Programmation objet TD 3
Classes et méthodes abstraites, Polymorphisme
Variables et méthodes de classe

Exercice 1

1. Quelles sont les différences entre méthode abstraite et méthode finale ?
2. Quelles sont les différences entre classe abstraite et classe finale ?
3. Quelles sont les différences entre variable abstraite et variable finale ?
4. Qu'est ce qu'une variable de classe ?
5. Qu'est ce que le polymorphisme ?

Exercice 2

Nous avions vu des classes dans les TD précédents qui permettaient de gérer des agences de location. Nous souhaitons dorénavant regrouper ces agences dans le cadre d'une entreprise particulière. Nous pourrons ainsi gérer plusieurs entreprises particulières et à travers elles avoir accès à leurs agences. Rappelons l'organisation des agences fournie aux td précédents.

```
Classe publique Agence
 champs... privé administratifs :nbe ,privé nom :Chaine,
 méthode... public taillePersonnel():nbe retourner administratifs
Classe publique Bureau sous classe de Agence
 champs... privé commerciaux :nbe, privé garages : ensemble de Garage
 public taillePersonnel():nbe retourner super.taillePersonnel() + commerciaux
 public nbMecaniciens():nbe i<-0,pour chaque E de garages, i=i+E<=nbMecaniciens(),
return i
Classe publique Garage sous classe de Agence
 champs privé mecaniciens nbe
 public nbMecaniciens(): nbe  retourner mecaniciens
 public nbe taillePersonnel() retourner super.taillePersonnel() + mecaniciens
```

Il est demandé de créer une classe représentant l'`Entreprise` de location dans son entier, c'est à dire l'ensemble des `agences`, qu'il s'agisse de bureaux de location ou de garages.

1. Ecrire la déclaration de la classe représentant l'entreprise de location.
2. Ecrire un constructeur sans paramètre qui pour la classe `Entreprise` remplisse le champs `agences`. Ecrire la méthode `ajouter` qui ajoute une agence à une entreprise de location.
3. Ecrire une méthode `taillePersonnel` qui retourne le nombre de personnes travaillant dans l'entreprise de location.
4. Que pensez vous de la classe `Agence`; existera t il des instances de cette classe ? que proposez vous ? quel est le gain espéré ? la méthode `taillePersonnel` de cette classe doit elle devenir abstraite, et pourquoi ?
5. a) Soit une entreprise `E` contenant les deux agences `A1` qui est un bureau, et `A2` qui est un garage, lors de l'appel de la méthode `taillePersonnel` sur E, quelles sont les méthodes `taillePersonnel` qui seront utilisées parmi celles de `Entreprise, Agence, Bureau, Garage` ?
   b) lors de l'appel précédent, quelle(s) méthode(s) de nom `taillePersonnel` va être appelée(s) sur A1 (parmi celles de `Entreprise, Agence, Bureau, Garage)` ? et sur A2 ? comment appelle t on ce phénomène ?
6. Ecrire une méthode `recherche` dans la classe `Entreprise` qui recherche une agence par son nom, et retourne l'agence en question si elle existe et null sinon. Attention, cette méthode peut nécessiter l'écriture d'une autre méthode.
7. Modifier `Agence` et ses sous-classes pour rajouter à chacune la méthode `taillePersonnelNonAdministratif` retournant le nombre de personnels non administratifs de l'agence.

8. Ecrire une méthode `taillePersonnelNonAdministratif` retournant le nombre de personnels non administratifs de toutes les agences d'une entreprise de location.

Exercice 3

Nous avions vu des classes dans les TD précédents qui permettaient de gérer des véhicules de location. Nous souhaitons revoir les prix de location pour les cars, ainsi que la numérotation des véhicules.

```
classe Vehicule
 champs      ...privé dateAchat: date, privé prixAchat: nombre, privé
numeroImmatriculation: chaine
 public new (m: chaine, d: date, p: nombre,n: chaine, pe: chaine)
       modele<-m, dateAchat<-d, prixAchat<-p, numeroImmatriculation<-n, permis<-pe
 public age():nbe...
 public coutLocation(): nombre si self<=age() < 365 alors Retourner prixAchat / 200
                            sinon Retourner prixAchat / 250
classe Camion sous classe de Vehicule...
classe Autocar sous classe de Camion
 champs privé places: nombre
 public coutLocation(): nombre
      si places > 40 alors Retourner super.coutLocation() + 50
      sinon Retourner super.coutLocation()
```

1. Il est demandé d'effectuer un changement dans le calcul du coût de location des autocars de plus de 40 places. Jusqu'à présent le coût de location d'un tel véhicule était majoré de 50 € par rapport au calcul classique du coût de location. Le défaut de la méthode que nous avions écrite est que la valeur 50 est présente directement dans le source de la méthode : si on désire modifier cette valeur, il faut modifier le code des méthodes.
    a. La majoration est donnée par le programmeur une fois pour toute, mais pas dans le code d'une méthode. Fournissez votre solution et réécrire `coutLocation()`.
    b. Il est maintenant demandé de pouvoir modifier la valeur de la `majoration`, par une méthode `fixerMajoration(_)`. Fournissez votre solution.
2. Le numéro d'immatriculation permet de repérer chaque véhicule de façon unique, mais il n'est pas très facile d'emploi car il s'agit d'une chaîne de caractères. Il serait intéressant d'avoir, pour chaque véhicule géré par notre application, un numéro d'`identification`, repérant ce véhicule de façon unique (1, 2,...). Le constructeur du `vehicule` affecterait automatiquement la valeur (non utilisée jusqu'à présent) de l'identification et sauvegarderait l'identification du véhicule suivant (`identificationNonUtilisee`); la numérotation des véhicules est alors effectuée automatiquement. Définir le champ et la variable nécessaires dans la classe `Vehicule`, et reprogrammer le constructeur de la cette classe.
3. Proposez une méthode de classe `RAZ` qui remette à zéro (ou 1) la variable de classe précédente. Donner un exemple d'appel de cette méthode dans le programme principal.
4. On veut éviter que les combi (camionnettes à neuf places) soient considérées comme une sorte d'autocar; aucune classe ne doit hériter d'autocar. Programmer cette contrainte.

Exercice 4: Tableau trié d'entiers

1) Définir une classe `TriSimple` permettant de gérer un tableau trié d'entiers (ordre croissant). La classe aura en champ privé le tableau d'entiers `tab`. La capacité initiale du tableau (le nombre de places) sera stockée dans un champ privé `capacité` qui pourra être

fournie par le constructeur (à programmer). Le nombre d'éléments courants dans le tableau est stocké dans un champs `nbElts`.

2) Ecrire les méthodes:

`public void inserer(int entier)` qui insère un entier dans le tableau en respectant l'ordre croissant des éléments

`public void supprimer(int entier)` qui retire un entier donné du tableau

3) Lorsque le tableau devient trop petit pour l'insertion, il est agrandi: en fait on crée un tableau plus grand où on recopie un à un les valeurs et ce tableau remplace l'ancien. La différence entre la taille de l'ancien tableau et la taille du nouveau sera définie par l'ajout de 100 places. Réécrire (compléter) la méthode `inserer`

1) `oueur2` si `score1` est superieur à `score2`.