Université Paris Diderot **Langages à objets avancés**
M1 **Année 2015-2016**

TP n° 8 : Héritage – templates

Les centauriens : suite

Exercice 1 A des fins de contrôle de la population, les Centauriens sont numérotés dès la naissance par un numéro unique. Implémentez ce mécanisme.

Exercice 2 La reproduction est une affaire sérieuse. Pour les Centauriens, elle est aussi des plus bizarre. Il faut que trois centauriens de plus de 18 glups se réunissent dans un ascenceur. Celui qui invite les deux autres donne son nom au nouveau-né, et le sexe du bébé dépendra du sexe de celui qui a initié la reproduction de la facon suivante :

Par ailleurs, on va maintenant considérer que les centauriens ne vieillissent que lorsqu'ils se reproduisent ou donnent leur nom.

— Si celui qui a pris l'initiative est un Truc, le bébé est un Truc
— Si celui-ci est un Machin alors le sexe du bébé sera celui de la majorité des participants. S'ils sont tous de sexes différents, c'est le Machin qui domine.
— Si celui-ci est un Bidule, alors le sexe du bébé est laissé au hasard.

Réalisez la méthode de reproduction des centauriens. On utilisera parcimonieusement `dynamic_cast` qui permet de déterminer la nature d'une instance dynamiquement : `dynamic_cast<A*>(x)` renverra `nullptr` ssi x ne contient pas un pointeur vers un objet de type compatible avec A. N'utilisez pas cette méthode si l'héritage permet de l'éviter. Comment déclarez-vous cette méthode dans la classe mère des centauriens ?

On utilisera aussi `srand (time(nullptr));` pour amorcer un générateur aléatoire basé sur le temps et `rand()` pour obtenir un nombre aléatoire entre 0 et `RAND_MAX`.

Exercice 3 Les Centauriens forment une société très inégalitaire. Les Trucs "valent plus" que les Machins qui valent plus que les Bidules. À sexe égal c'est le plus vieux qui "vaut plus". Deux individus de même sexe et de même âge seront départagés par leur numéros (le plus petit "gagne").

Redéfinissez les opérateurs "<" pour tenir compte de ce fait. (Il faudrait bien sûr aussi redéfinir les autres opérateurs de comparaison.)

Exercice 4 En dehors de toute classe implémentez une méthode

```
template<class T> T* maximum(T* tab[], int longueur)
```

qui retourne le plus grand élément d'un tableau en utilisant l'opérateur "<".

Exercice 5 Testez cette méthode sur un tableau de pointeurs sur Centauriens.