Grenoble INP
Ensimag
Deuxième année

Analyse et Conception Objet de Logiciels

Corrections d'exercices

Table des matières

Exercice 15. Courriers électroniques : analyse et expression des besoins

1. Acteurs

L'utilisateur.

On peut à la rigueur ajouter des acteurs secondaires : serveur entrant, serveur sortant, imprimante.

2. Cas d'utilisation

Le diagramme de cas d'utilisation est représenté figure 1.

3. Diagrammes de séquence système

À compléter...

Exercice 17. Courriers électroniques : diagramme de classes d'analyse

Le diagramme de classes est représenté figure 2.

On suppose ici que les boîtes par défaut sont associées à un compte utilisateur (le cahier des charges n'est pas clair sur ce point).

Exercice 18. Courriers électroniques : architecture

On propose d'utiliser une architecture en couches, avec les couches suivantes :

- *Couche Présentation.* La couche Présentation gère l'affichage de l'application. On y trouve en particulier les différentes fenêtres, avec leur composants (menus, boutons...).
- *Couche Application.* La couche Application traite les différentes fonctionnalités du système.
- *Couche Domaine.* On trouve dans la couche Domaine la plupart des classes de la modélisation objet du domaine, structurées en différents paquetages : BoitesEtCourriers, Comptes.
- *Couche Infrastructure* La couche Infrastructure contient les éléments de bas niveau : stockage des courriers, lien avec l'imprimante, liens avec les serveurs entrants et sortants. Pour le stockage des courriers, le cahier des charges ne présise pas comment les courriers doivent être stockés. On peut imaginer soit un stockage par fichiers et répertoires pour les boîtes, soit une base de données.

La figure 3 représente l'architecture du système.

