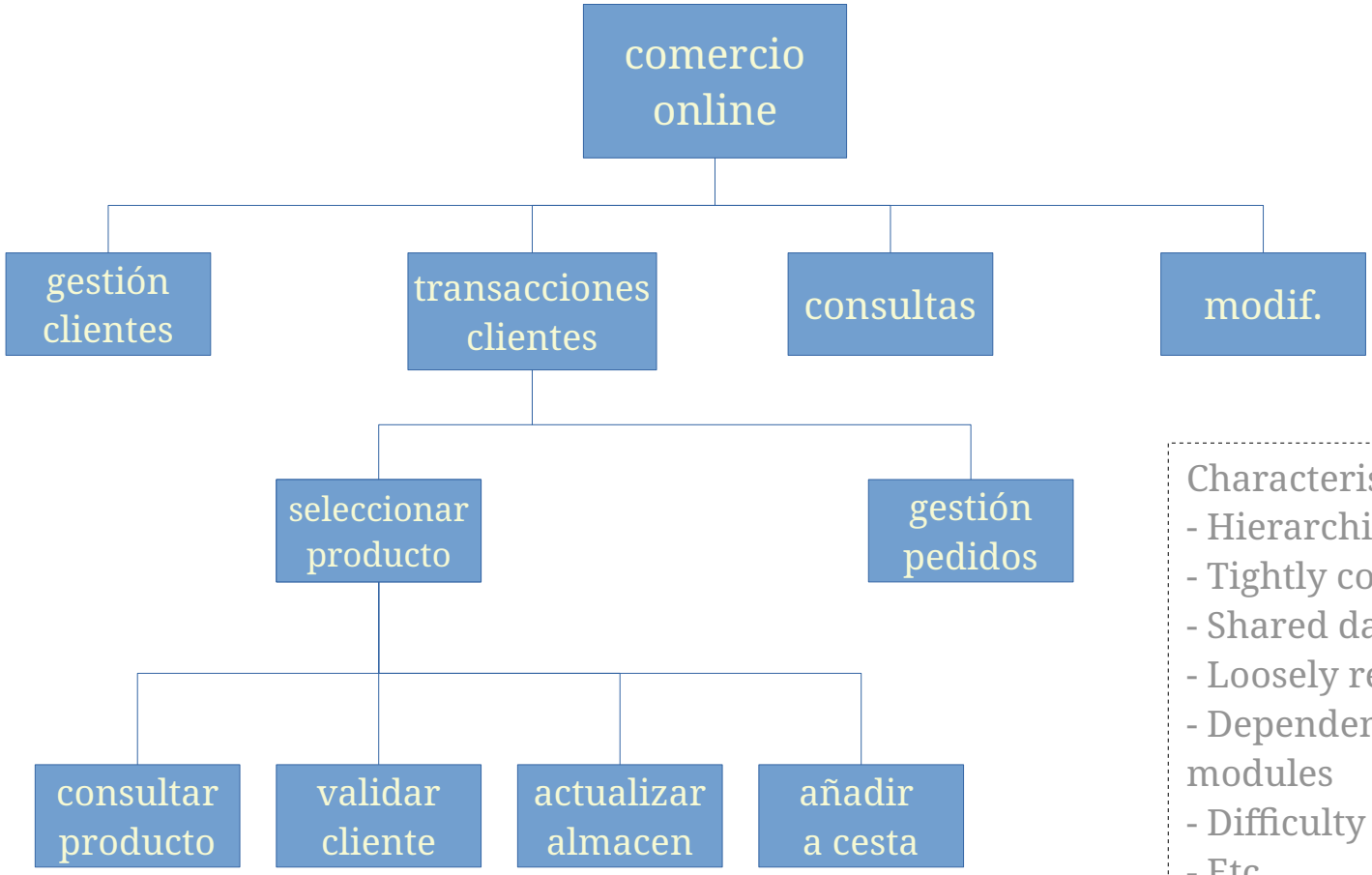
TEMA 4:
TDD vs OOD. ESPECIFICACIÓN E IMPLEMENTACIÓN DE TAD. EJEMPLOS

Descomposición Top-Down Design (TDD) vs Object Oriented (OOD)

- Descomposición funcional **vs** orientada a objetos
- Descomposición arriba-abajo (top-down design, TDD) **vs** orientada a objetos (object oriented design, OOD)
- Ejemplo: aplicación de comercio online

Ejemplo comercio online - TDD



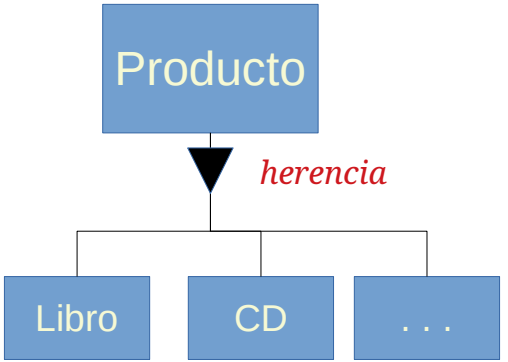
Characteristics:
- Hierarchic
- Tightly coupled
- Shared data structures
- Loosely reusable
- Dependency between modules
- Difficulty working in a team
- Etc...

