Laboratorio de Computación III

Clase XXII

Andrés Fortier

Problema

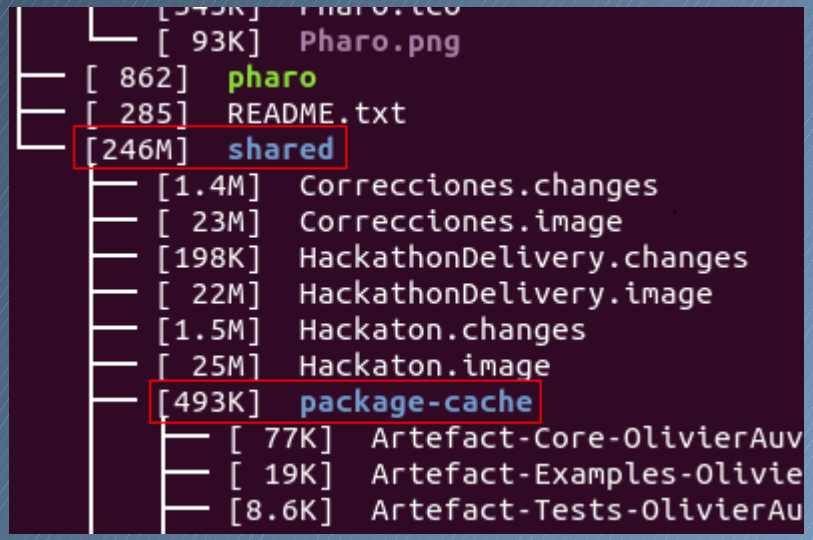
- Queremos modelar un sistema de archivos
- Dos tipos de componentes
  - Archivos. Tienen un nombre y un tamaño en bytes.
  - Directorios. Tienen un nombre y su tamaño es la suma del tamaño de su contenido + 4096 bytes.
- Nos interesa poder calcular el tamaño de cualquier directorio.