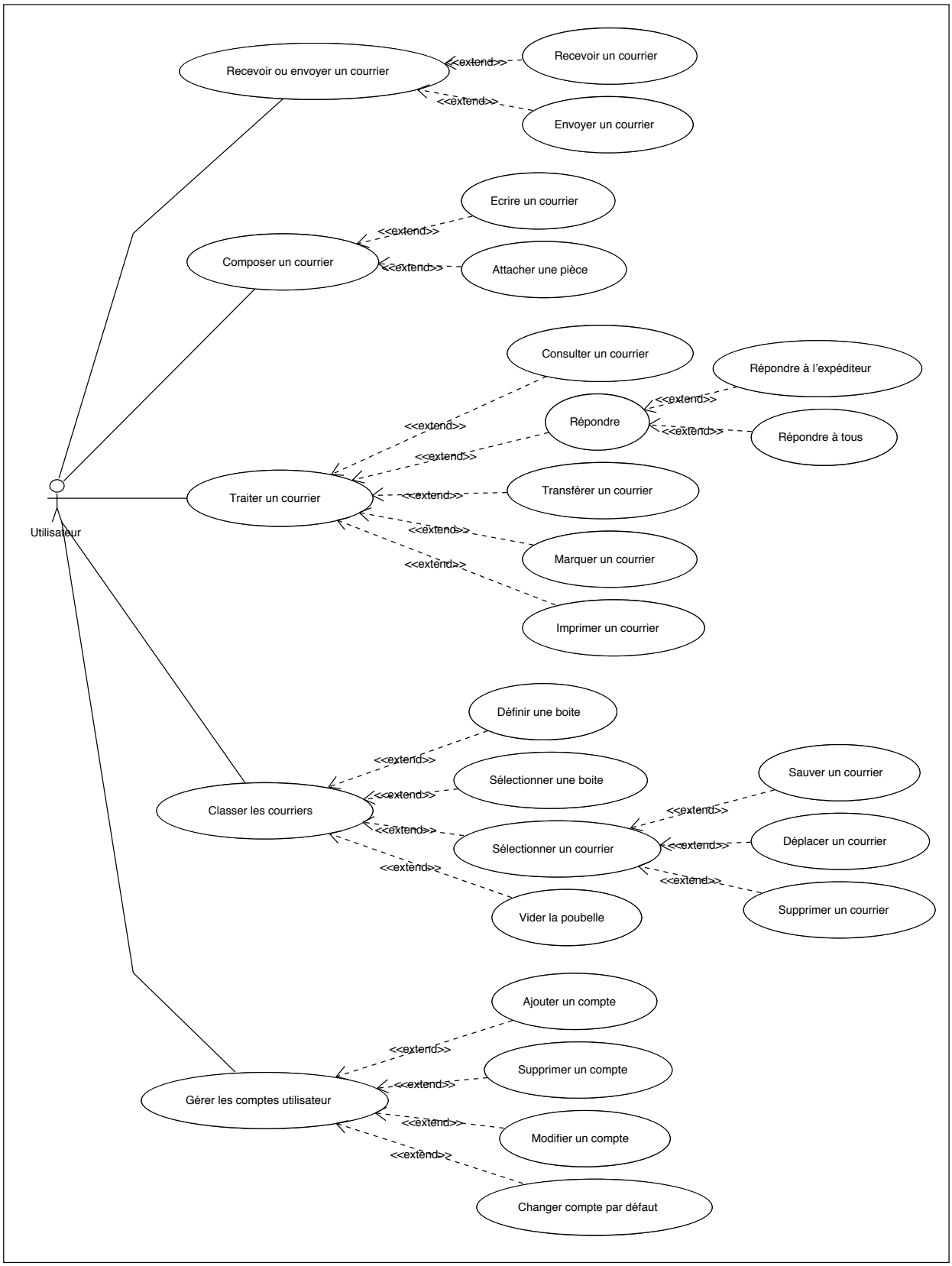

FIGURE 1 – Diagramme de cas d'utilisation

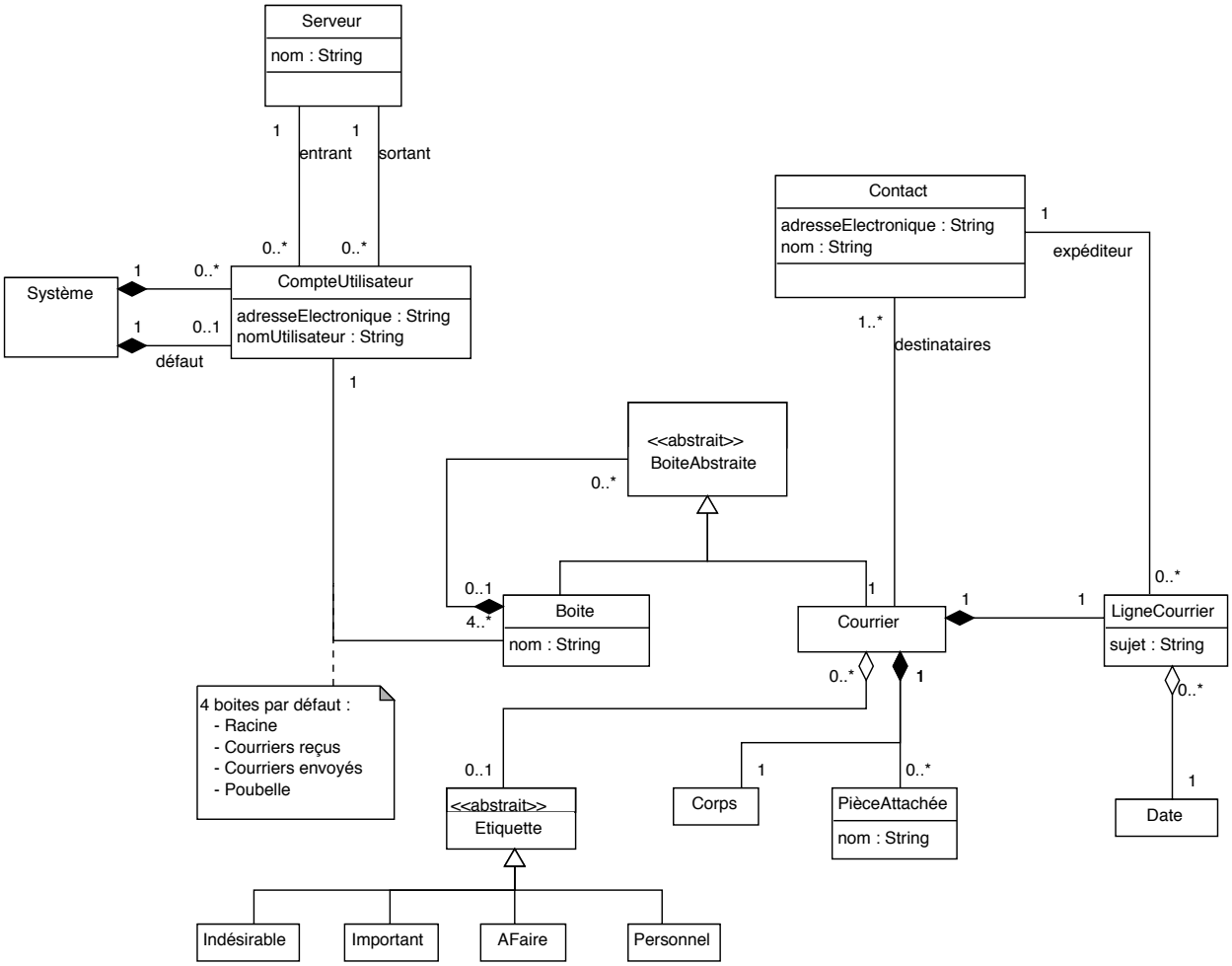

FIGURE 2 – Diagramme de classes

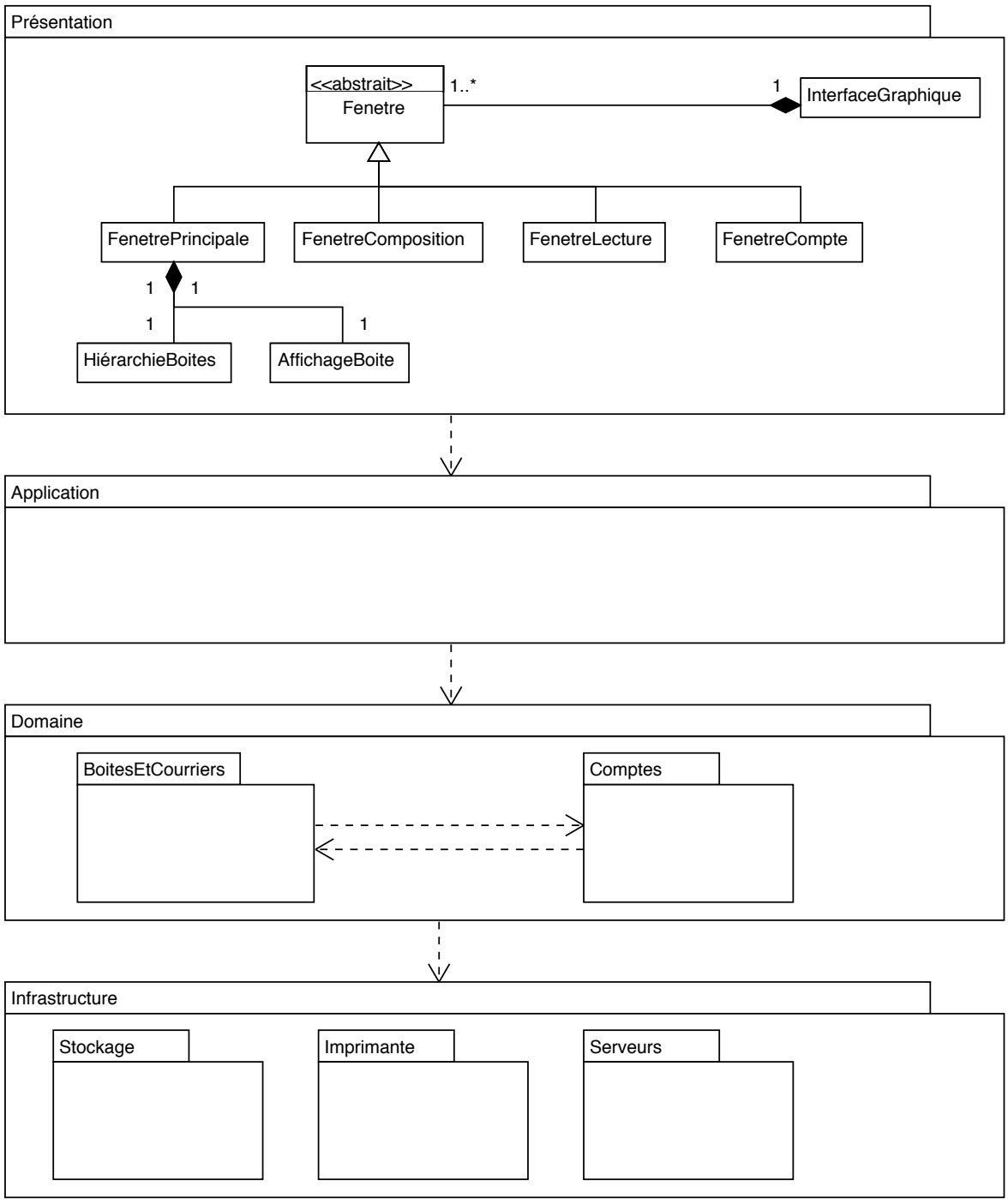

FIGURE 3 – Architecture du système

Exercice 29. Courriers électroniques : conception

1. Patron pour la hiérarchie de boîtes

Le patron utilisé pour la hiérarchie de boîtes est le patron Composite. La classe BoiteAbstraite correspond au Composant, la classe Boite correspond au composé, et la classe Courrier à un composant simple (cf. figure 4).

