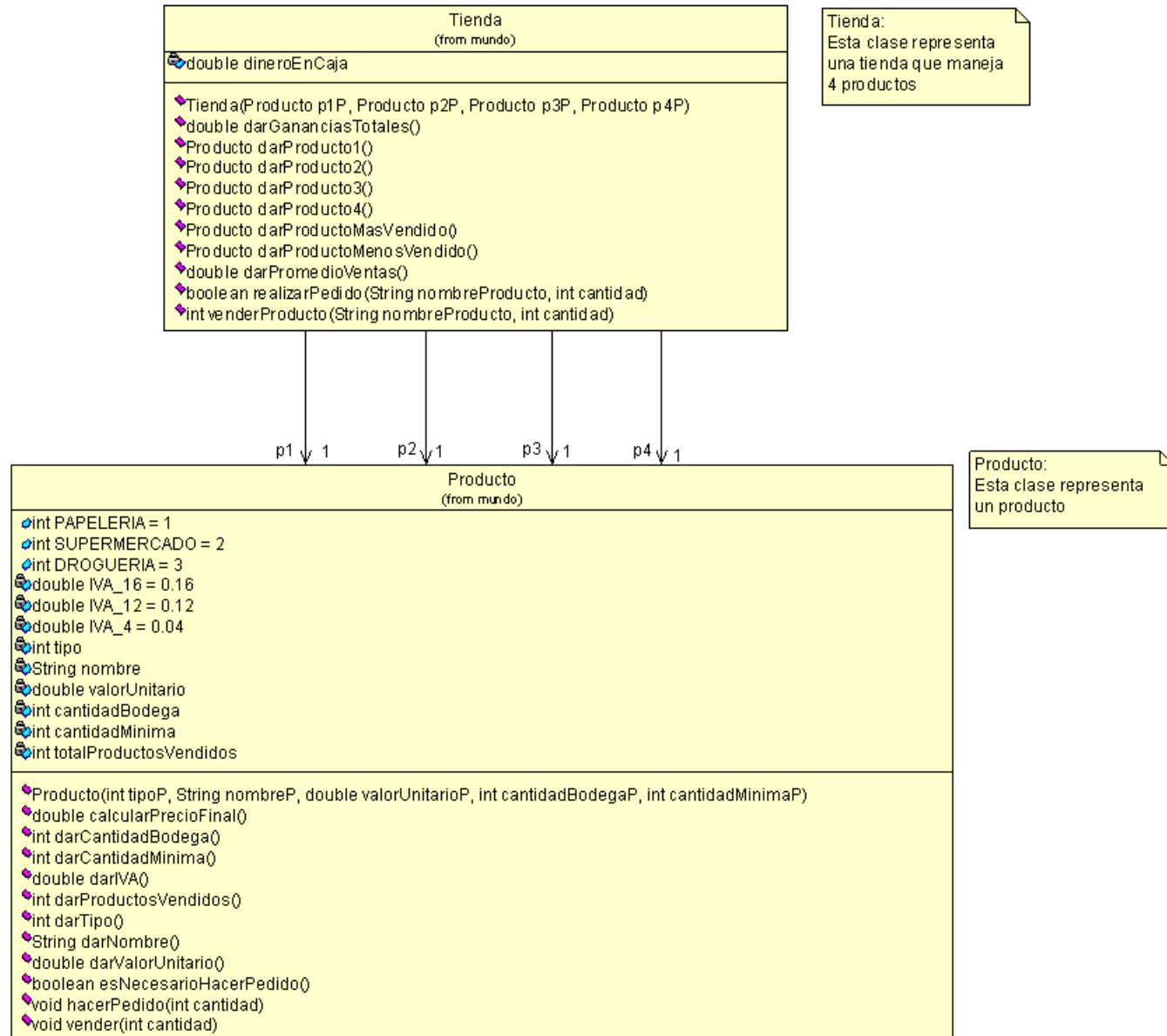


Tarea 3

Objetivo: Generar habilidad en la utilización de las asignaciones y las expresiones como medio para transformar el estado de un objeto.