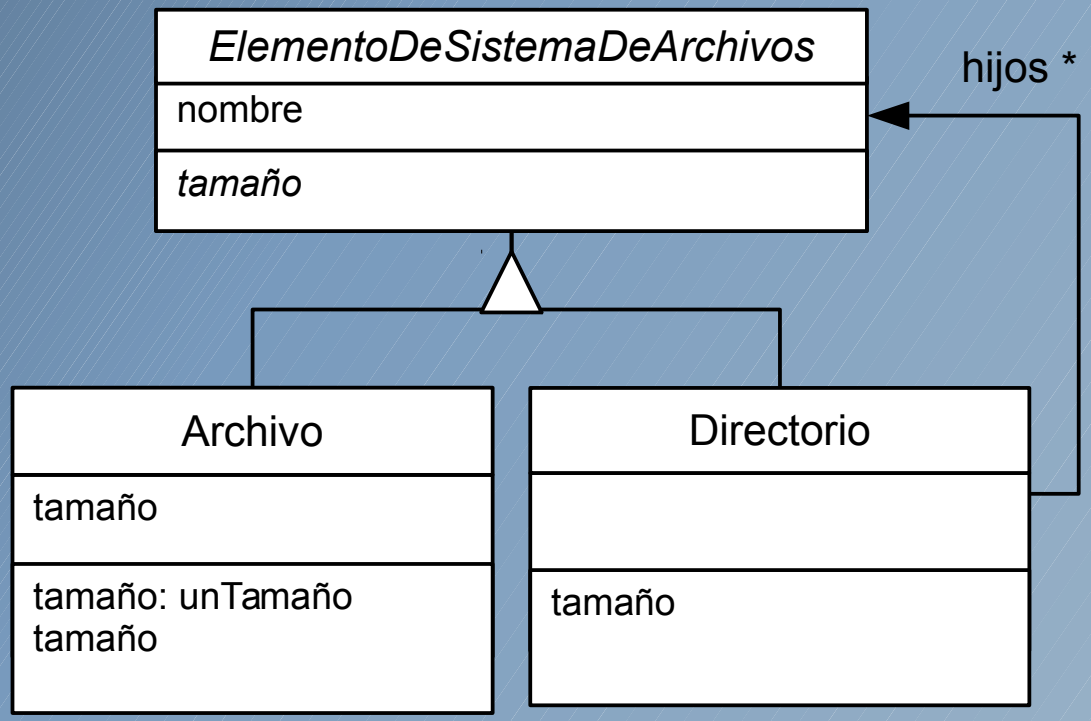
Ejemplo

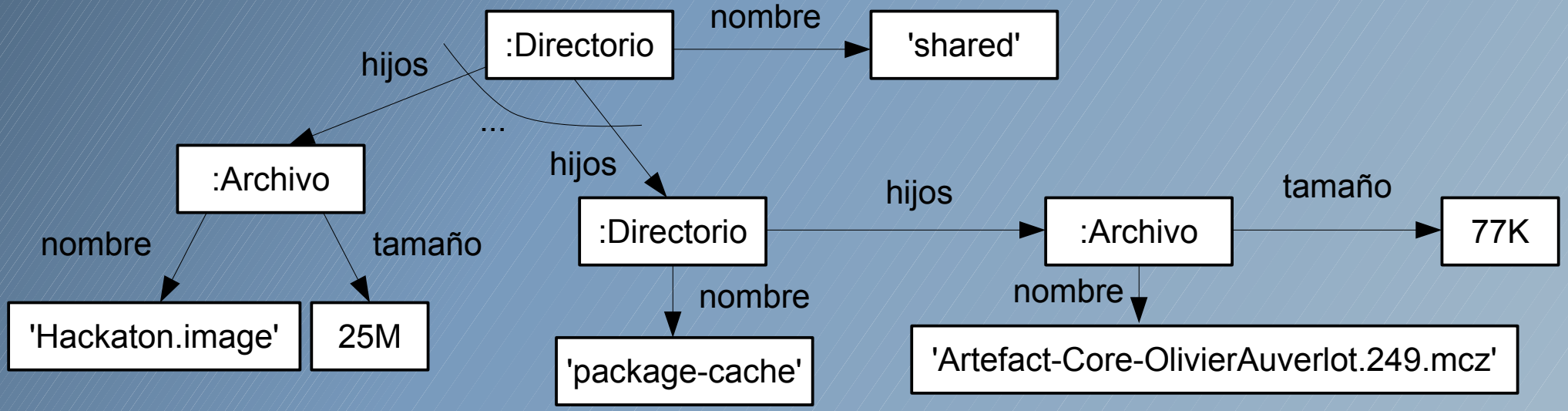
- Notar que los directorios se pueden anidar

```
    └── [ 93K]  Pharo.png
├── [ 862]  pharo
├── [ 285]  README.txt
└── [246M]  shared
    ├── [1.4M]  Correcciones.changes
    ├── [ 23M]  Correcciones.image
    ├── [198K]  HackathonDelivery.changes
    ├── [ 22M]  HackathonDelivery.image
    ├── [1.5M]  Hackaton.changes
    ├── [ 25M]  Hackaton.image
    ├── [493K]  package-cache
    │   ├── [ 77K]  Artefact-Core-OlivierAuverlot.249.mcz
    │   ├── [ 19K]  Artefact-Examples-OlivierAuverlot.2.mcz
    │   ├── [8.6K]  Artefact-Tests-OlivierAuverlot.2.mcz
    │   ├── [7.4K]  ConfigurationOfArtefact-OlivierAuverlot.14.mcz
    │   ├── [5.1K]  ConfigurationOfMinimalConnectors-GuillermoPolito.24.mcz
    │   ├── [ 11K]  ConfigurationOfObjectBrowser-ClaraAllende.100.mcz
    │   ├── [ 11K]  ConfigurationOfSmallUML-ClaraAllende.88.mcz
    │   ├── [6.6K]  ConfigurationOfUnits-StephaneDucasse.12.mcz
    │   ├── [8.3K]  MinimalConnectors-ConnectableShapes-GuillermoPolito.21.mcz
```

