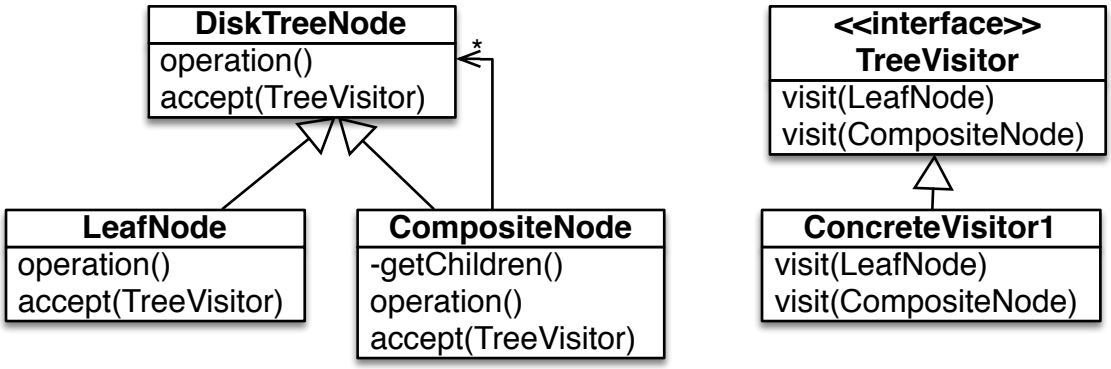
Problema 1. (5 punts)

Per raons que no venen al cas, hem implementat una estructura arborescent (utilitzant el patró *composite*) en disc.



Els enllaços entre nodes consisteixen en la posició i mida del registre associat al disc i, per obtenir els fills d'un node, cal buscar cada registre al disc (això ho fa de forma transparent l'operació privada `getChildren`). Això sí, aquesta operació **és molt més costosa** que si l'arbre estigués representat en memòria.

Per deixar oberta la possibilitat d'ampliar la funcionalitat dels arbres, s'ha implementat , usant el patró *visitor*, la possibilitat d'acceptar visitants. El mètode `accept` de `CompositeNode` es qui fa el bucle sobre els fills fent que aquests acceptin el visitant, és a dir:

```
1 public void accept(TreeVisitor visitor) {
2   for (DiskTreeNode node : this.getChildren()) {
3     node.accept(visitor);
4   }
5 }
```

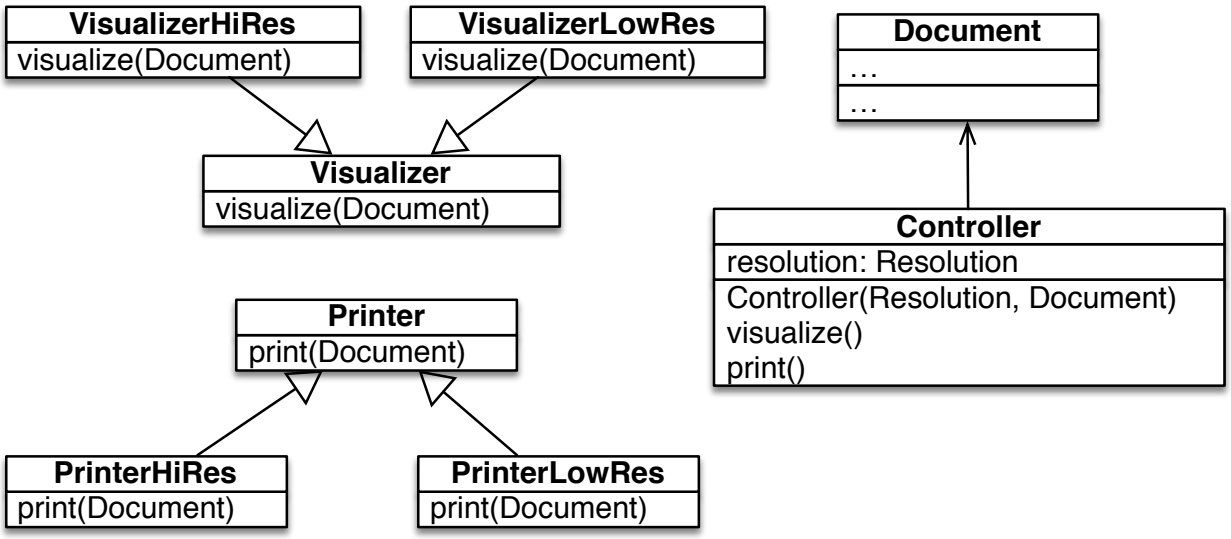
Tenim varis visitants implementats (al diagrama en mostrem un) i, fins ara, hem d'invocar el mètode `accept` sobre l'arrel de l'arbre una vegada per cadascun d'ells.

El que voldríem és, amb un sol recorregut, poder realitzar les tasques de tots els visitants, ja que així les operacions de moviment sobre l'arbre s'executarien una sola vegada.

a) Com solucionaríeu el problema? Quin patró aplicaríeu i per què?
b) Mostreu les noves classes/interfícies, tant en forma de diagrama de classes com la seva implementació en Java.

Problema 2. (5 punts)

En una aplicació ofimàtica es disposen de les següents classes que tracten sobre com mostrar i imprimir documents en diferents resolucions (no es mostren les dependències, ja que `Controller` depèn de totes les classes tant de visualització com d'impressió).



El mètode `visualize` (`print` és similar) és el següent:

```
 6 public void visualize() {
 7   Visualizer visualizer;
 8   if (this.resolution == Resolution.HI) {
 9     visualizer = new VisualizerHiRes();
10   } else { // Resolution.LOW
11     visualizer = new VisualizerLowRes();
12   }
13   visualizer.visualize(this.document);
14 }
```

On `Resolution` és un tipus enumerat amb valors possibles `HI` i `LOW`.

Al principi només es van considerar aquests dos tipus de resolucions, però amb la proliferació de tabletes de mides diferents es preveu que hi hagi molta més varietat.

Es demana:

a) Quins problemes presenta el disseny anterior respecte dels mètodes `visualize` i `print` de la classe `Controller`?
b) Com el milloraríeu? Quin patró usaríeu i per què?
c) Mostreu les noves classes/interfícies, tant en forma de diagrama de classes com de codi, i la nova implementació del mètode `visualize`.