Modelo del Mundo de la Tienda



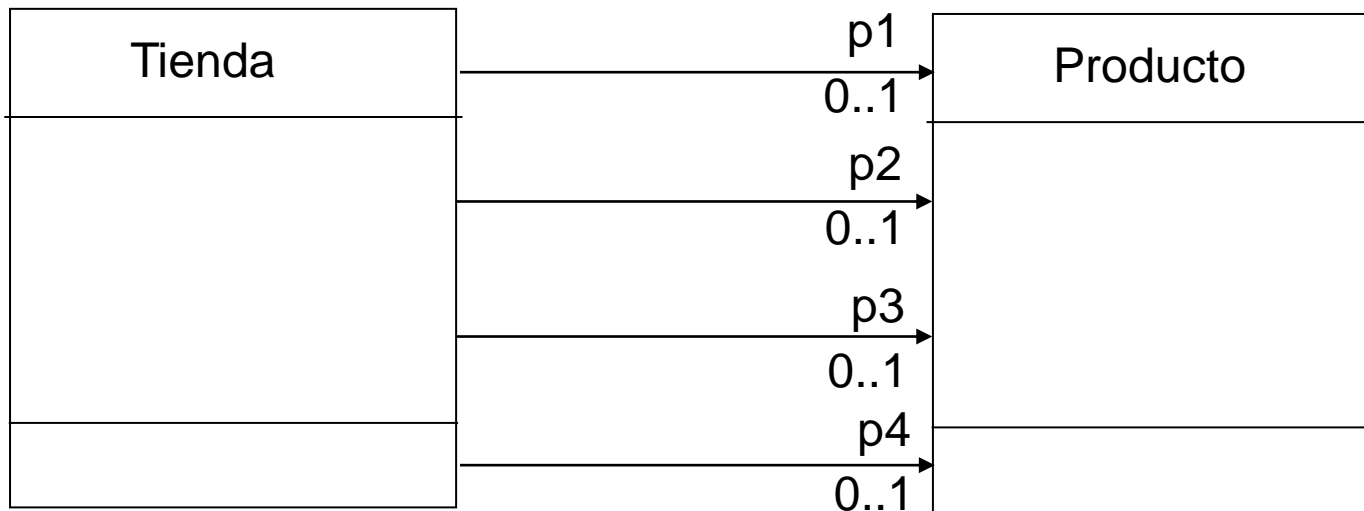
En un método de la clase...	la siguiente modificación de estado...	se logra con las siguientes instrucciones...
Producto	Se vendieron 5 unidades del producto (suponga que hay suficientes).	<code>totalProductosVendidos += 5;</code> <code>cantidadBodega -= 5;</code>
Producto	El valor unitario se incrementa en un 10%	
Producto	Se incrementa en uno la cantidad mínima para hacer pedidos.	
Producto	El producto ahora se clasifica como de SUPERMERCADO	
Producto	Se cambia el nombre del producto. Ahora se llama “teléfono”.	

En un método de la clase...	la siguiente modificación de estado...	se logra con las siguientes instrucciones...
Tienda	Se asigna al dinero en caja de la tienda la suma de los valores unitarios de los cuatro productos.	<pre> dineroEnCaja = p1.darValorUnitario() + p2.darValorUnitario() + p3.darValorUnitario() + p4.darValorUnitario(); </pre>
Tienda	Se venden 4 unidades del producto 3 (suponga que están disponibles).	
Tienda	Se decrementa en un 2% el dinero en la caja.	
Tienda	Se hace un pedido de la mitad de la cantidad mínima de cada producto, suponiendo que la cantidad en bodega de todos los productos es menor al tope mínimo.	
Tienda	Se pone en la caja el dinero correspondiente a las unidades vendidas de todos los productos de la tienda.	

