UE Programmation Orientée Objet

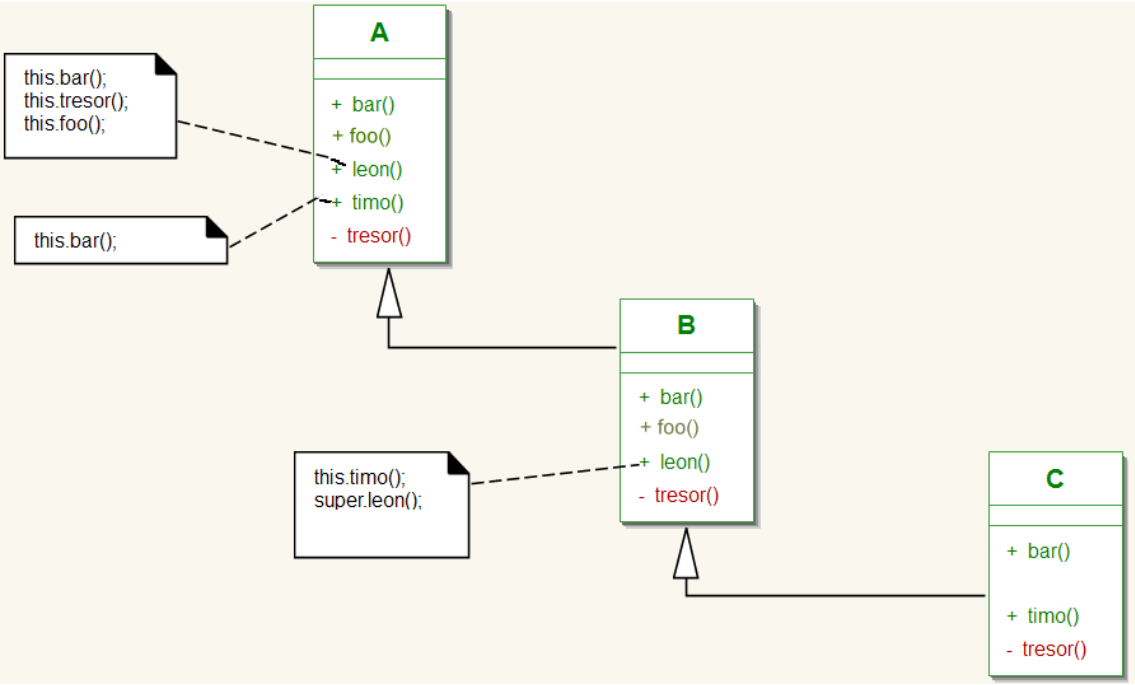
Devoir Surveillé

Mercredi 20 mai 2015 – 8h-10h
Copie des diapositives de cours + une feuille recto-verso de notes personnelles autorisées
Dictionnaire de langue (papier ou électronique "dédié") autorisé
Tout autre document interdit

Les exercices sont indépendants. Leur ordre ne préjuge pas de leur difficulté. Il est conseillé de lire entièrement un exercice avant de le traiter.
La javadoc et les méthodes de test ne sont à fournir que si elles sont demandées.
Les durées indiquées sont données à titre indicatif et prennent en compte la lecture du sujet.

Exercice 1 : Lookup (15mn)

On donne le diagramme de classes suivant :



En plus des portions de code indiquées sur le diagramme, chacune des méthodes commence par l'instruction : « `System.out.println("`*`NomDeClasse.nomMéthode`*`");` » où *NomDeClasse* est évidemment remplacé par le nom de la classe dans laquelle le corps de la méthode est codé et *nomMéthode* par le nom de cette méthode[1].

Q 1 . Indiquez **précisément** ce qu'affiche le programme suivant (sans explication complémentaire) :

```
public static void main(String[] args) {
   A ref;

   ref = new B();
   System.out.println("--- appel leon 1 ---");
   ref.leon();

   ref = new C();
   System.out.println("--- appel leon 2 ---");
   ref.leon();

   System.out.println("--- appel timo ---");
   ((A) ref).timo();
}
```

[1] L'exécution de « `new A().bar()` » produit donc l'affichage « `A.bar` »

Exercice 2 : Bibliothèque (65mn)
On s'intéresse à la gestion d'une bibliothèque et en particulier à la gestion des emprunts des ouvrages.

Tous les types de cet exercice appartiendront au paquetage `bibliotheque`.

Les livres Les livres de la bibliothèque sont représentés par le type :

bibliotheque::Livre
- title : String - id : String - dureeMaxEmprunt : int
+ Livre(title : String, id : String) + getTitle() : String + getId() : String + getDureeMaxEmprunt() : int ...

