# UEF4.3. Programmation Orientée Objet

## EXAMEN FINAL

**02 heures-Documents interdits**

### Questions de cours (4pts)

1. Soient A et B deux classes telles que B dérive de A. Comment se déroule la création d'un objet de type B ? (Citez les différentes étapes)

**Réponse :**

Dans une classe dérivée ( class B extends A {...}), la création d'un objet se déroule comme suit:

1. Allocation mémoire pour un objet de type B
2. Initialisation implicite de tous les champs de B
3. Initialisation explicite des champs hérités de A (éventuellement)
4. Exécution du corps du constructeur de A
5. Initialisation explicite des champs propres à B (éventuellement)
6. Exécution des corps du constructeur de B

2. Le polymorphisme se base sur deux règles essentielles. Lesquelles ?

**Réponse :**

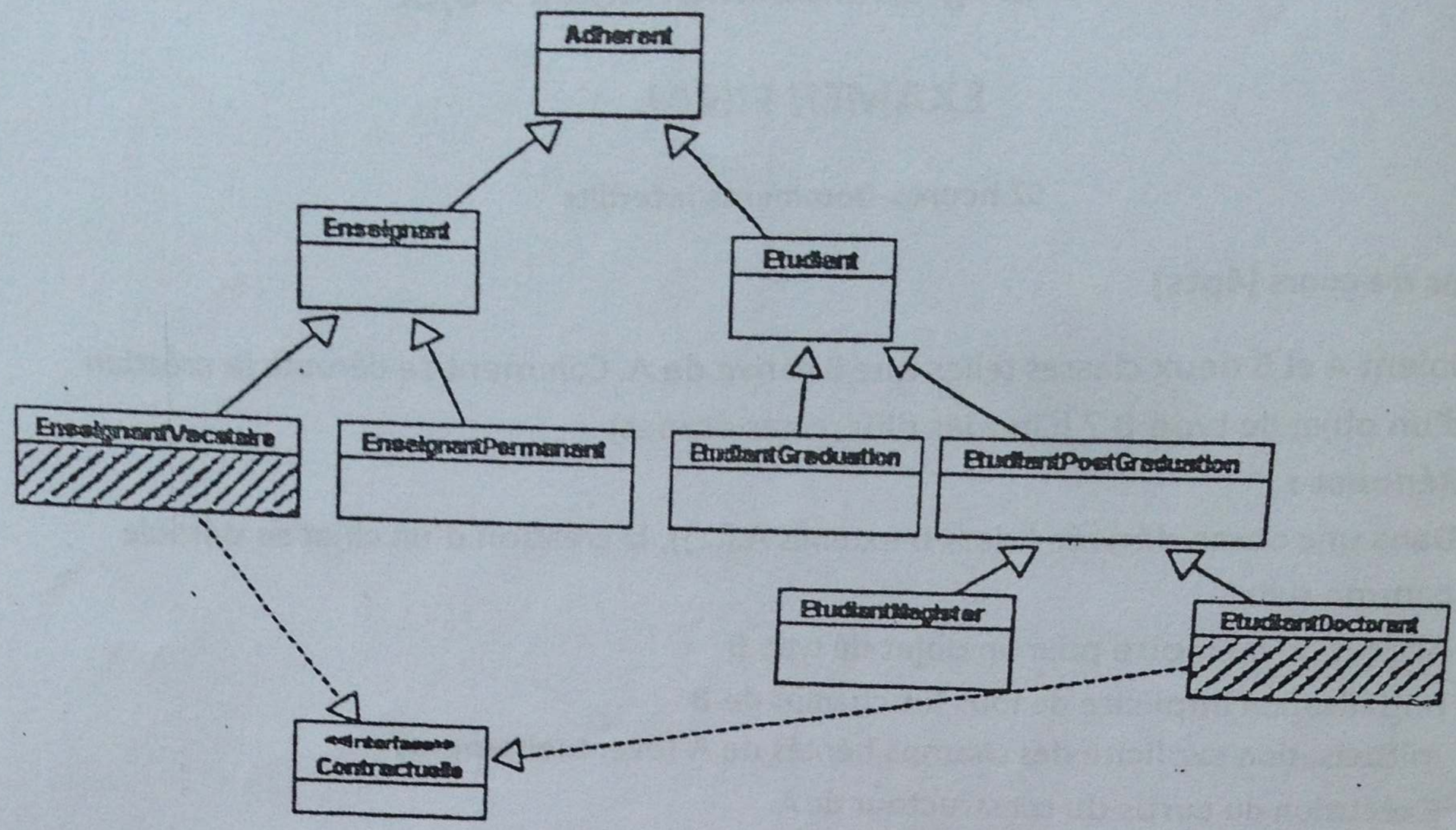
**Règle 1** la possibilité d'affecter à une variable objet, une référence à un objet d'un type dérivé (Doivent être taillés)