Manejo de asociaciones opcionales

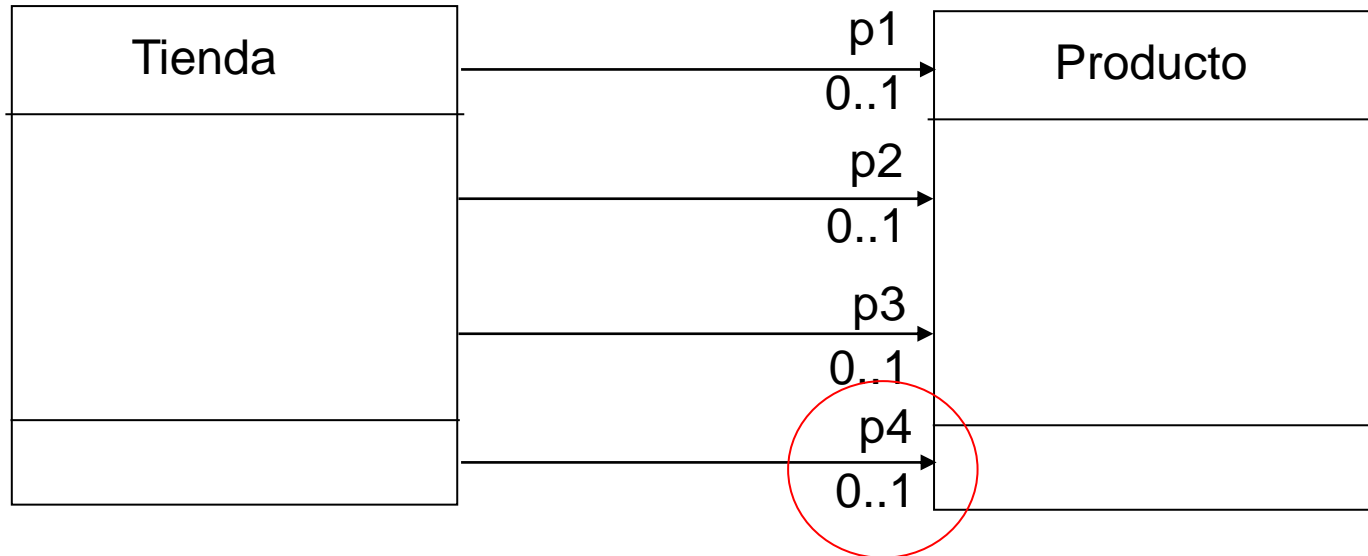
Asociaciones opcionales

- En algunos problemas, las asociaciones pueden o no existir.
- Ejemplo:
 - La tienda puede tener 1, 2, 3 ó 4 productos.
 - No todos tienen que existir obligatoriamente.



