Problème (15 Points)

Raconter des histoires aux enfants est un véritable métier. Nous vous proposons de coder un outil en C++ pour aider un conteur à trouver parmi tous les contes qu'il connaît ceux où apparaît tel ou tel personnage (toutes les histoires de « loup » par exemple).

Le code devra être écrit pour minimiser la copie d'objets et respecter au maximum la constance des objets. Il faut préciser également les entêtes à inclure pour chaque objet prédéfini et la manière d'y accéder.

1/ Créer une classe **Personnage** avec un attribut privé **nom** de type chaîne de caractères. Mettre un constructeur public qui permet d'initialiser l'attribut avec une liste d'initialisation. Si vous définissez d'autres attributs, mettre des valeurs par défaut. On veut ne pas avoir la possibilité de créer de Personnage avec un nom par défaut.

Si vous voulez ajouter des attributs (on ne vous le demande pas), on peut considérer qu'un **Personnage** est gentil ou méchant (avec un booléen), et qu'il est plus ou moins intelligent. Sur une échelle de 0 à 10, le loup des trois petits cochons aura une intelligence de 5 (valeur par défaut) et les trois petits cochons, respectivement 3, 5 et 9.

2/ Créer une classe **Animal** qui dérive de **Personnage** et qui permet d'initialiser le **nom**.
Les autres attributs possibles (comme par exemple la faculté parler) ne nous intéressent pas.

3/ Créer une classe **Conte** avec un constructeur qui permet d'initialiser le **nom** du **Conte**. Cette classe doit pouvoir stocker des pointeurs sur des objets de la classe **Personnage**. Vous utiliserez un conteneur de la stl.

3pts 4/ Ecrire le code nécessaire pour réaliser les tâches suivantes :

```
Conte tpc("trois petits cochons");
tpc.add(new Animal("loup"));
Conte pcr = tpc;
pcr.setNom("petit chaperon rouge");
cout << tpc << endl;
cout << pcr << endl;
```

La redirection d'une instance de **Conte** sur un flux affiche le **nom** du conte ainsi que le **nom** des différents personnages.
Les personnages d'un conte sont dupliqués si un conte est dupliqué. Aucune fuite mémoire ne peut être acceptée.

5/ On veut vérifier que l'instance d'**Animal** a bien été dupliquée par l'instruction :

```
cout << Personnage::getCompteur();
```

Donner les modifications à apporter (et où) pour prendre en compte ce besoin.

6/ Définir une méthode `decrire()` pour la classe **Personnage**. Cette méthode permet d'afficher les propriétés des personnages sur un flux donné en paramètre. Bien entendu, cette méthode devra être redéfinie dans les sous-classes de **Personnage.**

7/ Comment rendre la classe **Personnage** abstraite ? Comment limiter la duplication de code dans les classes filles ?

On suppose que la classe **Humain** a été écrite. Un **Humain** est un garçon ou une fille et peut avoir un âge stocké dans une énumération : JEUNE, ADO, ADULTE, VIEILLARD.

8/ On veut maintenant traiter le cas d'un **Monstre** qui un mélange subtil d'**Humain** et d'**Animal**. Quel concept faut-il utiliser et comment permettre l'exécution du code suivant en ne créant qu'une seule instance de Personnage pour la "Bete" ? Si vous devez modifier du code écrit précédemment, veuillez juste donner les modifications.

```
Conte bb("la Belle et la Bete");
bb.add(new Humain("Belle"));
bb.add(new Monstre("Bete"));
```

Là encore, les attributs de **Monstre** ne nous intéressent pas mais on pourrait prendre en compte l'aspect dans une énumération : COMIQUE, MIGNON, INSIGNIFIANT, TERRIFIANT

9/ Créer une classe **Conteur** qui mémorise des pointeurs sur des **Conte** et stocke un index des noms de personnages avec le conte associé. L'index sera une **multimap** associant une chaîne de caractères avec un pointeur sur le conte concerné. Le conteur aura également un **nom** défini à l'instanciation.

10/ Implémenter une méthode pour créer l'index à partir de la liste des contes.

11/ Implémenter une méthode qui permet d'afficher l'index sur un flux donné préférentiellement avec un algorithme de la STL.

12/ Implémenter une méthode qui affiche tous les contes où un personnage donné par son nom apparaît.

13/ Les classes **Conte** et **Conteur** sont similaires (attribut **nom** pour chacun et une collection). Comment limiter la duplication de code ? Justifier (sans écrire tout le code).