**Règle 2** Le choix de la méthode appelée se fait selon le type effectif de l'objet référencé au moment de l'exécution (ligature dynamique)

suffit pour (1)

### Exercice 1 (6pts)

Dans une bibliothèque universitaire, nous avons plusieurs types d'adhérents définis par les classes suivantes :




**Question 1 (2 pts):** Qu'affiche le programme suivant ?

```
public static void main(String args[]) {
Adherent P1 = new Adherent ();
Adherent P2 = new EnseignantPermanent();
Enseignant P3 = new EnseignantVacataire ();
Adherent P4 = new EtudiantPostGraduation();
Etudiant P5 = new EtudiantDoctorat ();
EtudiantPostGraduation P6 = new EtudiantMagister();
EtudiantGraduation P7 = new EtudiantGraduation();
EtudiantPostGraduation P8 = new EtudiantDoctorat ();
P1.emprunter(); P2.emprunter(); P3.emprunter(); P4.emprunter ();
P5.emprunter(); P6.emprunter(); P7.emprunter(); P8.emprunter();
}
```

**Réponse :**

```
je peux emprunter des livres
je peux emprunter 5 livres pour 15 jours
je peux emprunter 3livres pour 15 jours
je peux emprunter des livres
je peux emprunter 3 livres pour 15 jours
je peux emprunter 2 livres pour 15 jours
je peux emprunter 2 livres pour 10 jours
je peux emprunter 3 livres pour 15 jours
```

0,25 x 8

**Question 2 (4 pts) :** Dans la suite d'instructions suivantes, indiquez pour chacune :

- si elle est correcte ou si elle provoque une erreur à la compilation ou à l'exécution.
- si on peut résoudre le problème de la compilation par un cast (et donnez ce cast), et si oui, si il reste une erreur à l'exécution ou non.

-0.25 pour ⊗

Réponse :

| | Instruction | Réponse |
|---|---|---|
| 1 | Adherent a1 = new Adherent (); | ⊗ |
| 2 | Contractuelle a2 = new EnseignantVacataire (); | Ok (0,25) |
| 3 | Contractuelle a3 = new EtudiantDoctorat (); | Ok (0,25) |
| 4 | Etudiant a4 = new EtudiantGraduation (); | ⊗ |
| 5 | Etudiant a5 = a3; | Erreur de compilation : Impossible de convertir contractuelle à étudiant(0,25)<br>Cast : Etudiant a5 = (Etudiant) a3;<br>Ou<br>Cast : Etudiant a5 = (EtudiantDoctorat) a3;<br>(0,25) |
| 6 | a5 = a4; | Ok(0,25) |
| 7 | a2 = a3; | Ok (0,25) |
| 8 | Enseignant a6 = new Enseignant (); | ⊗ |
| 9 | a4 = a6; | Erreur de compilation : Impossible de convertir étudiant à enseignant(0,25)<br>Pas de cast possible(0,25) |
| 10 | EnseignantVacataire a7 = a6; | Erreur de compilation : Impossible de convertir enseignant à enseignant vacataire(0,25)<br>Cast : EnseignantVacataire a7 = (EnseignantVacataire)a6; ou (0,25)<br>Mais reste erreur à l'exécution(0,5) |
| 11 | a6 = a2; | Erreur de compilation : Impossible de convertir contractuelle à enseignant(0,25)<br>Cast : a6 = (Enseignant )a2;<br>Ou a6 = (EnseignantVacataire)a2 ;<br>(0,25) |
| 12 | EtudiantPostGraduation a8 =(EtudiantPostGraduation) a4; | Erreur d'exécution :<br>EtudiantGraduation ne peut pas être convertit en EtudiantPostGraduation(0,5) |

**Exercice 3 (10pts) : Conception d'un système de gestion de fichier**

**Questions** 12 + 12

1. Quelle structure de données proposez-vous pour :

**Solution :**

a. Conserver les fichiers dans le système, indexés par leurs emplacements.
HashMap<String, Fichier> : String est l'emplacement du fichier