En UML

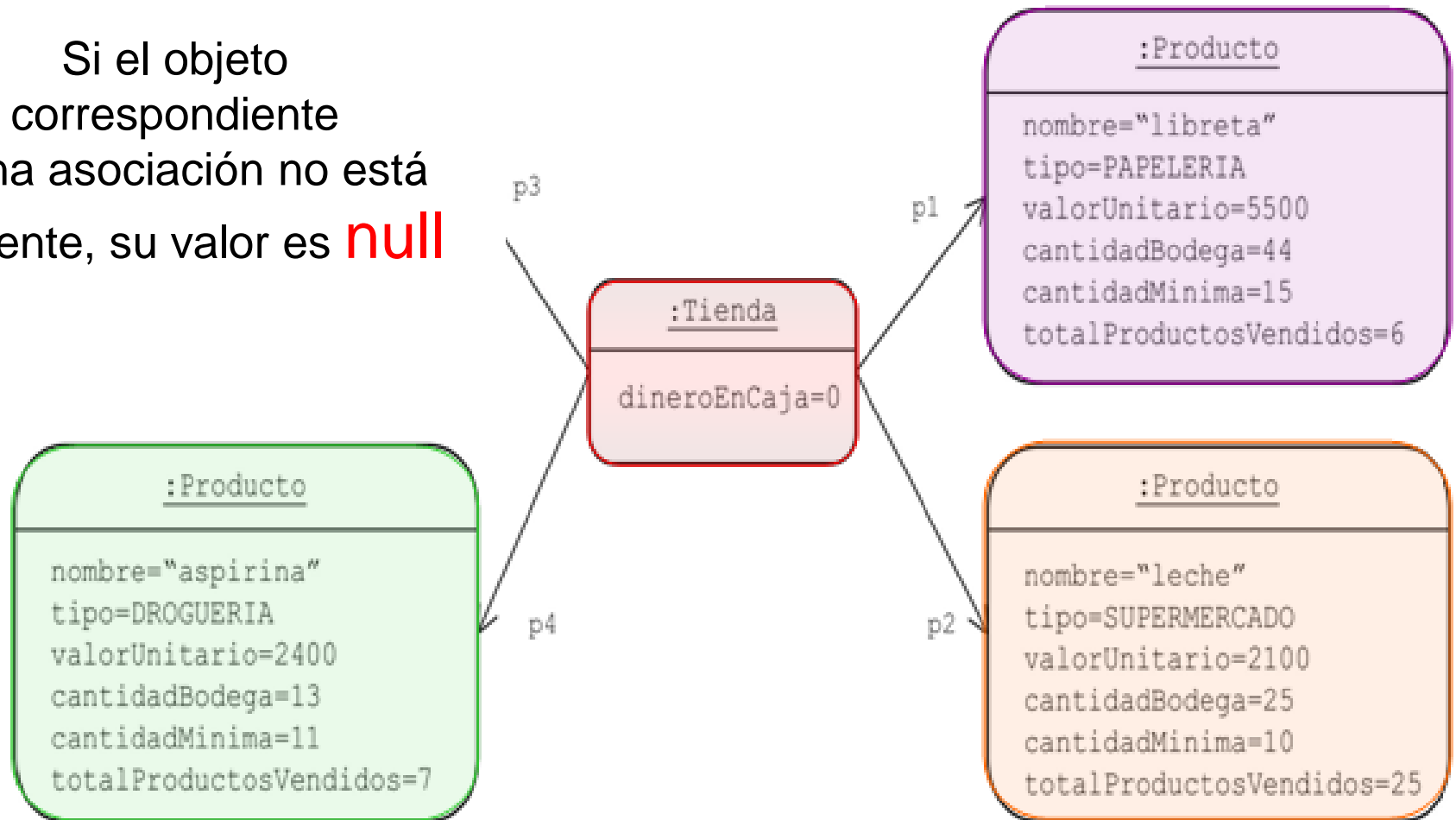
Asociaciones opcionales



- 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.

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

```
}
```

Ejemplo 1 en la clase Producto

Class Producto

```
{  
    public int vender ( int cant )  
    {  
        int unidadesVendidas = 0;  
        if ( cant > cantidadBodega )  
        {  
            totalProductosVendidos += cantidadBodega;  
            unidadesVendidas = cantidadBodega;  
            cantidadBodega = 0;  
            return unidadesVendidas;  
        }  
        else  
        {  
            totalProductosVendidos += cant;  
            cantidadBodega -= cant;  
            return cant;  
        }  
    }  
}
```

Ejemplo 1 en la clase Producto

Class Producto

```
{
    public int vender ( int cant )
    {
        int unidadesVendidas = 0;
        if ( cant > cantidadBodega )
        {
            totalProductosVendidos += cantidadBodega;
            unidadesVendidas = cantidadBodega;
            cantidadBodega = 0;
        }
        else
        {
            totalProductosVendidos += cant;
            cantidadBodega -= cant;
            unidadesVendidas = cant;
        }
        return unidadesVendidas;
    }
}
```

Ejemplo 2 en la clase Producto

Class Producto

```
{
    public boolean haySuficienteParaVender ( )
    {
        boolean suficiente = true;

        if ( cantidadBodega > 0)
        {
            suficiente = true;
        }
        else
        {
            suficiente = false;
        }

        return suficiente;
    }
}
```

Ejemplo 2 en la clase Producto

Class Producto

```
{
    public boolean haySuficienteParaVender ( )
    {
        boolean suficiente = true;

        if ( cantidadBodega == 0)
        {
            suficiente = false;
        }

        return suficiente;
    }
}
```

Ejemplo 2 en la clase Producto

Class Producto

```
{
    public boolean haySuficienteParaVender ( )
    {
        boolean suficiente = false;

        if ( cantidadBodega > 0)
        {
            suficiente = true;
        }

        return suficiente;
    }
}
```


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)
    {
        return IVA_PAPEL;
    }
    else if ( tipo == SUPERMERCADO)
    {
        return IVA_MERCADO;
    }
    else
    {
        return IVA_FARMACIA;
    }
}
```

Ejemplo – Otra opción

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

Ejemplo – En la Clase Tienda

