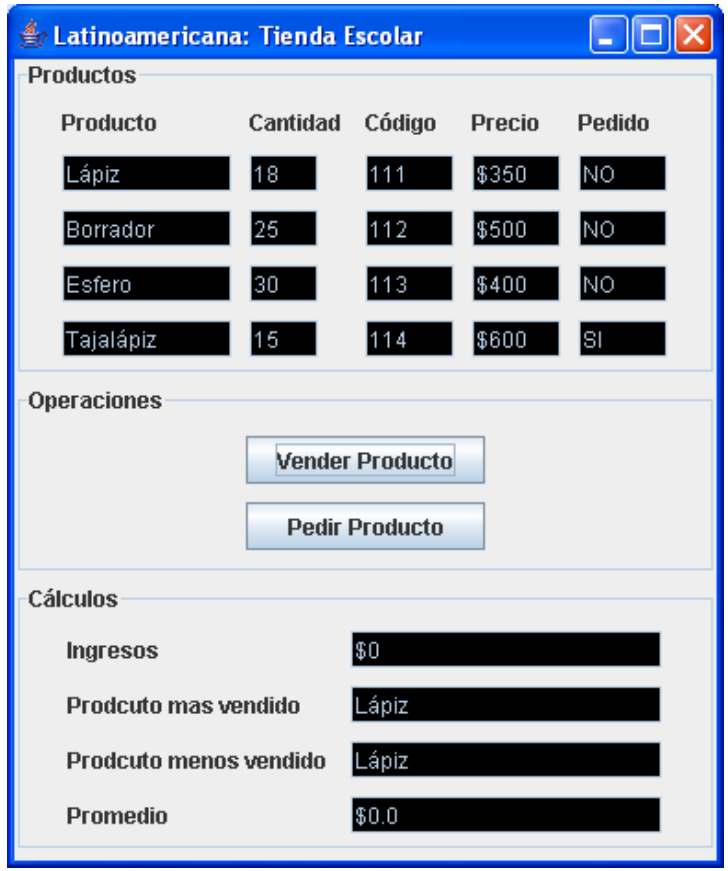
Nivel 2:
Definiendo Situaciones y Manejando Casos

Constantes, Expresiones y Operadores

Enunciado: Ejemplo Tienda

En este caso, se tiene una pequeña tienda que vende cuatro productos, para cada uno de los cuales se debe tener (1) su nombre, (2) su tipo (puede ser un producto de papelería, supermercado o droguería), (3) la cantidad actual del producto en la tienda (# de unidades disponibles en bodega), (4) el # de productos por debajo del cual se debe hacer un nuevo pedido al proveedor y (5) el precio base de venta por unidad.



Para calcular el precio final de cada producto se deben sumar los impuestos que define la ley. Dichos impuestos dependen del tipo de producto: uno de papelería tiene un IVA del 16%, uno de supermercado del 4% y uno de droguería del 12%.

El programa de la tienda debe permitir: (1) vender a un cliente un cierto número de unidades de un producto, (2) hacer un pedido de un producto para el cual ya se llegó al tope mínimo definido y (3) mostrar algunas estadísticas de la tienda, que son (a) el producto más vendido, (b) el producto menos vendido, (c) las ventas totales, y (d) el promedio de ventas de la tienda.

Requerimientos Funcionales Tienda

Nombre	R1 - Vender un producto
Resumen	Vender a un cliente un cierto número de unidades de un producto
Entradas	
Nombre	**Descripción**
Nombre del producto	Nombre del producto a vender
Cantidad a vender	Cantidad de productos a vender
Resultados	
Nombre	**Descripción**
Resultado	Si había suficiente cantidad se vende la cantidad solicitada, si no, la total en la tienda. Se guarda el dinero en la caja y se le informa al cliente la cantidad vendida