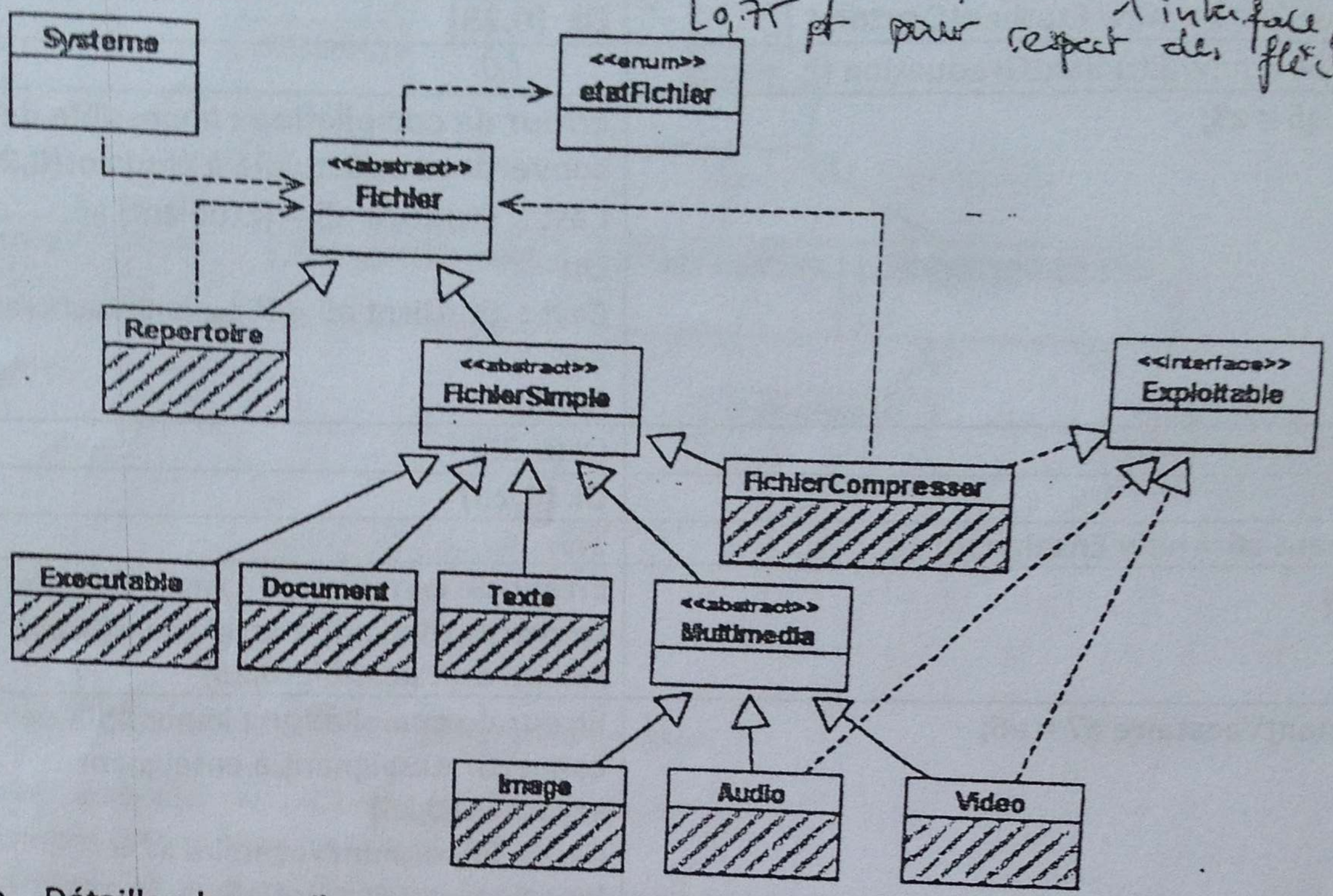
a. Conserver les fichiers d'un répertoire triés par leurs noms.
TreeSet<Fichier> : car on n'accepte pas les doublons+les fichiers doivent être triés

2. Proposez une modélisation orientée objet pour concevoir ce système de gestion de fichier :

   a. Tracez le diagramme des classes :

   3,5 pt pour l'identification de la classe
   1,25 pt pour type classe (3 abstraite, 1 interface, 1 enum)
   0,75 pt pour respect des flèches (0,25 x 3)

   **Réponse :**




   b. Détaillez les attributs et les méthodes de chaque classe dans un tableau (Ne pas inclure les constructeurs et les accesseurs (getters et setters)). Le tableau devra avoir la forme suivante : (Total = 11,5 pt)