Chaque livre est identifié de manière unique par la valeur de son attribut `id`.
La méthode `getDureeMaxEmprunt` fournit la durée d'emprunt maximale autorisée pour ce livre exprimée en nombre de jours.

Dates d'emprunt Chaque emprunt fait à la bibliothèque est daté. Les dates d'emprunt sont représentées par des objets de la classe `DateEmprunt`.
Une date d'emprunt est représentée par une année et un entier correspondant au numéro du jour dans l'année. Ainsi pour le *20 mai 2015*, ce nombre est *140* car le *20 mai* est le 140ème jour de l'année *2015*. Ainsi on crée un objet `DateEmprunt` correspondant au *20 mai 2015* par `new DateEmprunt(2015,140)`.
Le « squelette » publique de la classe `DateEmprunt` est le suivant :

```
package bibliotheque;
public class DateEmprunt {
    ...
    /** construit une date avec l'année et le numéro du jour donnés
     * @param annee l'année de cette date
     * @param numJour le numéro du jour dans l'annnée pour cette date
     */
    public DateEmprunt(int annee, int numJour) { ... }
    /** @return l'année de cette date */
    public int getAnnee() { ... }

    /** @return le numéro du jour dans l'année de cette date */
    public int getNumJour() { ... }

    /** fournit le nombre de jours (positif) entre cette date et autreDate
     * (Les années des 2 dates peuvent être différentes)
     * @param autreDate l'autre date avec laquelle l'écart en jours est calculé
     * @return le nombre (positif) de jours d'écart entre cette date et autreDate
     */
    public int ecartEnJours(DateEmprunt autreDate) {
      // ...
    }
    /** @return un objet date correspondant à la date du jour de
     * l'appel de la méthode, soit <q>aujourd'hui</q> */
    public static DateEmprunt aujourdhui() { ... }
}
```

La méthode statique `aujourdhui` fournit un objet `DateEmprunt` correspondant à la date du jour où elle est invoquée. Le jour du devoir surveillé il s'agira donc d'un objet tel que `new DateEmprunt(2015,140)`.

Q 1 . (7mn) Donnez le code d'une méthode de test[2] « JUnit » qui permet de vérifier le bon fonctionnement d'une implémentation de la méthode `ecartEnjours`.

[2] Uniquement la méthode de test pas la classe.

Emprunteurs et bibliothèques Les bibliothèques et leurs emprunteurs sont modélisés par les classes :

bibliotheque::Bibliotheque
+ NB_EMPRUNTS_DEFAUT : int = 3 - emprunteurs : List<Emprunteur>
+ emprunteursHorsDelai() : List<Emprunteur> ...

bibliotheque::Emprunteur
- nom : String - id : String - nbMaxEmprunts : int - emprunts : Map<Livre,DateEmprunt>
+ Emprunteur(nom : String, id : String) + emprunteLivre(livre : Livre) + rendLivre(livre : Livre) : int + estHorsDelai() : boolean ...

Pour la classe `Emprunteur` :

- l'attribut `nbMaxEmprunts` correspond au nombre maximum de livres que peut emprunter l'emprunteur, à la construction sa valeur est définie par la constante `NB_EMPRUNTS_DEFAUT` de la classe `Bibliotheque`.
- l'attribut `emprunts` gère les livres empruntés en associant dans la table chaque livre emprunté avec sa date d'emprunt

Q 2 . (6mn) Donnez le code java de l'entête, des attributs et du constructeur de la classe `Emprunteur`.

Q 3 . (6mn) Que faut-il nécessairement ajouter à la classe `Livre` pour une bonne gestion de la table `emprunts` de `Emprunteur` ?

Donnez le code java correspondant.

Q 4 . (6mn) Donnez le code java de la méthode `empruntеLivre` qui réalise l'emprunt d'un livre par l'emprunteur.

Cette méthode lève une exception `TropDEmpruntsException` si l'emprunteur a déjà atteint son nombre maximal d'emprunts (avant cet emprunt).

Q 5 . (3mn) Donnez le code java de la classe `TropDEmpruntsException`.

On supposera la classe `ErreurEmpruntException` définie de manière similaire.

Q 6 . (10mn) Donnez le code java et la documentation « javadoc » de la méthode `rendLivre` qui permet de rendre un livre emprunté. Le livre est alors supprimé des emprunts et la durée de l'emprunt en nombre de jours est fournie en résultat. Le livre est supposé rendu le jour où la méthode `rendLivre` est appelée.

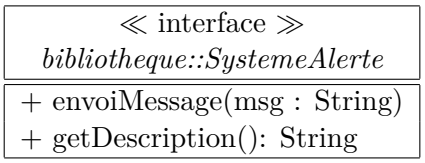
Cette méthode lève une exception `ErreurEmpruntException` si cet emprunteur n'a pas emprunté le livre fourni en paramètre.