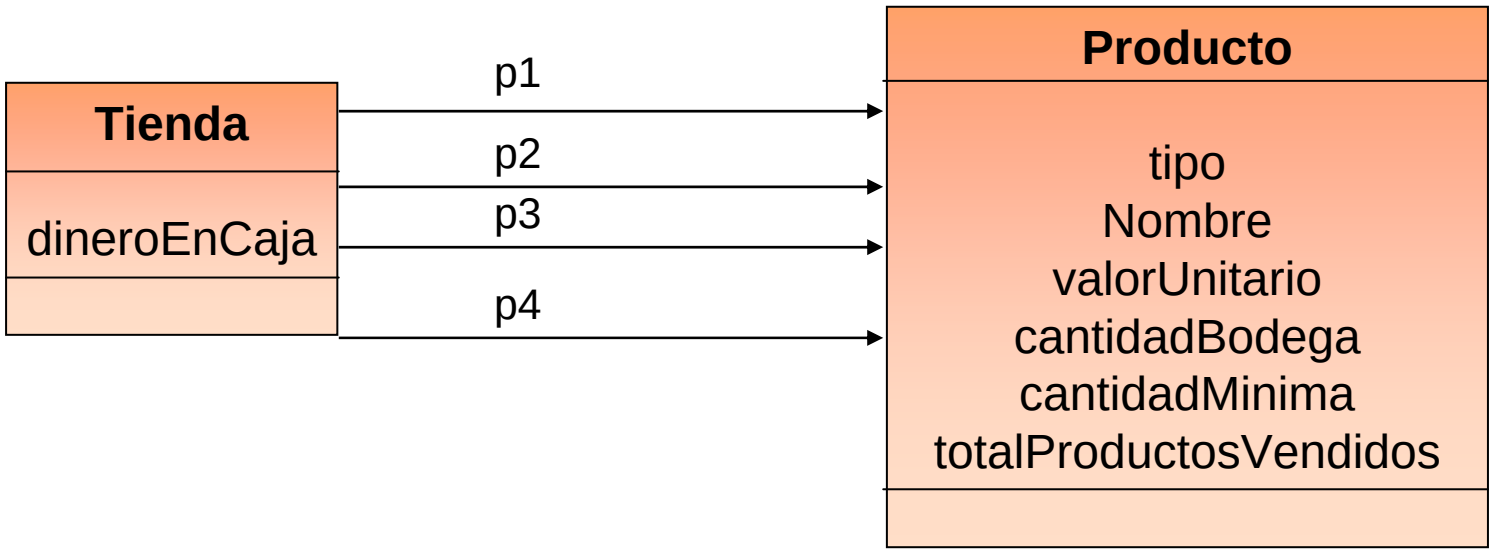
Requerimientos Funcionales Tienda

Nombre	R2 - Realizar un pedido del producto
Resumen	Aumenta la cantidad en bodega de un producto para el cual ya se llegó al tope mínimo definido
Entradas	
Nombre	**Descripción**
Nombre del producto	Nombre del producto a pedir
Cantidad a vender	Cantidad de productos del código dado a pedir
Resultados	
Nombre	**Descripción**
Resultado	Si se pudo realizar el pedido, o no. Aumenta la cantidad en bodega del producto

Requerimientos Funcionales Tienda

Nombre	R3 – Calcular Estadísticas de Ventas
Resumen	Mostrar las estadísticas de (a) el producto más vendido, (b) el producto menos vendido, (c) las ventas totales, y (d) el promedio de ventas de la tienda
Entradas	
Nombre	**Descripción**
Ninguna	
Resultados	
Nombre	**Descripción**
Resultado	Se presentan las estadísticas por pantalla

Modelo Básico del Mundo Tienda



Tipos de Datos

Tipo en Java	Significado	Ejemplo de Literales
`int`	Número entero	15 - 402
`double`	Número real	3.2 - 0.56
`String`	Cadena de caracteres	“la casa es verde” “ “ “”
`boolean`	Valores lógicos: verdadero o falso	`true` `false`
`char`	Caracter	‘a’ ‘A’ ‘*’

Tipos de Datos - *Ejemplos*

`boolean`

Declaración

```
public class Producto
{     ...
      // Característica que determina si un producto es perecedero o no
      private boolean perecedero;
}
```

Asignación

```
perecedero = true;
perecedero = false;

boolean valorLogico = perecedero;    // en otra variable
```

Tipos de Datos - *Ejemplos*

char

Declaración

```
public class Producto
{     ...
  // Característica que define si la calidad de un producto es A, B o C
        private char calidad;
}
```

Asignación

```
calidad = 'A';
calidad = 'C';
calidad = 67;                    // código UNICODE para la C

char letra = calidad;    // en otra variable
```

Constantes

Se usan para representar valores inmutables (constantes) que no cambiarán durante la ejecución de un programa o para definir el dominio de un atributo

Declaración

```
public class Producto
    { …
    //-------------------------------------------------------------------
    // Constantes
    //-------------------------------------------------------------------
    private final static double IVA_PAPEL = 0.16;
    private final static double IVA_FARMACIA = 0.12;  // inmutables…

    // Valores posibles del atributo de tipo producto
    private final static int PAPELERIA = 1;
    private final static int SUPERMERCADO = 2;
    private final static int DROGUERIA = 3;
    }
```

