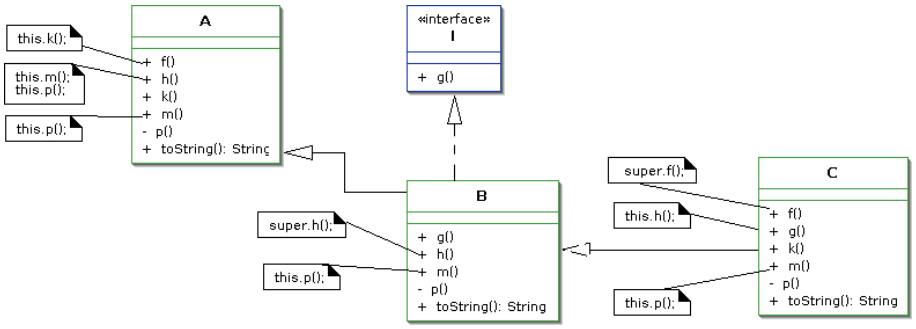
UE Conception Oriente Objet

Devoir Surveillé

durée : $1\text{h}\frac{1}{4}$

Exercice 1 : Héritage

On donne le diagramme de classes suivant :



La méthode `toString` de chaque classe renvoie une chaîne de caractères correspondant au nom de la classe. En plus des portions de code indiquées sur le diagramme, chacune des autres méthodes commence par l'instruction : "`System.out.println("`*NomDeClasse*`.`*nomMéthode*`");`" où *NomDeClasse* est évidemment remplacé par le nom de la classe où est déclarée la méthode et *nomMéthode* par le nom de cette méthode.

Q 1 . Soit la déclaration : *T* `ref = new C();`

Indiquez toutes les valeurs que peut prendre le type *T*.

Q 2 . Indiquez précisément ce qu'affiche le programme suivant :

```
public static void main(String[] args) {
     List<A> listeA = new ArrayList<A>();
     listeA.add(new A());
     listeA.add(new B());
     listeA.add(new C());

     for(A ref : listeA) {
         System.out.println("---"+ref+".f---");
         ref.f();
         System.out.println("---"+ref+".h---");
         ref.h();
     }
}
```

Exercice 2 : Location

On s'intéresse à des "choses" qu'il est possible de louer à la journée. Pour être louée une telle chose, de type `Louable`, doit être disponible (méthode `estDisponible():boolean`) et le client-locataire (de type `Personne` supposé défini et disposant de toutes les fonctionnalités nécessaires) doit satisfaire la contrainte de location (méthode `estLouable(p : Personne):boolean`. Un prix de location à la journée, fixée à la construction, est associé à chaque objet (méthode `getTarifJournee():float`).

Parmi les "choses" louables on trouve :

- les *vidéos*, qui sont caractérisées par une durée et un titre, il n'y a par défaut pas de contrainte de location pour les vidéos, cependant on distingue le cas particulier des "vidéos pour adultes" que seul un majeur peut louer ;
- les *chambres* (louées dans un hôtel), qui disposent d'un numéro (méthode `getNumero():int`), la contrainte de location est une caution égale au tarif de la location pour une journée ;
- les *véhicules* pour lesquels la contrainte de location est d'avoir le permis. Parmi les véhicules on distingue les *voitures* qui sont caractérisées par un nombre de passagers possible (méthode `getNbPlaces():int`) et les *véhicules utilitaires* qui sont caractérisés par un volume (méthode `getVolumeMax():float`) et un poids maximum transporté `getPTAC():int`.

Contraintes. Les contraintes de location qui s'appliquent aux objets louables peuvent être de natures différentes :

- *aucune contrainte* : tout le monde peut louer
- *contrainte d'âge* : il faut avoir un âge minimum fixé par la contrainte pour louer l'objet
- *contrainte permis* : il faut avoir le permis de conduire pour louer l'objet
- *contrainte caution* : il faut que le client accepte de laisser une caution d'un montant fixé à la construction de la contrainte
- etc.

Les contraintes sont modélisées par un type `Contrainte` qui dispose d'une méthode `estSatisfaite(p:Personne):boolean` dont le résultat est `true` si la personne `p` satisfait la contrainte (`p` peut donc louer l'objet concerné).
Il existe également des *contraintes multiples* qui cumulent les conditions de plusieurs autres contraintes, par exemple elles imposent qu'il est nécessaire d'avoir le permis et de déposer une caution, ou encore qu'il faut avoir le permis et au moins 21 ans.

Q 1 . Indiquez sous forme de diagrammes UML comment vous proposez de modéliser les contraintes, en faisant apparaître celles mentionnées dans le sujet. En particulier, faites apparaître clairement votre proposition pour les contraintes *multiples*.

Q 2 . Donnez le code java de la méthode `estSatisfaite` pour les contraintes multiples.

Q 3 . Indiquez sous forme de diagrammes UML détaillé comment vous proposez de modéliser les choses `Louables`. A nouveau faites apparaître les notions mentionnées dans le sujet.

Q 4 . Donnez le code java des types `Louable` et `Chambre`.

Q 5 . On appelle `Loueur` une société qui a pour objet la location de ces objets `Louables`. Un objet représentant une telle société doit disposer d'une méthode `loue(Personne p)` qui a pour résultat l'objet loué à la personne `p`.

On trouve des loueurs "universels" qui louent tout ce qu'il est possible de louer et des loueurs spécialisés tels que les *vidéoclubs* qui sont des sociétés spécialisées dans la location de vidéo, les *hôtels* qui sont des sociétés spécialisées dans la location de chambre et les *loueurs de voiture* qui ne louent que des voitures. Evidemment, il est nécessaire que pour des objets modélisant ces entités la méthode `loue` ait pour résultat un objet du bon type (`Video` pour le premier, `Chambre` pour le second, etc.).

Q 5.1. Faites une proposition pour un type `Loueur` : fournissez un diagramme de classe détaillé et donnez la signature de la méthode `loue` associée à ce type.

Q 5.2. Indiquez comment créer un objet *vidéo-club* et un objet *hôtel*.