```
public int cuantosPapeleria ( )
{
    int cuantos = 0;
    if (p1.darTipo( ) == Producto.PAPELERIA)
    {
        cuantos = cuantos + 1;
    }
    if (p2.darTipo( ) == Producto.PAPELERIA)
    {
        cuantos = cuantos + 1;
    }
    if (p3.darTipo( ) == Producto.PAPELERIA)
    {
        cuantos = cuantos + 1;
    }
    if (p4.darTipo( ) == Producto.PAPELERIA)
    {
        cuantos = cuantos + 1;
    }
    return cuantos;
}
```

Calcula el número de productos en la Tienda que son de tipo PAPELERIA. El resultado del método es un valor **entero** entre 0 y 4

Ejemplo 2 en la clase Producto

Class Producto

```
{
    public boolean haySuficienteParaVender ( )
    {
        boolean suficiente;

        if ( cantidadBodega > cantidadMinima)
        {
            suficiente = true;
        }
        else
        {
            suficiente = false;
        }

        return suficiente;
    }
}
```

Ejemplo 2 – Otra opción

Class Producto

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

Ejemplo 3 en la clase Producto

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

Class Producto

```
{
    public double darPrecioFinalPapeleria ( boolean conIVA)
    {
        double precioFinal = valorUnitario;

        if ( conIVA == true)
        {
            precioFinal = valorUnitario + (valorUnitario * IVA_PAPEL);
        }

        return precioFinal;
    }
}
```

Ejemplo 3 – Otra opción

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

Class Producto

```
{
    public double darPrecioFinalPapeleria ( boolean conIVA )
    {
        double precioFinal = valorUnitario;

        if ( conIVA )
        {
            precioFinal = valorUnitario + (valorUnitario * IVA_PAPEL);
        }

        return precioFinal;
    }
}
```


Tarea 5 – En la Clase Producto

```
public void subirValorUnitario ( )
```

```
{
```

Labo

Aumentar el valor unitario del producto, utilizando la siguiente política: si el producto cuesta menos de \$1000, aumentar el 1%. Si cuesta entre \$1000 y \$5000, aumentar el 2%. Si cuesta mas de \$5000 aumentar el 3%

```
}
```

Tarea 5 – En la Clase Producto

```
public void hacerPedido ( int cantidad )  
{
```

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

```
}
```

Tarea 5 – En la Clase Producto

```
public void cambiarValorUnitario ( )  
{
```

Labo

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%.

```
}
```

Tarea 6 – En la Clase Producto

```
public String nombreTipoProducto ( )  
{
```

Dar el nombre del tipo del producto. Esto es, si por ejemplo el producto es de tipo SUPERMERCADO, el método debe retornar la cadena: “El producto es de supermercado”.

```
}
```

Tarea 6 – En la Clase Tienda

```
public double darPrecioProducto ( int numProd)  
{
```

Retornar el precio final
del producto identificado
con el número que llega
como parámetro. Por
ejemplo, si numProd es
3, debe retornar el precio
del tercer producto (p3).

```
}
```

Comparación de Strings

Método equals de la clase String

- Para saber si dos strings (string1 y string2) son idénticos:

`string1.equals (string2)`

- Es verdadero si `string1 == string2`, falso de lo contrario

Ejercicio – En la Clase Tienda

```
public void venderUnidad ( String nombreProducto )  
{
```

```
    String nombre1, nombre2, nombre3, nombre4;
```

```
    nombre1 = p1.darNombre( );
```

```
    nombre2 = p2.darNombre( );
```

```
    nombre3 = p3.darNombre( );
```

```
    nombre4 = p4.darNombre( );
```

Vender una unidad del
producto que tiene el
nombre que llega como
parámetro

```
    if ( nombre1.equals( nombreProducto ) == true )
```

```
        p1.vender ( 1 );
```

```
    else if ( nombre2.equals( nombreProducto ) == true )
```

```
        p2.vender ( 1 );
```

```
    else if ( nombre3.equals( nombreProducto ) == true )
```

```
        p3.vender ( 1 );
```

```
    else if ( nombre4.equals( nombreProducto ) == true )
```

```
        p4.vender ( 1 );
```

```
}
```


Tarea 5 – 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.

```
}
```

Tarea 6 – En la Clase Tienda

```
public double darPrecioProducto ( String nomProd )  
{
```

Retornar el precio final
del producto identificado
con el nombre que llega
como parámetro.

