Master Informatique – 1ère année
Université Paris Descartes
F. Cloppet / D. F. Gonzalez Obando

TD Langage C++ n°4
Héritage - Polymorphisme

Exercice 1:
Imaginons que l'on souhaite écrire un programme de gestion globale d'une UFR d'université, s'occupant à la fois des étudiants et du personnel. Les étudiants ont chacun une moyenne annuelle. Tout membre du personnel a un numéro de bureau. Dans le personnel, on distingue le personnel administratif du personnel enseignant. Chaque enseignant détient un numéro de casier. Tout membre du personnel reçoit un salaire à la fin du mois. Cependant, les vacataires, qui font partie du personnel enseignant, sont payés à l'heure et n'ont donc pas de salaire mensuel fixe. Chaque personne est désignée par un nom et une adresse.

1) Donner le graphe d'héritage qui permet de modéliser ce problème.

Exercice 2 :
Lire l'ensemble du texte relatif à l'exercice, et dessiner le graphe d'héritage avant de commencer à programmer. Sur le graphe d'héritage vous représenterez les attributs et les méthodes propres à la classe.

a) Créer une classe générale regroupant des informations relatives à tout animal. Cette classe mère sera appelée la classe Animal.
Chaque animal sera caractérisé par un entier age et une *chaîne* de caractères (string) *nom_du_cri*.

- Ajouter un constructeur sans paramètre (qui demande à l'utilisateur de saisir les valeurs initiales de ces attributs), un constructeur à deux paramètres (avec la liste d'initialisation des champs), un constructeur de recopie et le destructeur par défaut.
- Définir les méthodes d'accession et de modification des attributs
- Déclarer et définir une méthode ***vieillir*** qui augmente d'une unité l'âge.
- Déclarer et définir une méthode ***presenter*** qui affiche l'age et le type de cri de l'animal (aboiement, hennissement, roucoulement, ...) avec un message du type :
  « L'animal a xx ans et son cri est yy."

Mettre un message qui s'affichera à l'écran dans les constructeurs et le destructeur, de manière à pouvoir suivre la création et la destruction des objets.

b) Définissons maintenant une classe fille Chien qui hérite publiquement de la classe Animal. Nous supposerons et admettrons ici que l'aboiement du chien dégage différentes sonorités : " waaf" ou "ouaf" ou encore "grrrh".

1. Ajouter à la classe fille un champ cri et les constructeurs et destructeur habituels (avec un message qui s'affichera lors du passage dans ces constructeurs et destructeur),
2. Redéfinir dans cette classe fille la méthode présenter. Elle présentera l'animal différemment suivant son âge.
S'il est jeune (moins de 6 ans) : "Le chien a xx ans et aboie : nom_du_cri nom_du_cri nom_du_cri."
Sinon : "Le chien a xx ans et aboie : nom_du_cri."
Dans le corps de votre main, définir un objet de la classe Chien et lui appliquer la méthode presenter. Constater quelle méthode est appelée. Essayer ensuite avec Animal ::presenter().
Déclarer un objet Animal chouchou et définir ensuite un objet Chien rex avant d'essayer d'exécuter chouchou = rex.. Essayez aussi l'inverse. Discuter !
Modifier le code nécessaire pour obtenir le résultat attendu.

Tester l'exécution du code ci-dessous :

```
Animal *c;
c = new Chien(4, "aboiement", "rrrrh");
delete c;
```

Que pensez vous du résultat ? Proposer une modification du code pour que le résultat soit correct.