Identificarlas antes de los atributos

Nombres en mayúscula separados por _

Constantes

Asignación

Dentro de métodos de la clase Producto:

- tipo = PAPELERIA;
- precio = valorUnitario * (1 + IVA_FARMACIA);

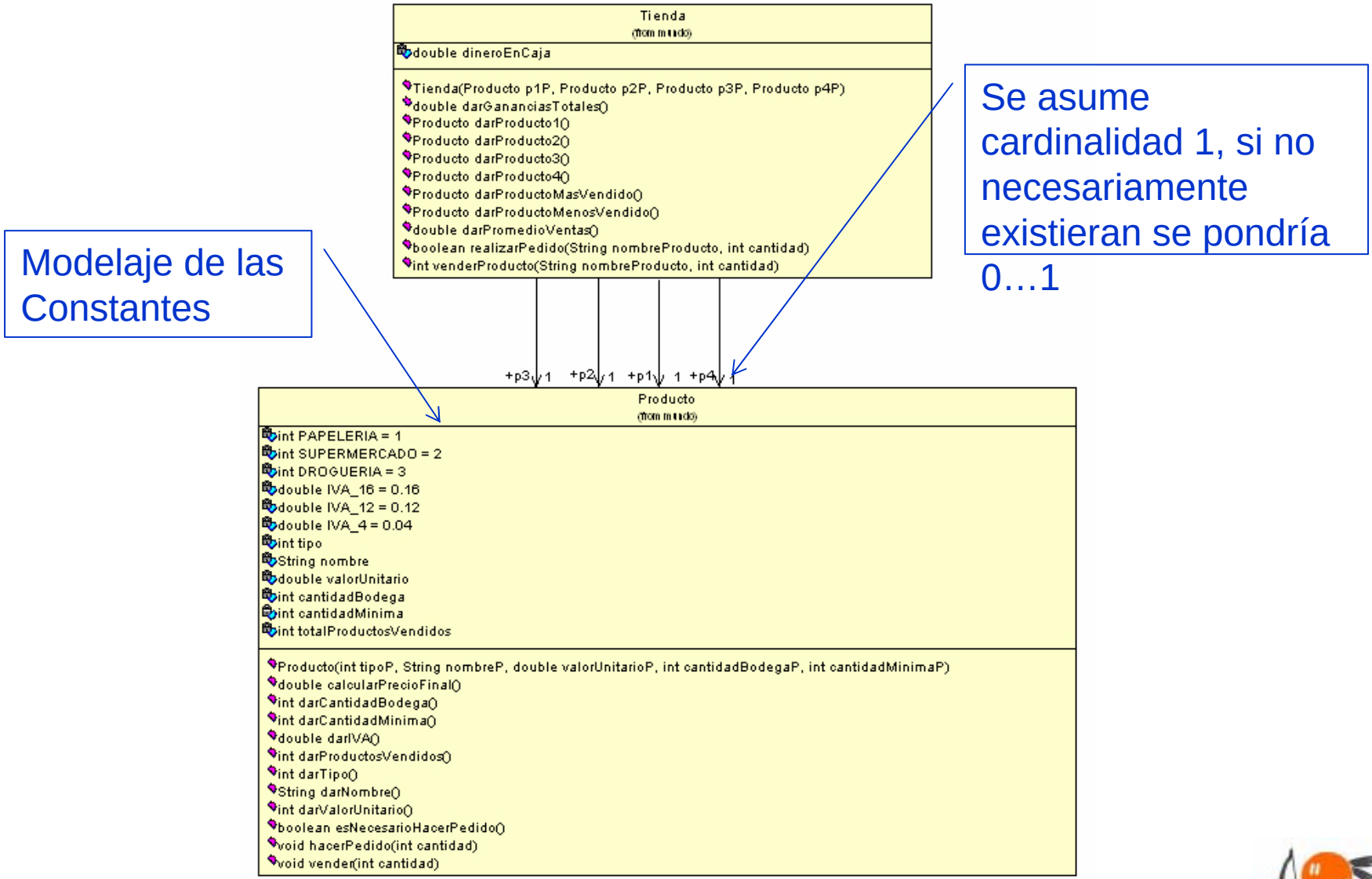
Llamado desde otra Clase

Dentro de un método de la clase Tienda:

tipoVenta = Producto.PAPELERIA

Indicar la clase donde fueron creadas precedida de un punto (.) y el nombre de la constante

