Travaux pratiques N°4

Programmation Orientée Objet en Java
2ème Année Génie Informatique
Enseignant : Tarik BOUDAA
AU : 2018/2019

Exercice I : Gestion des PFE des étudiants de l'ENSAH (*Extrait de l'examen 2015*)

Le Service de stage de l'ENSAH souhaite réaliser une application Java pour la gestion des projets de fin d'études (PFE), ainsi il a rédigé la description ci-dessous :

L'étudiant réalise un ou plusieurs PFE durant son cursus à l'ENSAH, un PFE est caractérisé par son sujet, sa date de début, le nom de l'entreprise dont laquelle a été réalisé, ses livrables, sa moyenne générale et son encadrant à l'ENSAH (Un professeur).

Actuellement, les livrables peuvent être des rapports ou/et des codes sources. Tous les livrables doivent être imprimables.

Un code source est caractérisé par son contenu (sous forme d'une chaîne de caractères), et on précise son langage de programmation.

Un rapport est caractérisé par son nombre de page, sa langue de rédaction, son contenu (sous forme d'une chaîne de caractères) et son titre.

La soutenance d'un PEE se fait devant un jury d'au moins 3 professeurs de l'ENSAH. Les professeurs peuvent être membres dans plusieurs jurys.

Un étudiant est caractérisé par les informations suivantes : CNE, Nom, Prénom, Télé, Email et Filière.

Un professeur est caractérisé par les informations suivantes : Nom, Prénom, Télé, Email, Grade et CIN.

Un étudiant stagiaire a réalisé une conception préliminaire des classes métier sur papier, pour modéliser la description précédente. Cette conception préliminaire ; décrite sur la figure 1 à la page 2 ; a été validée, ainsi elle ne doit être complétée qu'en ajoutant les attributs nécessaires.

Questions :

1. Expliquer l'intérêt éventuel d'avoir l'interface ***Livrable*** et la classe ***Personne***.
2. Compléter le diagramme de classe préliminaire de la figure 1, en y ajoutant les attributs nécessaires. (*L'ajout des méthodes dans ce diagramme n'est pas demandé*).
3. Les classes ***Personne, Prof*** et ***Etudiant*** disposent d'un constructeur qui permet d'initialiser correctement leurs attributs, la classe ***Personne*** dispose, en plus, d'une redéfinition de la méthode *equals*. On considère que deux objets de type Personne sont égaux s'ils ont les mêmes valeurs d'attributs.
   Les classes ***Prof*** et ***Etudiant*** possèdent les méthodes nécessaires pour associer un PFE à son encadrant (à un professeur) et un PFE à son réalisateur (à un étudiant).
   Ecrire le code des classes ***Personne***, ***Prof*** et ***Etudiant***.
4. La classe ***Jury*** a un constructeur permettant d'initialiser son attribut avec une liste d'objets de type ***Prof***.
   Ecrire le code de la classe ***Jury.***
5. La méthode ***print*** de la classe ***Rapport***, imprime le titre et le texte du rapport dans la console, et celle de la classe ***CodeSource*** imprime sur la console le nom du langage utilisé et le contenu du code source. Ces deux classes disposent des constructeurs nécessaires pour initialiser leurs attributs.
   Ecrire le code de l'interface ***Livrable*** et des classes ***CodeSource*** et ***Rapport.***

6. La classe ***Pfe*** disposent d'un constructeur qui permet d'initialiser correctement tous ses attributs. Elle possède également les méthodes ***printAllLivrables*** et ***affecterJury*** . La première permet d'imprimer tous les livrables d'un PFE et la deuxième permet d'affecter un jury de soutenance à un PFE.
   Ecrire le code de la classe ***Pfe***.

7. Ecrire une classe MainProg constituant le programme principal (ayant la méthode main) permettant d'exécuter les actions suivantes dans l'ordre:

a- Créer un étudiant ayant les informations suivantes :

CNE	Filière	Nom	Prénom	Télé	email
1010100	GI	Erradi	Said	06125522	said@gmail.com

b- Créer les Professeurs ayant les informations suivantes :

Grade	CIN	Nom	Prénom	Télé	email
PA	R1221	Salimi	Amine	06125525	amine@gmail.com
PH	R1222	Ramali	Rafik	06125526	rafik@gmail.com
PES	R1223	Nafili	Karima	06125527	sadik@gmail.com
PH	R1224	Bahi	Salim	06125521	bahi@gmail.com

c- Créer le PFE ayant les informations suivantes :