Ejemplo comercio online - OOD

Clases:

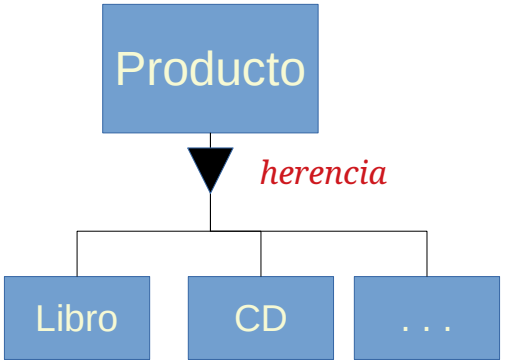
Ejemplo comercio online - OOD

```
Algoritmo  Seleccionar Producto (Cliente cliente)

Producto producto

Almacen almacen

ListaProductos lp

if (lp = almacen.filtar("Memoria USB")) then

  lp.show() #muestra lista al usuario

  id = lp.selected() # usuario selecciona producto
                     # de la lista.

  producto=almacen.get(id)

  producto.show() # muestra detalles del producto
                  # al usuario.

if (producto.selected() AND cliente.ok()) then

  cliente.cesta.insert(producto)

  almacen.reserva(producto, 1) # reserva 1 unidad del
                               # producto.

Fin
```



Ejemplo comercio online - OOD

```
Algoritmo  Gestión Pedido (Cliente cliente)
Producto producto
Almacen almacen
ListaProductos lp
GestorPago gp
lp=cliente.cesta.getProductos()
for i in lp:
  cantidad = cliente.cesta.getCantidad(i)
  almacen.salida(i, cantidad, cliente.getId())
gp.iniciaPago(cliente)
gp.confirmaPago(cliente)
Fin
```



Programación con TAD (con abstracción)

Ejemplo: el fichero estadístico

123 -5609 -7889 33 -1
674 8998 456434 2 334 -
234 98 6754 -456721
1098 789 66 5437 98
199 -590643 8667 67
899 1009 234 34 134 45
67 5634 -25 467 4674 2
45 34567 -89 90 56 78
456 2 341 34 -2354 -567
567 867 890 54 83 562
546 34567 546 73 -67
45678 -56 783 567 -356
78.....

?

Ejemplo: el fichero estadístico

```
Algoritmo estadistica(fichero f)
Tabla t
mientras no fin(f) hacer
   t.añadir(leer(f))
fin_mientras
para i de 1 a t.total() hacer
    x=t.infonum(i)
    frecx=t.infofrec(x)
    escribir("entero ",x, "frecuencia =", frecx)
  fin_para
Fin
```

Especificación de TAD

TAD Tabla

DESCRIPCIÓN

Gestiona un conjunto de enteros y sus estadísticas

OPERACIONES

- **PROC** añade(entero i) **DEV** ()
  REQUIERE: True
  MODIFICA: Ø
  EFECTOS: incrementa la frecuencia del entero "*i*" en una unidad.

- **PROC** total() **DEV** (entero n)
  REQUIERE: True
  MODIFICA : Ø
  EFECTOS: devuelve en "n" el número de enteros distintos en la tabla.

Especificación de TAD

TAD Tabla

DESCRIPCIÓN

Gestiona un conjunto de enteros y sus estadísticas

OPERACIONES

- **PROC** infonum(entero i) **DEV** (entero x)

  REQUIERE: True

  MODIFICA: Ø

  EFECTOS: devuelve el entero “x” que ocupa la posición i-ésima. Si la posición “i-ésima” no existe, lanza la excepción FUERA_DE_RANGO

- **PROC** infofrec(entero x) **DEV** (entero f)

  REQUIERE: True

  MODIFICA : Ø

  EFECTOS: devuelve en “f” el número de veces que se repite “x” en la tabla.

Especificación de TAD

La especificación informa de manera precisa de los detalles relevantes del TAD:

- **RELEVANTE:** ¿qué? (elementos subrayados)
- **IRRELEVANTE:** ¿cómo?