Modelo del Mundo Tienda



Expresiones

Forma como expresamos algo sobre el estado de un objeto. Operadores + Operandos

Ejemplos

- valorUnitario * 3
- cantidadBodega – cantidadMinima
- valorUnitario * (1 + IVA_FARMACIA / 2);

Operador módulo o residuo (%)

Calcula el residuo (lo que “sobra”) de una división

- 4 % 2 → Devuelve 0
- 4 % 3 → Devuelve 1
- 4.6 % 2.1 → Devuelve 0.4

Operadores

Operadores de conversión o *Castings*

- Tomar el valor entero de un atributo de tipo double
  (i.e. **private double** valorUnitario):

```
int valorPesos = ( int ) valorUnitario;
```

La división entre enteros da un valor entero si no se hace un *casting*

- Obtener la división con decimales de dos enteros: (i.e. **private int** ventasTotales; **private int** numProductos;):

```
double promedioVentas = ( double ) ventasTotales / numProductos;
```

Operadores

Operadores Relacionales

Establecen un valor de verdad frente a una situación del mundo:

Operador	Significado	Ejemplo
= =	Es igual que	valorUnitario = 55.75
! =	Es diferente de	tipo ! = PAPELERIA
>	Es mayor que	cantidadBodega > cantidadMinima
<	Es menor que	cantidadBodega < 200
> =	Es mayor o igual que	valorUnitario > = 100.0
< =	Es menor o igual que	valorUnitario < = 100.0

Operadores

Operadores Lógicos

- Operando1 **&&** Operando2 : y. Es verdadero si los dos operandos son verdaderos

  (valorUnitario > = 10000) **&&** (valorUnitario < = 20000)

  (tipo = = SUPERMERCADO) **&&** (totalProductosVendidos == 0)

- Operando1 **||** Operando2 : o. Es verdadero si alguno de los dos operandos son verdaderos

  (tipo = = SUPERMERCADO) **||** (tipo = = DROGUERIA)

  (cantidadBodega > cantidadMinima) **||** (cantidadBodega > 1000)

- Operando1 **!** : no. Es verdadero si el operando es falso

  ! (tipo == SUPERMERCADO) *equivale a* (tipo ! = SUPERMERCADO)

Operadores

Operadores de Asignación Adicionales

Operador	Significado	Ejemplo

+ +	Incrementa en uno un atributo entero	*Si el atributo* cantidadBodega *tiene el valor 78, con la instrucción:*
		cantidadBodega ++;
		Quedará en cantidadBodega *el valor 79, que es equivalente a hacer:*
		cantidadBodega = cantidadBodega + 1;

- -	Decrementa en uno un atributo entero	cantidadBodega --; *equivale a* cantidadBodega = cantidadBodega – 1;
+ =	Incrementa un atributo en algún valor	valorUnitario + = 1000 *Incrementa el atributo* valorUnitario *en 1000, que es equivalente a hacer:* valorUnitario = valorUnitario + 1000;
- =	Decrementa un atributo en algún valor	*Si* valorUnitario *tiene el valor 400:* valorUnitario - = 100 *deja* valorUnitario *con el valor 300*

Operadores

Operadores sobre Cadenas de Caracteres

String cad1 = “Mi mamá me mima”
String cad2 = “ y me ama “

} *Declaración y asignación*

El tipo `String` no es un tipo simple sino una clase de Java que debe invocarse con sus propios métodos. 3 básicos:

Operadores

- Concatenación (+): Une dos cadenas de caracteres en una sola

```
String cad1 = “Mi mamá me mima”
String cad2 = “ y me ama “

String cad3 = cad1 + cad2
```

Deja en cad3 “Mi mamá me mima y me ama”

```
public String mifrase ( )
{
      return cad1 + 3 + “veces”
}
```

Retorna “Mi mamá me mima 3 veces”.

Note que no se debe convertir el 3 a una cadena

Operadores

- **Comparación (`equals`):** Compara dos cadenas de caracteres. Retorna un valor de tipo `boolean`

  En la clase Producto este método compara la cadena que recibe como parámetro con el atributo *nombre:*

```
public boolean esIgual ( String buscado )
{
    return nombre.equals ( buscado);
}
```

`equalsIgnoreCase` compara sin importar si son mayúsculas o minúsculas

Operadores

- **Extracción de un carácter (`charAt`):** Obtiene un carácter en la posición de una cadena. La primera posición es la cero (0)

String cad1 = “Mi mamá me mima”

cad1.**charAt** (1) → Devuelve ‘ i ‘

cad1.**charAt** (4) → Devuelve ‘ m ‘