Diseño

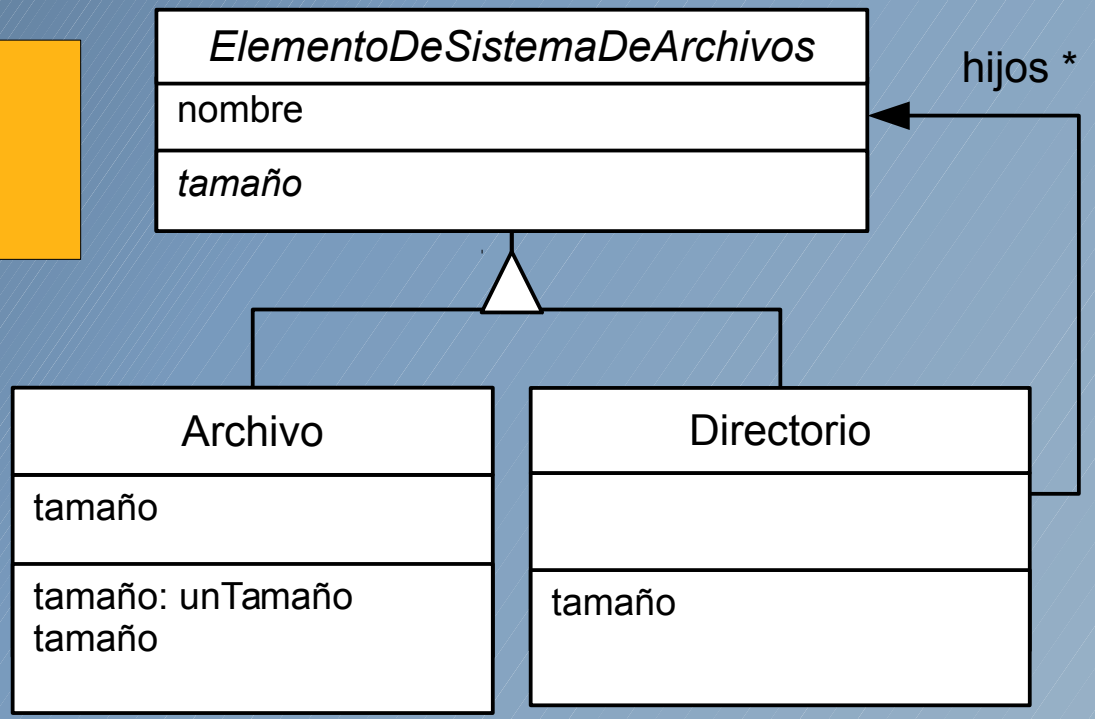

[93K] Pharo.png
[862] pharo
[285] README.txt
[246M] shared
[1.4M] Correcciones.changes
[23M] Correcciones.image
[198K] HackathonDelivery.changes
[22M] HackathonDelivery.image
[1.5M] Hackaton.changes
[25M] Hackaton.image
[493K] package-cache
[77K] Artefact-Core-OlivierAuv
[19K] Artefact-Examples-Olivie
[8.6K] Artefact-Tests-OlivierAu


ElementoDeSistemaDeArchivos
nombre
tamaño
hijos *
Archivo
tamaño
tamaño: unTamaño
tamaño
Directorio
tamaño


:Directorio
nombre
'shared'
hijos
...
:Archivo
nombre
tamaño
'Hackaton.image'
25M
hijos
:Directorio
nombre
'package-cache'
hijos
:Archivo
tamaño
77K
nombre
'Artefact-Core-OlivierAuverlot.249.mcz'

Implementación

```
ElementoDeSistemaDeArchivos>>tamaño

        ^self subclassResponsibility.
```




```
Archivo>>tamaño

        ^tamaño.
```

```
Directorio>>tamañoPropio

        ^4096.
```