```
}
```

Condicionales Compuestas

```
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 ( )
{
    double iva = 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;
}
```

Quién instancia los productos ?

Quién instancia los productos ?

R// La Tienda

En el método constructor de la Tienda

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

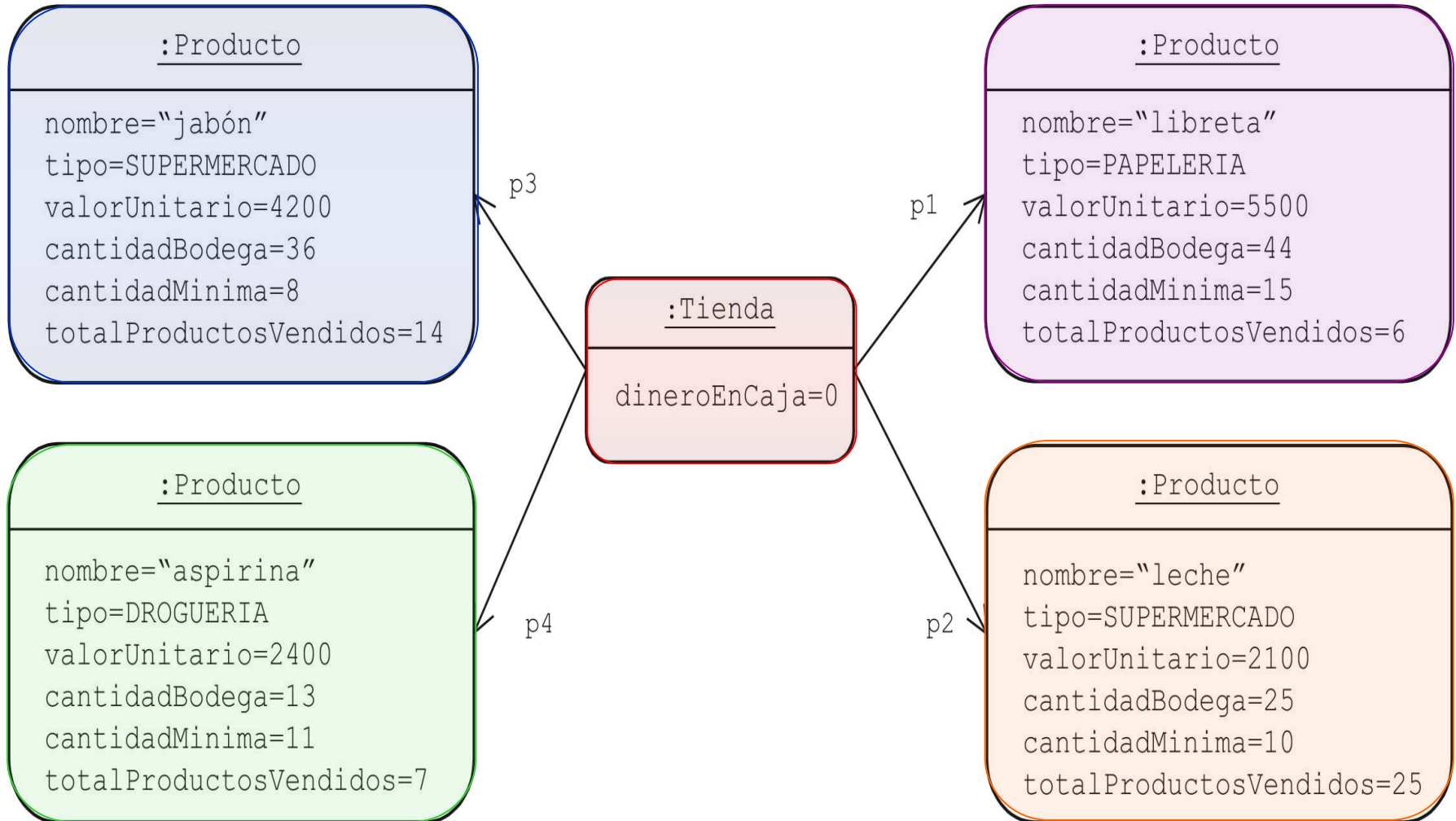

Quién crea los productos y la tienda?

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

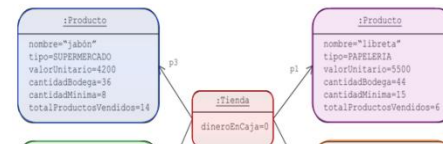


Creación de ese escenario

