COMPOSITE

Xavier Vallejo López

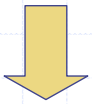


Descripció(I)

- Cada objecte es classifica en:
  - Individual
  - Compost (format per individuals o altres compostos)
- Interfície comuna per tots els tipus d'objectes.
  - Client tracta de la mateixa manera als components.

Descripció(II)

- Normalment es parla d'estructura d'arbre.



- Nodes terminals → Individuals
- Nodes no terminals →Composites

Windows/UNIX file system

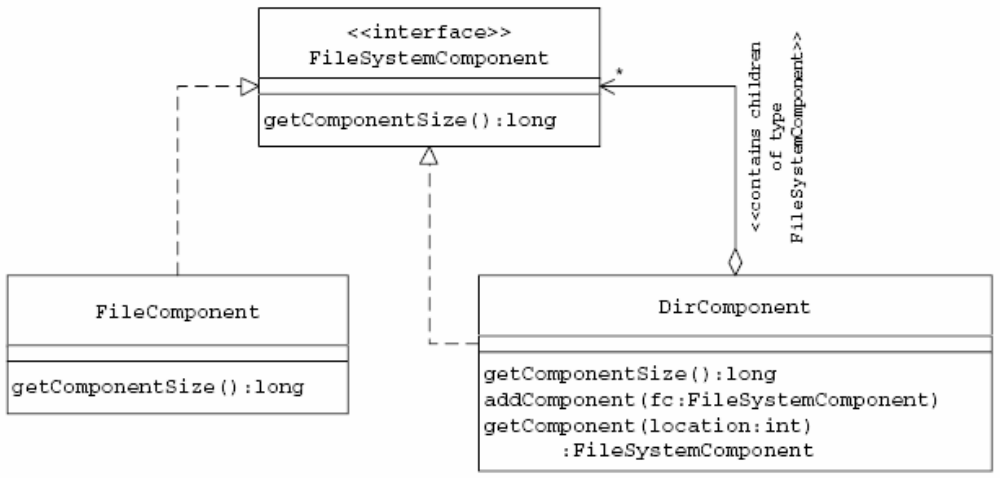
- Context:

- El sistema de fitxers està format per fitxers i directoris.
- Un directori pot contindre fitxers i altres directoris (subdirectoris).

Exemple: 1ªAproximació

- Dissenyem una **interfície** comuna per fitxers i directoris anomenada *FileSystemComponent*.
- Aquesta declara mètodes comuns pels dos *(getComponentSize()).*
- Es defineixen 2 classes més: Fitxer i Directori.
- Es poden afegir més mètodes (obert als canvis).

1r Diagrama de classes



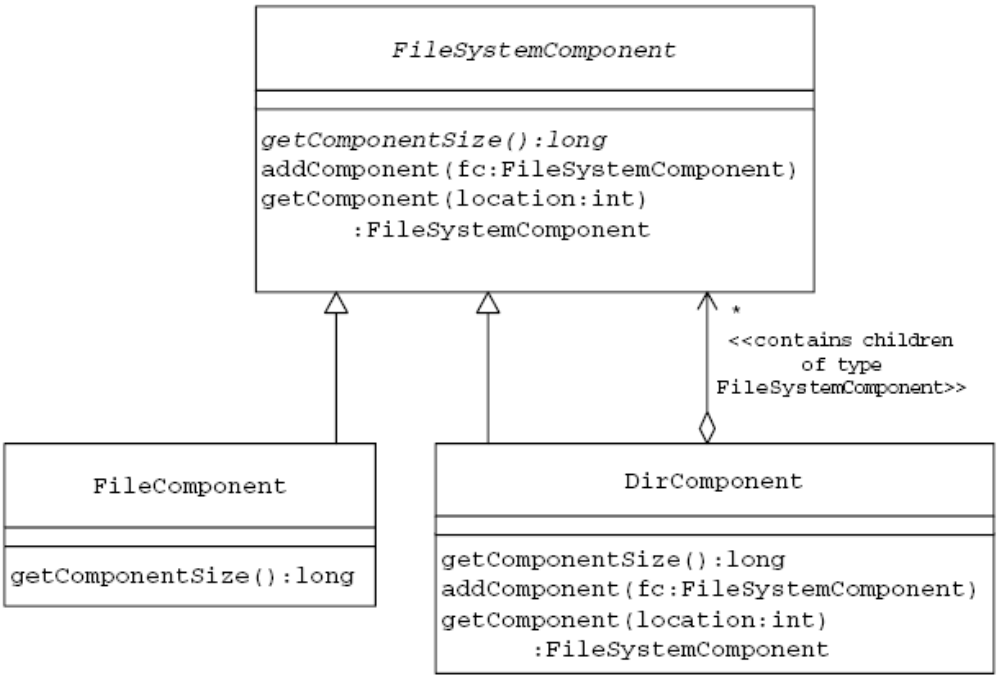
Pros i contres (I):

- (+) Client no es preocupa de la manera com es calcula el tamany d'un fitxer o un directori.
- (+) Tracta de la mateixa forma als dos.
- (-) El client ha de saber que l'objecte és un directori per tal d'aplicar els seus mètodes concrets.

Exemple: 2ªAproximació

- Defineix una **herència** en comptes d'una interfície.
- Els mètodes concrets del directori es declaren per defecte en la classe abstracte, i són definits en la concreta.
- Composite es basa en Strategy / Template?

2n Diagrama de classes



Pros i contres (II):

- (+) El client no s'ha de preocupar de quin objecte és a l'hora d'invocar els mètodes.

- (-) Si volem afegir nous mètodes (ex: *delComponent*), caldrà modificar la classe d'on hereda.