**Réponse :**

| **class Systeme** | |
|---|---|
| Liste attributs | |
| private HashMap<String,Fichier> ensembleFichers | Les fichiers du système 0,25 |
| private int espaceDisponible | Espace disponible dans le système 0,25 |
| Liste des méthodes | |
| ~~Constructeur~~ void supprimer (Fichier F) | Désignation |
| public void ajouterFichier(Fichier f) | Méthode permettant d'ajouter un fichier au système 0,25 |
| public ArrayList<Fichier> trouverFichier(String chaine) (ou un tableau comme type de retour) | Méthode permettant de cherche un fichier 0,25 |

| **enum Etat** | |
|---|---|
| Liste attributs | 0,5 |
| lectureSeule, Cacher | |
| | |

| abstract class Fichier implements Comparable<Fichier> | |
|---|---|
| Liste attributs | |
| private String emplacement | |
| private String nom | |
| private int taille | |
| private Date dateCreation | on peut accepter String |
| private Date dateModification | |
| private Etat etatFichier | Etat du fichier (lecture seule, caché) |
| Liste des méthodes | |
| public abstract ArrayList<Fichier> rechercher(String chaine) | Méthode permettant d'effectuer une recherche dans un fichier retourne la liste du ou des fichiers trouvés |
| public int compareTo(Fichier f) | Compare le nom du fichier courant avec celui du fichier donné en paramètre |

| class Repertoire extends Fichier | |
|---|---|
| Liste attributs | |
| private TreeSet<Fichier> listeFichiers | Les fichiers contenus dansla racine du répertoire |
| Liste des méthodes | |
| public ArrayList<Fichier> rechercher(String chaine) | Cherche si le nom du fichier donné dans chaine correspond au nom du répertoire ou à l'un des fichiers qui le composent |

| abstract class FichierSimple extends Fichier | |
|---|---|
| Liste attributs | |
| private String extension; | Extension du fichier archive |
| private ArrayList<String> motsCles; (on peut accepter tableau ou autre type de collection) | Liste de mots clés identifiant le fichier. |
| Liste des méthodes | |
| public ArrayList<Fichier> rechercher(String chaine) | Redéfinie la méthode abstraite de la classe Fichier pour chercher si le nom donné dans chaine correspond au nom du fichier ou à l'un de ses mots clés. |

| class Executable extends Fichier Simple | |
|---|---|

| class Document extends Fichier Simple | |
|---|---|
| Liste des attributs | |
| private String texte | |
| private String police | |
| private Color couleur(même String on l'accepte) | |

| class Texte extends Fichier Simple | |
|---|---|

| Interface Exploitable | |
|---|---|
| Liste des méthodes | |
| public Fichier Decompresser() | Permet de décompresser un fichier archive ou décoder un fichier audio ou vidéo 0,5 |

| class Compresser extends Fichier implements Exploitable 0,25 0,25 | |
|---|---|
| Liste attributs | |
| private TreeSet<Fichier> listeFichiers ; 0,25 | Les fichiers contenus dans l'archive |
| Liste des méthodes | |
| public Fichier Decompresser() 0,25 | |

| abstract class Multimedia extends FichierSimple 0,25 | |
|---|---|
| class Image extends Multimedia 0,25 | |

| class Audio extends Multimedia implements Exploitable 0,25 0,25 | |
|---|---|
| Liste des méthodes | |
| public Fichier Decompresser() 0,25 | |

| class Video extends Multimedia implements Exploitable 0,25 0,25 | |
|---|---|
| Liste des méthodes | |
| public Fichier Decompresser() 0,25 | |

3. Implémenter en java le code de la méthode *ajouterFichier(Fichier f)*permettant d'ajouter un fichier au système. Nous supposons qu'une erreur sera lancée si la taille du fichier est supérieure à l'espace disponible. ( Total : 2pts)

**Réponse :**

Dans la classe Système :

```
public void ajouterFichier(Fichier f) throws TailleInsuffisanteException {   0,5 (vérifier la taille)
  if (f.getTaille()>espaceDisponible) throw new TailleInsuffisanteException() ;   0,5
  else{
   ensembleFichers.put(f.getEmplacement, f) ;   0,25
   espaceDisponible= espaceDisponible-f. getTaille() ;   0,25
}}
```

avec :

```
TailleInsuffisanteException extends Exception { }   0,5
```