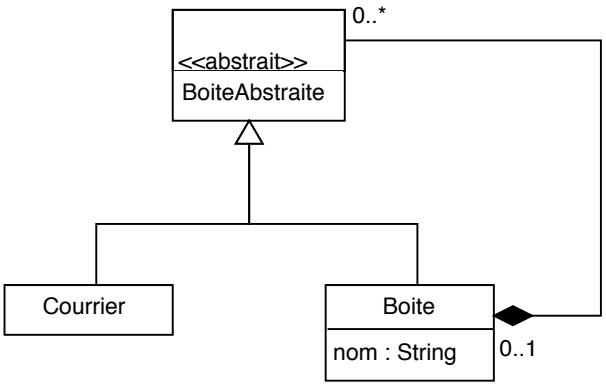

FIGURE 4 – Application du patron Composite

2. Patron pour l'affichage des boîtes

On peut utiliser le patron Observateur. Le sujet concret est la classe Boite, l'oservateur concret est la classe AffichageBoite, de la couche Présentation (cf. figure 5).

3. Patron Visiteur

- On ajoute une méthode `void applique (Visiteur v)` dans chaque classe de la hiérarchie BoiteAbstraite.
- On définit une interface Visiteur, avec une méthode `void visite(Element e)` pour chaque classe concrète Element héritée de BoiteAbstraite.
- Le codage de la recherche se fait en implémentant l'interface Visiteur.

La figure 6 montre le diagramme de classes correspondant.