Sujet	Date début	Entreprise	Moyenne	Rapport	Code sources	Jury	Encadrant	Réalisé par
Application de gestion des PFE	22/12/2015	Atos	16	**Titre** : Rapport de stage de l' Application de gestion des PFE **Langue** : Français	**Langage** : Java **Code source** : classe A{…}	**Profs** : -Ramali -Nafili -Bahi	**Prof** : Salimi	**Etudiant** : Erradi

d- Imprimer les livrables du PFE précédent

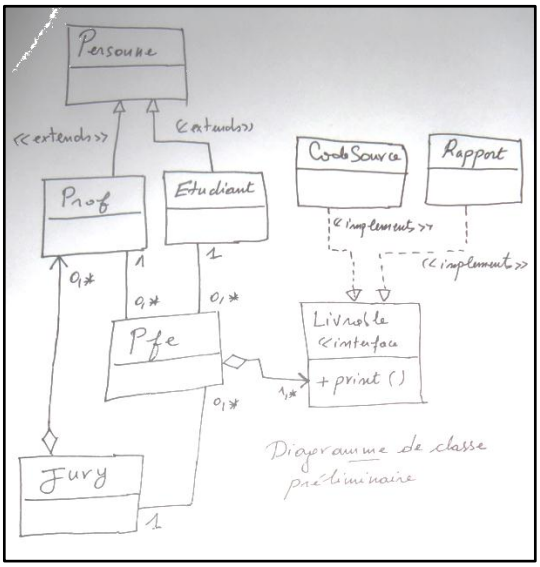

Figure 1 : conception préliminaire des classes métier

Exercice II : Application de gestion des contacts

Ecrire une application Java pour la gestion des contacts ayant des fonctionnalités similaires à celles qui se trouvent sur votre Smartphone en utilisant deux méthodes différentes :

1. Avec JDBC
2. Avec un ORM (Hibernate)

Exercice III : Principe d'injection de dépendance (*dependency injection*)

L'injection de dépendances (*dependency injection*) est un Design Pattern permettant, en programmation orientée objet, de découpler les liens de dépendances entre objets.

En programmation objet, les objets de type A dépendent d'un objet de type B si au moins une des conditions suivantes est vérifiée :

- A possède un attribut de type B (dépendance par composition) ;
- A est de type B (dépendance par héritage) ;
- A dépend d'un autre objet de type C qui dépend d'un objet de type B (dépendance par transitivité) ;
- une méthode de A appelle une méthode de B.

Si A dépend de B, cela implique que pour créer A, on a besoin de B ce qui, en pratique, n'est pas toujours le cas.

Pour supprimer la dépendance, un moyen possible consiste à utiliser le pattern d'injection de dépendance, qu'on essaiera de mettre en œuvre sur un exemple simple qui fait l'objet de cet exercice :

On suppose qu'une application est constituée de deux packages *com.ensah.composanta* et *com.ensah.composantb.*

Le package *composanta* contient entres autres une classe *MathCalcul* ayant une méthode *max2Entier* qui renvoie le maximum de deux entiers :

```
package com.ensah.composanta;

public class MathCalcul{
    /** Le max de deux entier*/
    public int max2Entier(int a, int b){
       return a>b ? a : b;
    }
}
```

Le package *composantb* contient entres autres une classe *MathSommation* ayant une méthode *max3Entier* qui renvoie le maximum de 3 entiers passés en paramètres.

On suppose que pour implémenter la méthode *max3Entier* on doit utiliser *max2Entier* de la classe *MathCalcul*. Implémenter la classe *MathSommation* et la méthode *max3Entier* sans introduire des dépendance dans le sens *MathSommation* vers *MathCalcul* (autrement dit vous avez pas le droit d'instancier un objet de type *MathCalcul* dans la classe *MathSommation*)

Exercice IV : Implémentation d'une architecture multicouches

Modifier le code de l'application Gestion des comptes du TP2 pour qu'il utilise une base de données pour stocker les comptes bancaires.

- Définir une **architecture en couches** pour l'application (DAO / BLL / BO / GUI)
- Utiliser le pattern **DAO** pour la conception des classes d'accès aux données.
- Utiliser le design pattern **Abstract Factory** pour garantir une indépendance avec le type du système de sauvegarde de données utilisé (XML, MySQL, Oracle, …).
- Utiliser **l'injection de dépendance** pour garantir une indépendance entre les couches de l'application
- Les règles de gestion doivent êtres implémentées dans la couche BLL.
- Utiliser **Hibernate** pour la gestion de la persistance