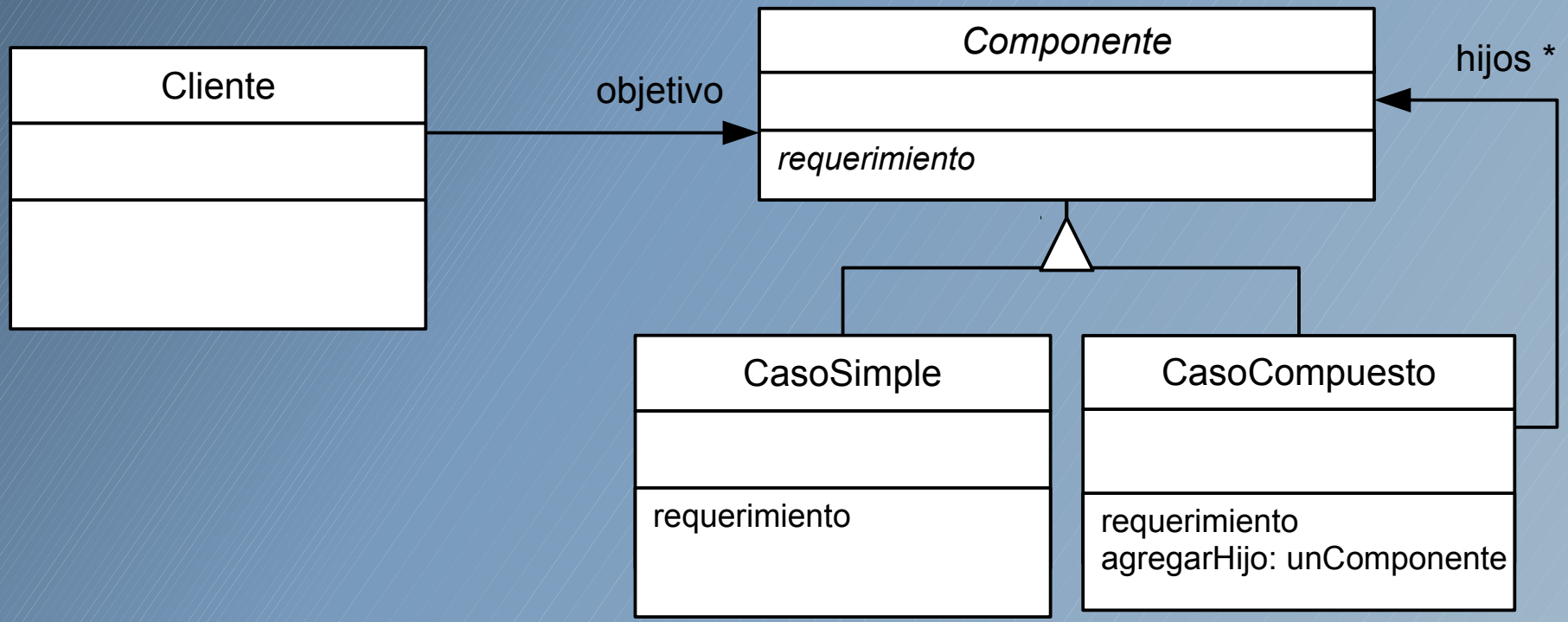
```
Directorio>>tamaño

        ^self hijos
                inject: self tamañoPropio.
                Into: [:sum :elemento | sum + elemento tamaño].
```

Tests

- Caso simple (archivo).
- Caso compuesto (directorio)
  - Vacío.
  - Un elemento.
  - N elementos.

Composite

- Objetivo
  - Combinar objetos simples para formar estructuras jerárquicas complejas.
  - Permite tratar a las partes individuales y a los objetos compuestos de manera uniforme.



Composite

- Participantes
  - Componente
    - Declara la interface pública que tienen que entender tanto los objetos simples como los compuestos.
  - CasoSimple
    - Representa los objetos terminales en la composición (no tienen hijos).
    - Definen el comportamiento del caso base.
  - CasoCompuesto
    - Maneja una colección de hijos e implementa la delegación del comportamiento a sus hijos.
  - Cliente
    - Crea y utiliza la estructura compuesta.

Composite

- Consecuencias
  - Define jerarquías de clases que constan de objetos simples y compuestos.
  - Los objetos se pueden componer recursivamente, generando estructuras jerárquicas.
  - Hace del cliente un objeto mas simple, ya que no tiene que separar entre objetos simples y compuestos.
  - Es fácil agregar nuevos casos simples y/o compuestos.
  - No es fácil restringir la composición.

Composite

- Variantes de implementación
  - Referencias explícitas al padre.
  - Interfaz de manejo de hijos compartida.
  - Colección de hijos vs. Cantidad definida de hijos.