TD 4 : HERITAGE SIMPLE

Objectifs du TD :

- Comprendre la notion d'héritage.
- Savoir utiliser `private`, `protected` et `public`.
- Comprendre l'héritage des méthodes.
- Comprendre l'héritage des constructeurs.
- Faire une première approche de `virtual`

1 La classe mère `animal`

- Déclarez une classe `Animal` qui a deux champs - un entier `age` et une chaine de caractères `nom_du_cri` -, le constructeur par défaut (qui ne fait rien de particulier pour l'instant), le constructeur à deux paramètres (avec liste d'initialisation), le constructeur de copie et le destructeur par défaut. Les méthodes sont déclarées `public` et les champs `protected`.
- Définissez les méthodes ci-dessus.
- Déclarez et définissez une méthode `vieillir` qui augment l'age de 1.
- Déclarez et définissez une méthode `affiche` qui affiche : "L'animal a `age` ans et `nom_du_cri`." (nom du cri pourrait être aboie, hennit, etc...)

2 Une classe fille `chien`

- Un chien est un animal qui aboie, mais peut faire "waaf" ou "woof" ou "grrrh"... Déclarez une classe `Chien` qui hérite publiquement de la classe `Animal`, possède en plus un champ cri, et les constructeurs et destructeur habituels. Les méthodes sont déclarées `public` et les champs `protected`.
- Définissez les méthodes ci-dessus. `nom_du_cri` est initialisé à "aboie" dans la liste d'initialisation du constructeur à paramètre avec la syntaxe `Chien :Animal(age,''aboie''),`.
- Définissez et déclarez une nouvelle méthode `affiche`. Elle affiche cette fois "Le chien a `age` ans et aboie : `cri cri cri`" si il est jeune (moins de 6 ans) et 'Le chien a `age` ans et aboie : `cri`." sinon (eh oui, il commence à fatiguer).

3 Essais d'utilisation

- Dans votre `main`, définissez un objet de la classe `Chien` et appliquez-lui la méthode `affiche`. Laquelle est utilisée (celle de `Animal` ou celle de `Chien`) ? Essayez ensuite avec `Animal : :affiche()`.
- Définissez un objet `Chien` c, déclarez un objet `Animal` a et essayez d'exécuter `a=c`. Essayez l'inverse. Que comprenez-vous ?
- Essayez de déclarer maintenant `vieillir` en `public` ou `protected`, et de déclarer l'héritage `private` ou `protected`. Repérez si un champ ou une méthode est accessible en fonction du type de déclaration dans la classe mère et du type d'héritage.

4 Etude des constructeurs/destructeurs

- Modifiez vos constructeurs et destructeurs en faisant afficher `Animal créé`, `Animal mort`, `Chien créé`, `Chien mort` selon les cas.
- Déclarez alors un objet de la classe `Chien` dans le `main`. Commentez le résultat obtenu.
- Testez léxécution de :

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

delete c; }
```

  Quelquechose vous choque ?
- Ajoutez `virtual` devant la déclaration du destructeur de `Animal` et rééssayez. A quoi sert `virtual` ici ?

5 Une autre classe fille ?

Ecrire une classe `Chien_d_aveugle` qui hérite de la classe
tt Chien et contient en plus une méthode `apte` qui renvoie le booléen `True` si le chien est en âge de travailler (entre 3 et 8 ans par exemple) et `False` sinon.