1 TD LOTS numéro 1

Ce TD est censé permettre de construire les briques de base permettant de faire les TD suivants, en JAVA, C++ et autres langages...

1.1 Classe de base

Définir une classe `Racine`, munie
- d'un attribut `id`, une string ou un entier, qui constitue un identifiant des objets ;
- d'un constructeur qui affecte automatiquement `id` ;
- d'une méthode `whoamI`, sans paramètre ni valeur de retour, qui affiche l'identifiant du receveur courant ainsi que son type dynamique.

Vous avez le choix pour l'identifiant, mais il faut qu'il soit non ambigu. Pour récupérer le type dynamique d'un objet, il y a différentes solutions. Dans le pire des cas (et des langages), il suffit de redéfinir la méthode `whoamI` dans chaque sous-classe...

1.2 Sous-classes

Définir ensuite deux familles de sous-classes :
- les classes $A, B, C, ..$, avec $C \prec B \prec A \prec$ `Racine` ;
- les classes $T, U, V, ..$ avec ou sans relation entre elles.

Dans A définir des méthodes `foo`, `bar` qui prennent un paramètre de type T, U, ou V.

Redéfinir ces méthodes dans les sous-classes de A.

Chaque méthode doit afficher la classe dans laquelle elle est définie, ainsi que le type dynamique du receveur et du paramètre au moyen de la méthode `whoamI`.

1.3 Main

Faire une procédure principale, déclarant des variables nommées ab, où a et b parcourent $A, B, C, ..$, de telle sorte que a désigne le type statique de la variable et b le type dynamique de sa valeur. On écrira par exemple `B bc=new C();`.

Faire de même pour les classes $T, U, V, ...$

Appeler ensuite les méthodes `foo`, `bar`, etc. sur les différents receveurs de type A (ou sous-type de), en leur passant des arguments de type T, ou sous-type.

1.4 En C++

Vérifier que le code C++ vérifie les consignes données dans le polycopié au chapitre sur le typage statique.