Nom : .....

Prénom : .....

Promo : .....

Répondre sur l'énoncé. Toutes les réponses sont courtes si vous vous contentez de répondre à la question posée, sans commentaires sauf s'ils sont explicitement demandés.

Questions de cours

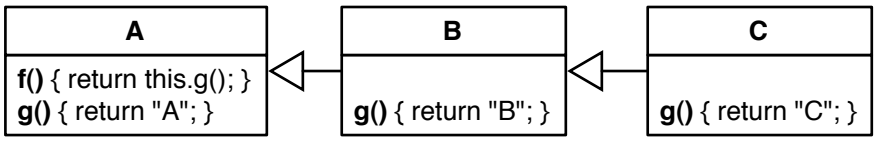
1. Qu'est-ce qu'un objet / une instance ?

Solution:
- une entité de programme ayant son identité propre, et groupant les données (l'état), avec le comportement associé
- un objet protège son état (encapsulation) ; on ne peut interagir avec lui que via les méthodes qu'il propose
- un objet délègue les traitements qui ne relèvent pas de sa responsabilité directe à d'autres objets
- tout objet est instance (individu) d'une classe

2. Qu'est-ce qu'une classe ?

Solution:
- les classes modélisent les concepts du cahier des charges
- chaque classe spécification commune d'une famille d'objets : ses instances
- chaque classe spécifie les variables d'instances, constructeurs, méthodes disponibles dans les instances de cette classe
- chaque classe permet d'obtenir des nouvelles instances (via le mot-clé `new` et les constructeurs)

3. On considère les trois classes `A`, `B` et `C` définies ainsi (toutes les méthodes ont pour type de retour `String`) :



(a) Lesquelles de ces instructions sont correctes ?

√ `A a=new A(); a.f();`	☐ `B b=new A(); b.f();`	☐ `C c=new A(); c.f();`
√ `A a=new B(); a.f();`	√ `B b=new B(); b.f();`	☐ `C c=new B(); c.f();`
√ `A a=new C(); a.f();`	√ `B b=new C(); b.f();`	√ `C c=new C(); c.f();`

(b) Quelle est la valeur de chacune de ces expressions Java, le cas échéant ?

`(new A()).f();`	`(new B()).f();`	`(new C()).f();`
Solution: `"A"`	**Solution:** `"B"`	**Solution:** `"C"`

(c) Même question, mais en ajoutant dans la classe `B` :

```
public String f() {
  return super.g() + this.g();
}
```

Solution: `"A"`	**Solution:** `"AB"`	**Solution:** `"AC"`

(d) Même question, mais avec, toujours dans la classe `B` :

```
public String f() {
  return super.f() + this.g();  // f au lieu de g pour le premier appel
}
```

Solution: `"A"`	**Solution:** `"BB"`	**Solution:** `"CC"`

Structure d'entreprise

Cahier des charges : Il y a des entreprises, des départements, et des employés.
— Chaque entreprise a un nom unique.
— Chaque entreprise consiste de départements possiblement imbriqués.
— Chaque département a un nom unique.
— Chaque département a un manager.
— Chaque département regroupe des employés et des sous-départements.
— Chaque employé a un nom unique et un salaire.
— Chaque employé ne travaille qu'à un seul poste dans l'entreprise.
— Les managers sont des employés aussi.

4. Identifiez les termes importants du cahier des charges, et rangez-les selon qu'ils désignent des classes, des variables d'instances, des relations ou des contraintes.

Solution:
classes :
- entreprise, département, employé
- possiblement : poste (mais les détails ne sont pas spécifiés), manager (comme sous-classe de employé)

variables d'instances :
- les noms des entreprises, des départements, des employés
- le salaire d'un employé
- possiblement : poste, manager (si on considère que manager est un poste, au lieu d'une catégorie d'employés)

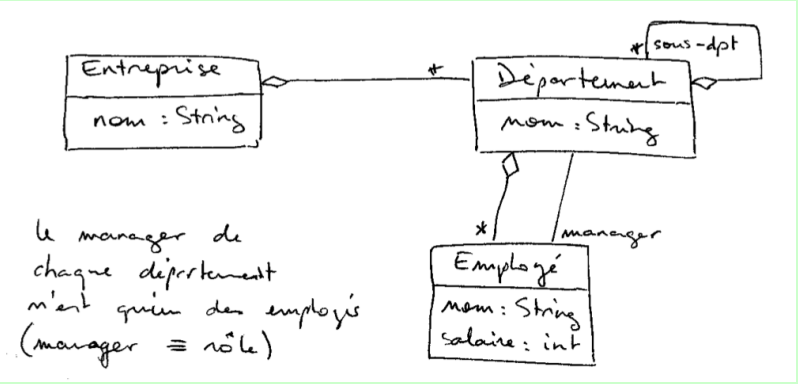
relations :
- composition des entreprises par des départements
- composition des départements en sous-départements
- l'attachement des employés à leur département
- possiblement : le manager d'un département (parmi ses employés), le poste d'un employé (si les postes sont modélisés par des objets), les postes disponibles dans l'entreprise

contraintes :
- unicité des noms entre entreprises, entre départements, entre employés
- selon la représentation des postes, unicité du poste pour chaque employé

