-

Programmation objet — TD 1
Classes, objets, héritage

Exercice 1

1. Une classe et un objet sont-ils deux choses identiques ?
2. Dire qu'une classe hérite d'une autre classe, est-ce que cela signifie que la première classe a plus d'champs que la seconde ?
3. Dire qu'une classe *C* hérite d'une classe *C'*, est-ce que cela veut dire qu'un objet instance de *C* est aussi instance de la classe *C'* ?
4. Si deux classes ont des champs communs, vaut-il mieux alors créer une troisième classe dont elles hériteront et qui factorise ces champs ?
5. Une méthode est-elle identique à une fonction en programmation classique (Pascal)?

Exercice 2

On prendra des noms " longs " pour les classes, les champs et les méthodes de sorte qu'ils soient explicites : par exemple `ProduitPerissable`

On veut programmer une application gérant les produits dans un magasin.

Les produits sont de divers types. Ils sont caractérisés par leur prix et leur nom, ainsi que pour l'habillement par la taille le pays de fabrication et le type de tissu, pour les produits périssables la date de fabrication, le nombre de jours de conservation et le fait d'être réfrigéré ou non. Il existe aussi des habits de mode, la mode étant caractérisée par la saison et l' année.

1. Dessiner l'arbre des classes, avec les noms des classes, des champs, et les liens d'héritages
2. Ecrire (en LOLO ou en Java) les définitions des classes (champs et liens d'héritage)

Exercice 3

Nous désirons développer un programme (en LOLO) pour la gestion d'un parc de `véhicules` destinés à la location. Pour chaque véhicule, il est demandé de représenter le nom du `modèle`, la `date d'achat`[1], le `prix d'achat`, le `numéro d'immatriculation` et le `permis` nécessaire à la conduite de ce véhicule.

Plusieurs types de véhicules sont disponibles à la location, certains ayant des caractéristiques particulières, ainsi, il est demandé de représenter une information supplémentaire pour les `voitures` de tourisme : la présence ou l'absence d'`autoradio`.

Un autre type de véhicule à définir est le camion (ou `utilitaire`): il est demandé, pour tous les camions, de représenter le `volume` de stockage utile du camion. Enfin, des `autocars` peuvent aussi être loués, et pour ces véhicules, il est nécessaire de représenter le `volume` de stockage (utilisé pour les bagages) ainsi que le nombre de `voyageurs` que chaque autocar peut contenir.

A. Modélisation des données

1. Dessiner l'arbre des classes, avec les noms des classes, les champs, et les liens d'héritages.
2. Ecrire les définitions des classes (hors méthodes).

B. Codage des méthodes

1. Ecrire des méthodes `ajouterAutoradio()` et `enleverAutoradio()` qui permettent de fixer la valeur du champ `autoradio` des voitures de tourisme.
2. Ecrire une méthode `age()` qui retourne l'age du véhicule (le nombre d'années). On suppose avoir une fonction `CurrentYear()` qui fournit en numérique l'année courante.

[1] Nous représenterons la date d'achat par l'année d'achat.

-

3. Ecrire une méthode `peutTransporterVolume`, qui à partir d'un volume donné retourne vrai si le véhicule peut permettre le transport de ce volume et faux sinon. Cette méthode est à programmer dans la classe `Utilitaire`.
4. Ecrire une méthode `coutLocation` calculant le coût quotidien de location : pour les véhicules de moins d'un an, le coût de location est le prix d'achat du véhicule / 200, et pour les véhicules plus anciens, le coût est le prix d'achat / 250
5. Ecrire, dans la classe `autocar`, une méthode `peutTransporterPassagers`, qui à partir d'un nombre de passagers et d'un volume par passager (correspondant au volume moyen de bagages par passager), retourne vrai si le véhicule peut transporter ce nombre de passagers accompagnés de leurs bagages et faux sinon.
6. Ecrire une méthode `afficher()`, permettant d'afficher à l'écran le modèle et le numéro d'immatriculation du véhicule; en LOLO, on suppose que la méthode `écrire`() est disponible pour toutes les classes[2].

C. Codage du programme Test

7. Ecrire un programme de test, créant une voiture `V1` de tourisme `Twingo`, achetée en `2005 10000` €, immatriculée `1234 TZ 49`, avec `autoradio` et nécessitant un permis `B`.
   Ajouter à la suite l'appel à la méthode `afficher` sur cet objet. Quel est le résultat sur l'écran ?
8. Compléter le programme de test pour créer un camion `CA1` de type `J9`, acheté `20000` € il y a 5 ans, immatriculé `2345 FA 49`, nécessitant un permis `B`, et permettant de transporter `5` m^3.
   Afficher le coût de location de ce véhicule, et afficher le fait qu'il permet ou non de transporter 4 m^3 (grâce à un appel à la méthode `peutTransporterVolume`*)*.
   Afficher le coût de location de la Twingo créée précédemment.
9. Modifier le programme de test pour créer un autocar `AU1` de type `FR1`, acheté `90000` € il y a 2 ans, immatriculé `3456 IJ 49`, nécessitant un permis `D`, permettant de transporter `53` passagers et possédant un coffre de `3` m^3. afficher si ce véhicule permet de transporter `40` passagers ayant chacun `0,1` m^3 de bagages.

Exercice 4

On souhaite définir des classes permettant de représenter des `figures` géométriques: `rectangle`, `carré`, `cercle`. Chaque figure est caractérisée par la position de son `centreX` et `Y.`, ses dimensions (`longueur`, `largeur`, `diamètre`, …) et on souhaite pouvoir calculer le `périmètre` (ou `circonférence`), la `surface` de chaque figure.

1. Proposer une modélisation en classes pour ces figures.
2. Définir les méthodes `surface`, `circonférence`…
3. Définir une méthode `afficher` pour chaque classe; elle affiche les intitulés et les contenus des champs concernés par la classe, par exemple pour le cercle la position de son centre et son diamètre.

[2] en Java la méthode `System.out.println(arg)` permet d'afficher la valeur de son argument `arg` (plus de détails en TP).

-

Exercice 5

On veut écrire un programme qui simule une `partie de dé` entre 2 joueurs. Le lancement du dé sera obtenu par un tirage aléatoire d'un nombre entre 1 et 6. On suppose l'existence d'une fonction `aleat(x y)` qui retourne un nombre aléatoire entre x et y.

Les `joueurs` qui sont caractérisés par leur `noms` lancent le dé chacun leur tour, *n* fois (*n* est un paramètre qui caractérise le `nombre de coups` d'une partie). Le `gagnant` est celui qui totalise le plus fort `score` à l'issue des *n* lancers.

Proposer une modélisation en classes faisant intervenir deux classes.

Programmer ces classes.

Proposer un programme de test avec deux joueurs et une partie de 4 coups.