```
public InterfazTienda( )  
{
```

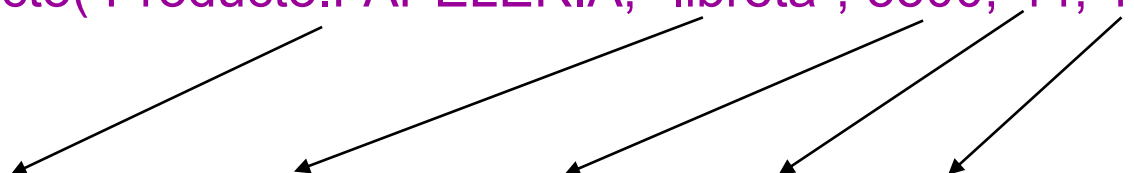
**1. Crear los 4 productos
como variables locales del
método InterfazTienda**

```
    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;
    totalProductosVendidos = 0;
}
```

The diagram illustrates the mapping of arguments from the constructor call to the constructor signature. Arrows point from the arguments in the call to the corresponding parameters in the signature: from 'Producto.PAPELERIA' to 'tip', from '"libreta"' to 'nom', from '5500' to 'val', from '44' to 'cant', and from '15' to 'min'.

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

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;

totalProductosVendidos = 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

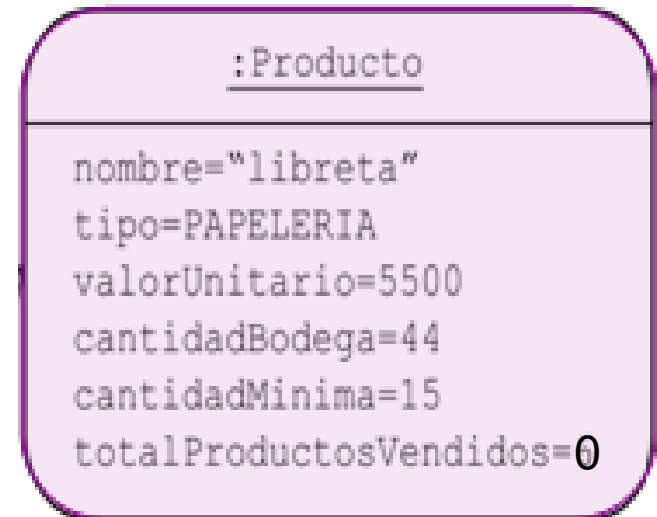
No todos los valores que se asignan a los atributos en el método constructor entran como parámetros. En la creación del objeto se pueden también dar valores por defecto a algunos atributos

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;
    totalProductosVendidos = 0;
}
```

X

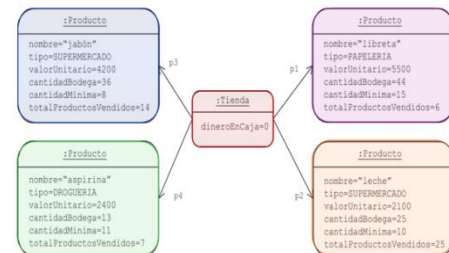


Volvamos a InterfazTienda ...

```
public InterfazTienda( )  
{
```

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

```
public InterfazTienda( )  
{
```

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

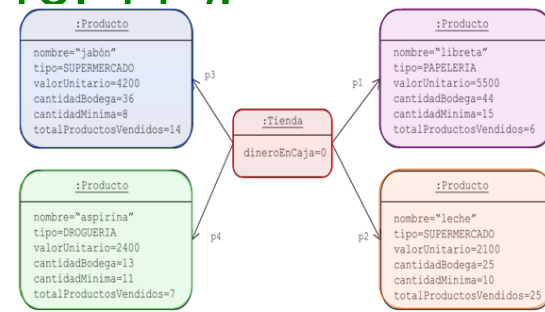
```
    Producto x = new Producto( Producto.PAPELERIA,  
                                "libreta", 5500, 44, 15 );
```

```
    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 Tienda con los parámetros en el ORDEN correcto

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 de la tienda que se está creando

Atributos de la tienda 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;

}