TAD TAD's name
DESCRIPTION short description of the TAD

OPERATIONS

PROC name (parameters) **DEV** (return value)
REQUIRES: condition
MODIFIES: list of modified parameters
EFFECTS: short description of procedure effects

. . .

Especificación de TAD

Más especificaciones de procedimientos: breves, concisas, precisas.

- **PROC** concat(string a, string b) **DEV** (string ab)

  REQUIRES: True

  MODIFIES: Ø

  EFFECTS: “ab” results in characters in “a” (same order than in “a”) followed by characters in “b” (same order than in ”b”)

- **PROC** deleteDuplicate(int a[]) **DEV** ()

  REQUIRES: True

  MODIFIES: a

  EFFECTS: deletes duplicate elements in “a” leaving the first one in his original position. Example: if a=[3, 13, 3, 6] before the call, the resultin a will be a=[3, 13, 6].

Excepciones

Más especificaciones de procedimientos: breves, concisas, precisas.

- **PROC** binarySearch(int a[], int x) **DEV** (int i) **EXCEPTIONS** NotFoundException

  REQUIRES: “a” ordered ascending

  MODIFIES: Ø

  EFFECTS: if “x” is not in “a”, the procedure **throw** the exception “NotFoundException”.
  If “x” is in “a”, then “i” is such that a[i]=x.

- **PROC** factorial(int n) **DEV** (int i) **EXCEPTIONS** NotPositiveException

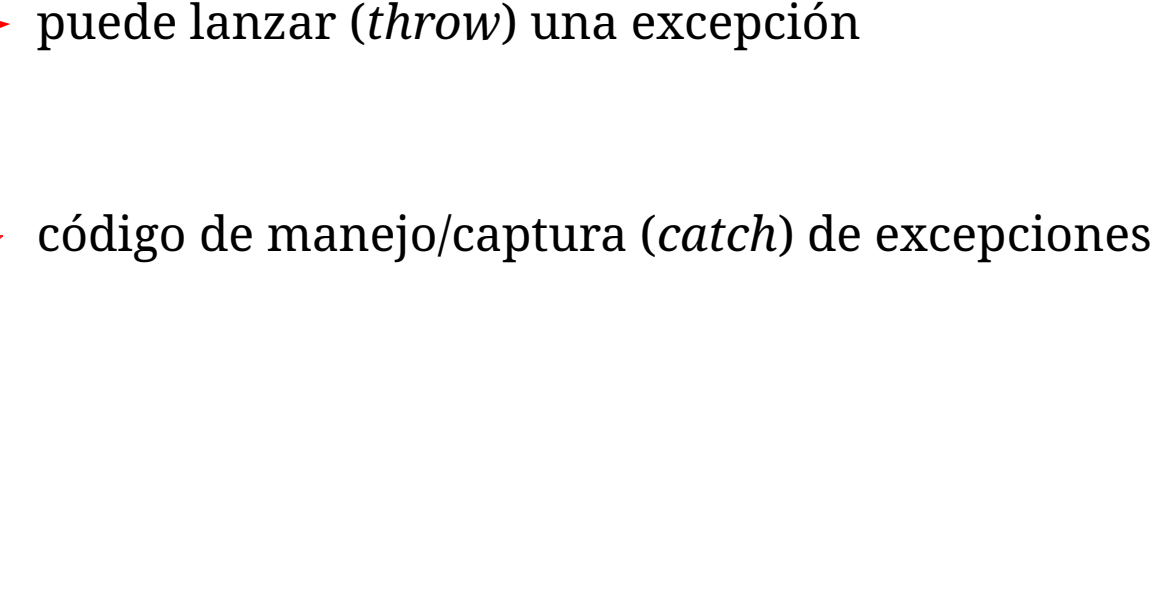
  REQUIRES: True

  MODIFIES: Ø

  EFFECTS: if “n” is not positive, the procedure **throw** the exception “NotPositiveException”.
  If “n” is positive, then “i” is the factorial of “n”.

Excepciones

```
try {
    proc(i,j)
    }
catch(A)
    {
    . . .
    }
catch(B)
    {
    . . .
    }
catch(C)
    {
    . . .
    }
```



Implementación de TAD

- Requisitos de la implementación:
  - Pequeña (la menor/más-simple que cumpla la especificación)
  - cerrada/abierta
  - Etc. (criterios, reglas y principios de descomposición modular de *Bertrand Meyer*)
- STUBS
  - Función/procedimiento pendiente de implementar pero *callable*
  - “estoy aquí”
  - Posteriormente se completa.
  - Prototipado rápido

Operaciones con TADS

- Constructores (A::A() en C++)
- Observadores: *getters*...
- Modificadores: *setters*...
- Destructores (~ en C++)