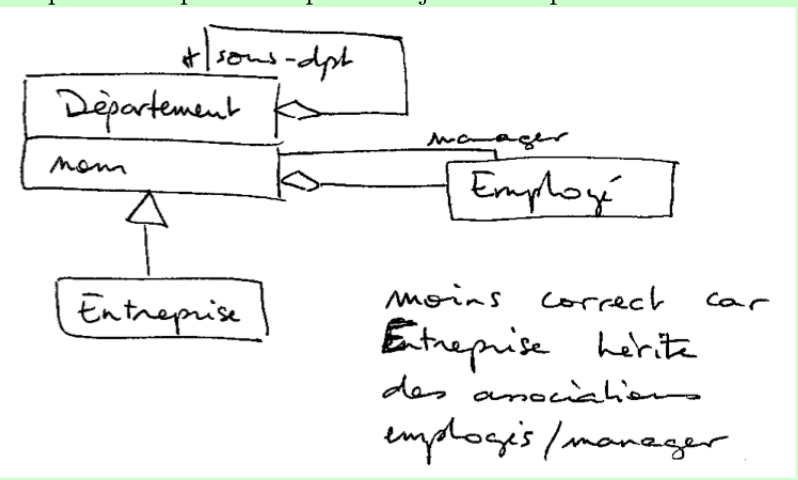
5. Modélisez sous forme de diagramme UML le système décrit ci-dessus. Concentrez-vous sur le modèle de données, c'est-à-dire les classes, leurs relations et variables d'instance, sans considérer les méthodes.

Solution: Il y a plusieurs solutions possibles, toutes acceptables, ce qui importe est la cohérence.

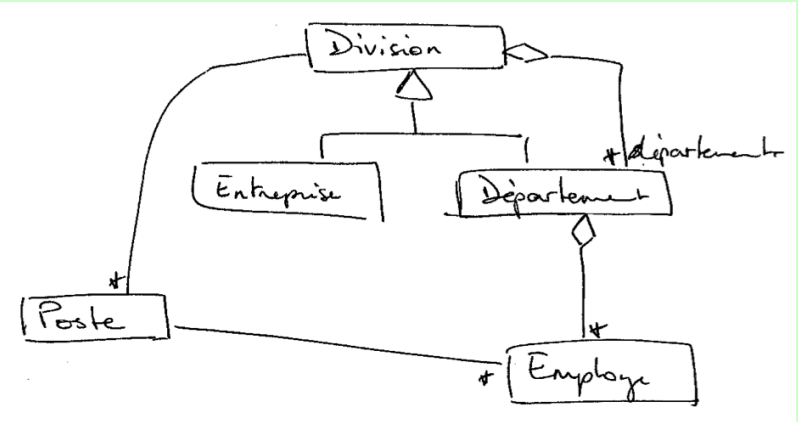
Solution assez directe, mais sans factorisation du nom et de l'aggrégation des départements :



On peut dire que l'entreprise est juste le département racine...



Ou adopter une version plus carrée mais complexe. Ici on a en plus modélisé les postes comme une classe à part, qui pourrait spécifier le salaire de tous les employés ayant ce poste, par exemple.



6. Donnez la déclaration en Java des classes du diagramme précédent. Pour économiser du temps, omettez les accesseurs `setXXX()`, `getXXX()`, `addXXX()`, etc, mais donnez bien les déclarations des variables d'instance et leurs valeurs d'initialisation.

Solution:

```
// correspond au 1er diagramme
// On considère qu'un employé a un poste quand il fait partie
// du personnel d'un département, avec une contrainte à respecter :
// un employé ne doit pas être dans plusieurs départements à la fois.
public class Entreprise {
    protected String nom;
    protected Set<Departement> departements;

    public Entreprise(String nom) {
        this.nom = nom;
        departements = new LinkedList<Departement>();
    }
}

public class Departement {
    protected String nom;
    protected Set<Departement> sousDepartements;
    protected Set<Employe> personnel;
    protected Employe manager;

    public Departement(String nom, Manager mgr) {
        this.nom = nom;
        sousDepartements = new LinkedList<Departement>();
        personnel = new LinkedList<Employe>();
        personnel.add(mgr);
        manager = mgr;
    }
}

public class Employe {
    protected String nom;
    protected int salaire;

    public Employe(String nom, int salaireInitial) {
        this.nom = nom;
        salaire = salaireInitial;
    }
}
```

7. Implémentez l'opération `masseSalariale()` renvoyant la somme des salaires des employés d'une entreprise, tous départements confondus. Donnez le code de toutes les méthodes nécessaires, en spécifiant bien pour chacune dans quelle classe l'ajouter.

Solution:

```
// idem, correspond au 1er diagramme et code précédent

// dans la classe Entreprise :
public int masseSalariale() {
    int sum = 0;
    for (Departement d : departements) {
        sum += d.masseSalariale();
    }
    return sum;
}

// dans Departement :
public int masseSalariale() {
    int sum = 0;
    for (Departement d : sousDepartements) {
        sum += d.masseSalariale();
    }
    for (Employe e : personnel) {
        sum += e.getSalaire();
    }
    return sum;
}
```