Q 7 . (7mn) Donnez le code java de la méthode `estHorsDelai` dont le résultat vaut `true` si et seulement si l'emprunteur a dépassé la durée maximale d'emprunt autorisée pour au moins l'un des livres en cours d'emprunt.

Pour la classe `Bibliotheque` :

Q 8 . (6mn) Donnez le code de la méthode `emprunteursHorsDelai` qui fournit la liste des emprunteurs qui ont dépassé la durée maximale d'emprunt autorisée pour au moins l'un de leurs emprunts.

Afin de pouvoir alerter les emprunteurs hors délai, on propose d'ajouter un système d'alerte automatique. L'interface suivante permet de représenter de tels systèmes.
Il existe différents systèmes d'alerte. Par exemple la classe `AlerteSMS` permet d'envoyer l'alerte par sms, la classe `AlerteTweet` envoie le message à un compte Twitter choisi et on peut en imaginer beaucoup d'autres, comme une classe `AlerteConsole` qui se contente d'afficher le message d'alerte sur la sortie standard.



La méthode `envoiMessage` réalise l'envoi du message d'alerte et la méthode `getDescription` fournit simplement un descriptif du système d'alerte.
Chaque emprunteur peut choisir son système d'alerte. La classe `Empunteur` est donc complétée par un attribut `systemAlerte` de type `SystemeAlerte` et les *getter* et *setter* associés (code non demandé).

Q 9 . (6mn) Donnez un code java pour la classe `AlerteConsole`.

Q 10 . (6mn) Donnez le code java d'une méthode `alerteEmprunteurs` de la classe `Bibliotheque` qui a pour effet d'envoyer le message d'alerte `"vous êtes hors délai"` à chaque emprunteur hors délai.

Exercice 3 : Biens immobiliers (40mn)

On s'intéresse ici à la modélisation de biens immobiliers[3].
Un bien `Immobilier` est caractérisé par une surface totale du bien (en nombre entier de m^2) et une adresse (de type `Adresse` supposée définie). Ces données sont fournies à la création. Un objet représentant un bien immobilier dispose d'une méthode `surfaceImposable` qui fournit la surface du bien soumise à impôt, par défaut il s'agit de la surface totale.
On distingue parmi ces biens, les `Habitation`s des bâtiments à usage professionnel (`BatimentProfessionnel`).
Les habitations sont caractérisées par un nombre de pièces fourni à la création. Parmi les habitations on trouve :

- les `Appartement`s qui sont caractérisés en plus par le numéro de l'étage où se trouve l'appartement. Les `Studio`s sont des appartements particuliers n'ayant qu'une seule pièce. Un coefficient de 0,9 est alors appliqué par rapport à la surface imposable retenue pour un appartement.
- les `Maison`s qui sont caractérisées en plus par la surface du terrain sur lequel est bâtie la maison.

Les bâtiments à usage professionnel sont divisés entre :

- les `Commerce`s qui sont caractérisés par le fait qu'une partie de leur surface est occupée par un entrepôt (d'une certaine surface fournie à la construction). La surface de cet entrepôt est déduite de la surface totale pour obtenir la surface imposable d'un commerce.
- les `Ecole`s caractérisées par un nombre de salles de classe et un `Niveau` (maternelle, élémentaire, collège ou lycée). Les écoles ne sont pas imposables.

Les différentes informations définissant les biens immobiliers (habitations ou commerces) sont fournies à la création des objets et des accesseurs sont disponibles.

Q 1 . (5mn) Faites une proposition pour le type `Niveau` et donnez son code.

Q 2 . (25mn) Faites une proposition de modélisation pour les données présentées ci-dessus. Vous présenterez votre solution sous la forme d'un diagramme UML **clair** et **détaillé** dans lequel apparaissent :

- les liens d'héritage/implémentation entre types,
- les noms et types de tous les attributs,
- les méthodes et constructeurs avec leurs paramètres et leurs types ainsi que les types des valeurs de retour. Les méthodes doivent apparaitre dans chaque classe qui définit un nouveau comportement pour cette méthode.

Vous devez faire apparaître dans votre diagramme **au minimum** tous les éléments `soulignés` du texte précédent.

Si besoin présentez votre diagramme en "format paysage" sur votre copie.

Q 3 . (10mn) Donnez un code java pour le type permettant de représenter les studios.

[3]Le sujet ne se veut absolument pas exhaustif sur ce thème, ni même réaliste, et certains choix très arbitraires sont faits pour les besoins de l'exercice