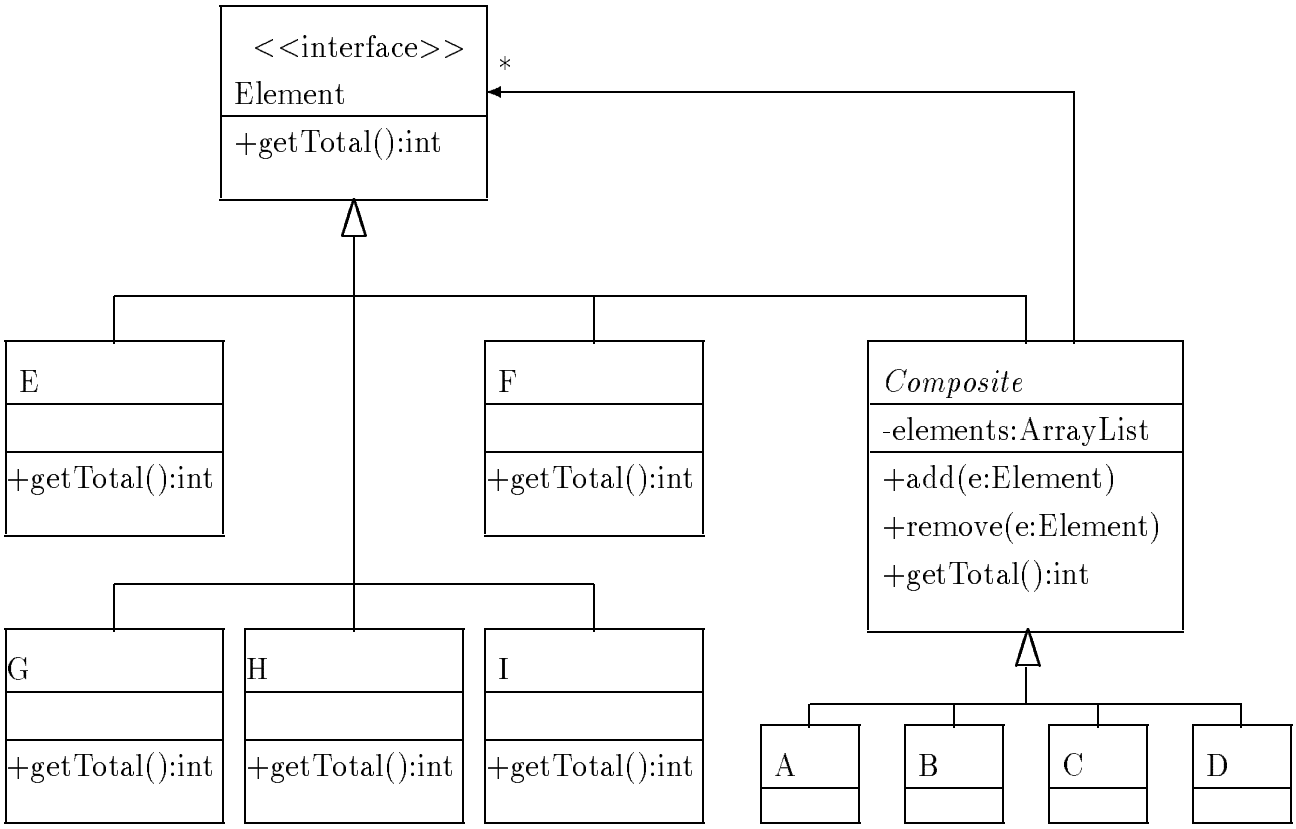
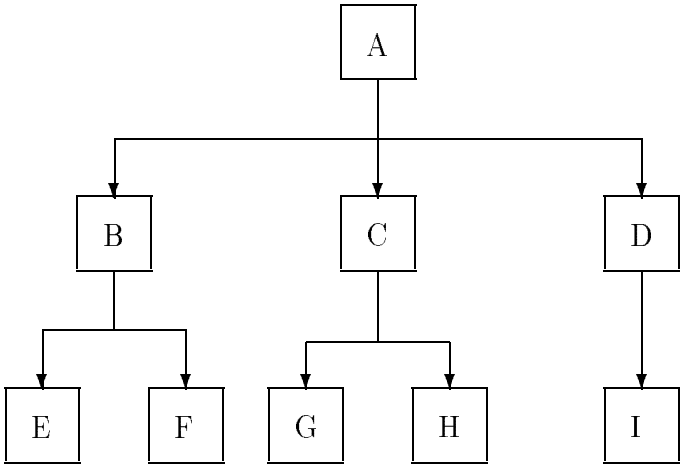
8 Exercise 7b: Composite Pattern [22]



- A..I qualify as an Element, all must have a getTotal() operation.
- A..D qualify specifically as a Composite, and automatically have a getTotal() implementation.
- Note that *Composite* is abstract.
- E..I are just Elements and must provide their own implementation of getTotal().
- Composite maintins an ArrayList, a container of many objects.
- Note that ArrayList has an arrow to MANY (*) Elements.
- add() and remove() allow Elements to be inserted/deleted from the ArrayList.
- Since A..I are Elements, the ArrayList can hold either more Composites, or just Elements.
- Therefore, Composites A..D form the *interior* nodes of a tree.
- Elements E..I form the *leaves* of a tree, and cannot have children.
- Composites getTotal() iterates through all children, and calls their getTotal().
- If the child happens to be another Composite, then the same implementation is called.
- This iteration stops at the leaves, and the net result is an accumulation of getTotal() on the tree.



- The arrow to MANY (*) Element children in the previous diagram is expanded in the above.
- Remember that A, B, C, D are Composites, and so can have many children.
- A has 3 children (B, C, D), B has 2 children (E, F), C has 2 children (G, H), D has 1 child (I).
- E..I are just Elements and cannot have children, hence they are leaves.
- But all Elements A..I have a getTotal() method.
- A..D are Composites and all inherit the same getTotal() iterator method.
- In this way, getTotal() is accumulated, starting from A (the root), throughout the entire tree.