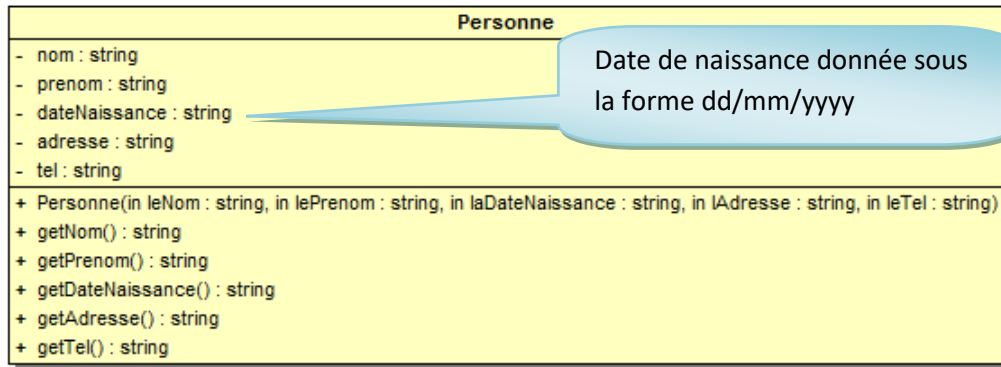
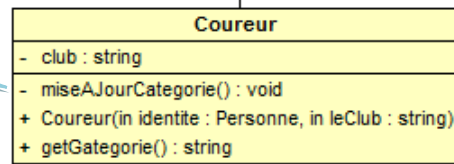


Exercice sur l'héritage

En vous inspirant du diagramme UML ci-dessous, créer le codage des classes `Personne` et `Coureur` et ajoutez les méthodes nécessaires pour construire les objets et récupérer les valeurs des attributs.



Mise à jour de la catégorie en tenant compte de la date de naissance du coureur et de la date courante.



Les catégories seront déterminées **automatiquement** en fonction de la date de naissance du coureur et de la date du jour.

Les catégories seront stockées dans une énumération. Voici la liste des catégories :

CATEGORIES : dépend de l'âge de la personne au 1er janvier de l'année courante.

- École d'Athlétisme : 9 ans et moins
- Poussin : 10-11 ans
- Benjamin : 12-13 ans
- Minime : 14-15 ans
- Cadet : 16-17 ans
- Junior : 18-19 ans
- Espoir : 20-22 ans
- Senior : 23-39 ans
- Vétéran 40 ans et plus
 - Les catégories vétérans sont par tranche de 10 ans, de V1 à V4

Créer les classes, leur connexion et un programme principal de test avec Visual Studio (mode console). Vous créerez plusieurs coureurs d'âges différents et afficherez toutes les informations les concernant. Vous prendrez notamment soin de bien vérifier leur catégorie. Attention, il faut que ce programme, sans aucune modification de code puisse continuer à fournir la bonne catégorie dans x années.

ANNEXE 1 : Récupération de la date courante

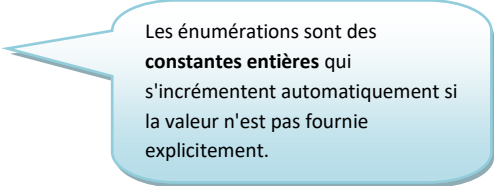
Sous Visual Studio, vous pourrez utiliser la classe CTime (inclure `<atlttime.h>`)

Vous regarderez l'aide de Visual Studio sur cette classe et essaierez de trouver la méthode qui permet de créer un objet de cette classe en l'initialisant à la date courante.

ANNEXE 2 : Utilisation des énumérations

Quand on a un nombre limité de choix, il est très intéressant d'utiliser des énumérations car contrairement aux chaînes de caractères, elles suppriment tous les problèmes liés à la saisie, à la casse, etc.

Inspirez-vous de cet exemple pour votre codage

	Cuisine.h
<pre>#pragma once // Une énumération correspond à une série de nombres (commençant à 0 par défaut) auxquels on attribue un nom enum LEGUMES { CHOU, // CHOU VAUT 0 NAVET, // NAVET VAUT 1 POMME_DE_TERRE, // VAUT 2 POIS=7, // VAUT 7 HARICOT, // VAUT 8 } ; class CCuisine { private: // Définition d'un attribut qui ne pourra prendre comme valeur que l'un des légumes définis LEGUMES accompagnement; public: CCuisine(void); };</pre>	
<pre>#include "Cuisine.h" CCuisine::CCuisine(void) { this->accompagnement = LEGUMES::HARICOT ; }</pre>	Cuisine.cpp