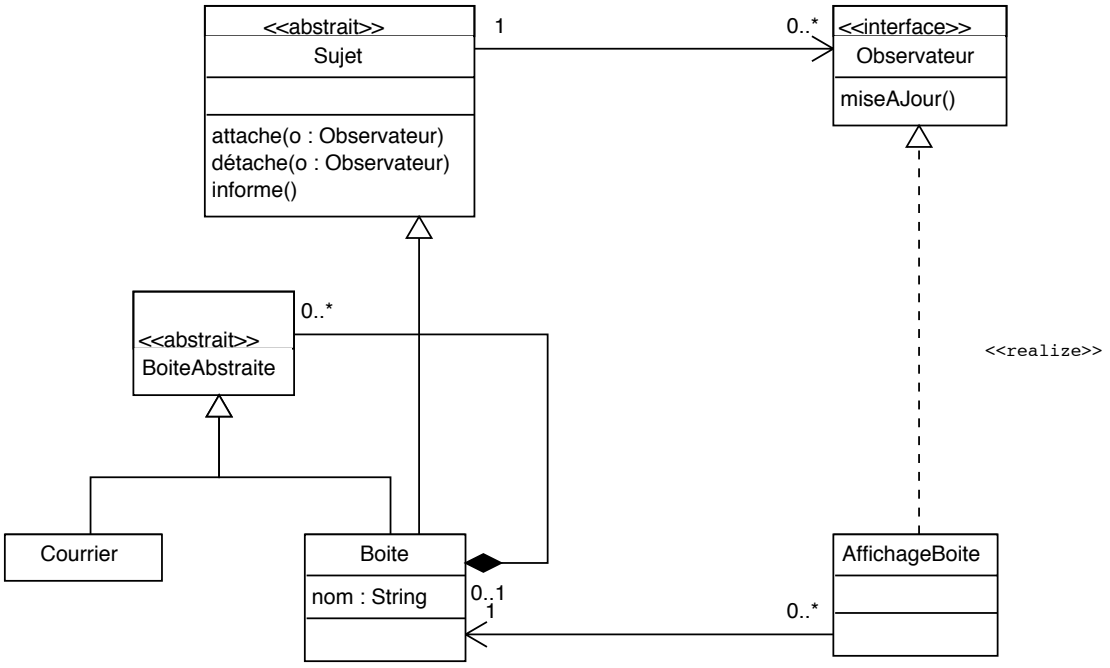

FIGURE 5 – Application du patron Observateur

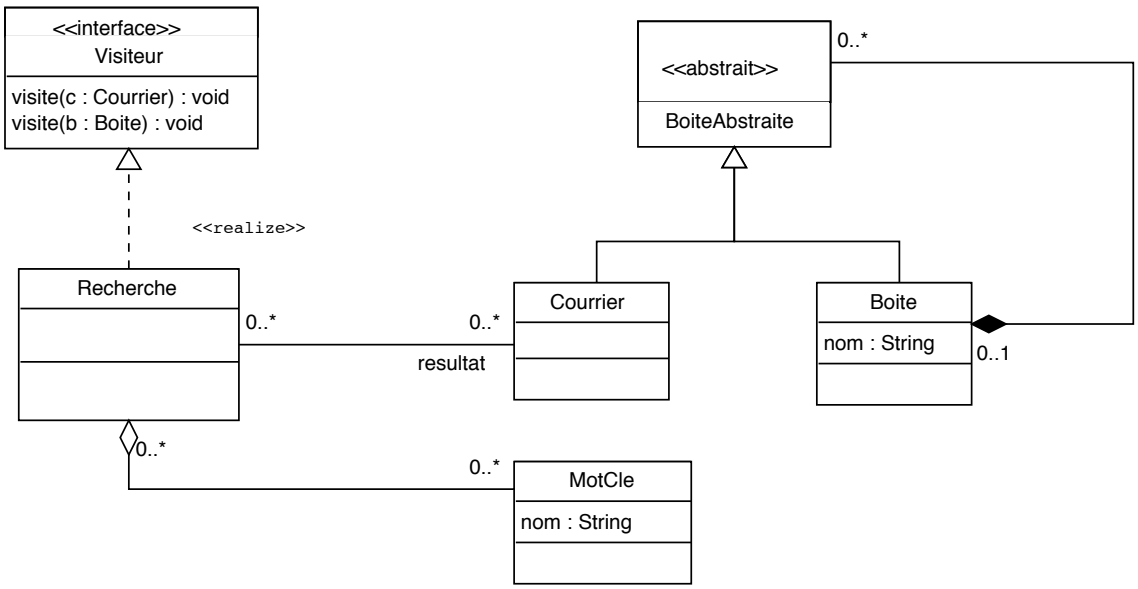

FIGURE 6 – Application du patron Visiteur

Code Java

```
// Fichier BoiteAbstraite.java
/**
 * Classe des boîtes abstraites. Les courriers et les boites
 * sont des boîtes abstraites.
 */
abstract class BoiteAbstraite {
   abstract void applique(Visiteur v);
}

// Fichier Courrier.java
/**
 * Classe des courriers.
 */
class Courrier extends BoiteAbstraite {
   String corps ; // Corps du courrier
   Courrier(String corps) {
      this.corps = corps ;
   }
   void applique(Visiteur v) {
      v.visite(this) ;
   }
}

// Fichier Boite.java
import java.util.Set ;
import java.util.HashSet ;
/**
 * Classe des boîtes, qui peuvent contenir d'autres boîtes,
 * ou des courriers.
 */
class Boite extends BoiteAbstraite {
   Set<BoiteAbstraite> contenu ; // Contenu de la boîte
   String nom ; // Nom de la boîte
   Boite(String nom) {
      this.nom = nom ;
      contenu = new HashSet<BoiteAbstraite>() ;
   }

   void applique(Visiteur v) {
      v.visite(this);
   }
}

// Fichier Visiteur.java
/**
 * Interface Visiteur du patron Visiteur.
 */
interface Visiteur {
   void visite(Courrier c) ;
   void visite(Boite b) ;
}
```

```java
// Classe Recherche.java
import java.util.Set ;
import java.util.HashSet ;
/**
 * Classe qui implémente la recherche de courriers qui contiennent
 * un ensemble de mots clés.
 * Le résultat se trouve dans this.resultat.
 */
class Recherche implements Visiteur {

   Set<Courrier> resultat ;
   Set<String> lesMotsCles ;

   Recherche(Set<String> lesMotsCles) {
      resultat = new HashSet<Courrier>() ;
      this.lesMotsCles = lesMotsCles ;
   }

   static boolean contient
         (Courrier courrier, Set<String> listeMotsCles) {
      boolean ok = true ;
      for (String motCle : listeMotsCles) {
         if (!courrier.corps.contains(motCle)) {
            ok = false ;
         }
      }
      return ok ;
   }

   public void visite(Courrier courrier) {
      if (contient(courrier, lesMotsCles)) {
         resultat.add(courrier);
      }
   }

   public void visite(Boite boite) {
      for (BoiteAbstraite ba : boite.contenu) {
         ba.applique(this) ;
      }
   }
}
```