UE : Programmation Orientée Objet (I4AEIL11)
Sujet de TD

Institut National des Sciences Appliquées de Toulouse
Yannick Pencolé, yannick.pencole@laas.fr

—

TD: premières classes (2-3 séances)

(12 février 2024)

1 Configuration de Netbeans

Tout au long des TD/TP/examen de POO, nous allons utiliser un environnement de développement intégré (IDE = *integrated development environment*). Ce logiciel s'appelle *Netbeans* et est disponible à l'INSA sous linux à partir des menus. *Netbeans* est un logiciel qui permet d'éditer, compiler et exécuter des projets logiciels développés en différents langages. Avec ce type de logiciel, il est inutile de gérer des Makefile, tout est géré automatiquement. Nous l'utiliserons exclusivement pour compiler du C++. Lors de sa configuration, *Netbeans* détectera automatiquement le compilateur C++ du système (ce compilateur s'appelle `g++` et étend le compilateur `gcc` pour le langage C++).

À la première utilisation de *Netbeans*, il faut activer une extension C++ (*plugin*) sur votre compte. Après avoir lancé *Netbeans*, suivez les étapes suivantes :

— Aller dans le menu *Tools* > *Plugins* > *Settings*
— Activer les options en cochant "*Netbeans 8.2 plugins Portal*"
— Aller dans *Available* et rafraîchir (*Check for newest*)
— Choisir C/C++ et valider les étapes

Ceci ne se sera à faire qu'une seule fois.

2 Un premier projet en C++

1. Commencez par lancer Netbeans et créer un projet C/C++ de type *C/C++ Application* avec le nom que vous voulez (mais quelque chose de reconnaissable est souhaitable),
2. **Attention** lors de la création du projet cochez bien `Create Main File` et choisissez son type comme étant C++ 14,
3. Regardez le fichier `main.cpp` qui a été généré automatiquement dans `Sources Files`,

4. Ajoutez du code pour afficher le message `Bonjour le monde !` avec un saut de ligne en utilisant les opérateurs entrées/sorties de la bibliothèque standard du C++
5. Compilez, exécutez, observez,
6. Supprimez la ligne `using namespace std;` du fichier `main` et refaites la même chose. Conclusion : À quoi sert cette ligne `using namespace std;` ?

3 Ma première classe

1. Créez un fichier d'entête (clique droit sur le projet, new→C++ *header File*) avec le nom `Personnage.h`),
2. Regardez ce que *Netbeans* a généré automatiquement pour vous.
3. Déclarez une classe `Personnage` qui contient trois attributs représentant les nom, prénom et âge du personnage. Pour les attributs de nom et prénom, vous choisirez des variables du type `std::string` et pour l'âge, un `unsigned int`. Le paradigme de la programmation orientée objet impose que ces trois attributs doivent être à **accès privé**.
4. Créez un fichier source en C++ (clique droit sur le projet, *new→C++ source File*) avec le nom `Personnage.cpp`.
5. Ajoutez un constructeur par défaut à **accès publique** dans lequel vous mettrez par défaut les valeurs à `John Doe 20` aux attributs du personnage par défaut.
6. Ajoutez à la fin du constructeur l'affichage du message : (pour l'affichage, utilisez `std::cout` qui est déclaré dans `iostream`)

   `Le personnage [prenom] [nom] est né et il a déjà [age] ans.`
7. Ajoutez un destructeur dans lequel vous afficherez simplement un message

   `RIP [prenom] [nom]`
8. Ajouter dans le `main` la déclaration d'une instance de personnage. Par exemple :

   ```
   Personnage johnDoe;
   ```

9. Compilez, exécutez, observez et expliquez ce qui se passe. Quelle ligne de votre programme appelle le constructeur du personnage ? Quelle ligne de votre programme appelle le destructeur ?

4 Accesseurs et Modificateurs

1. Supposons que le nom de votre attribut âge dans votre classe `Personnage` soit `_age`, dans votre programme `main`, après la création d'un `Personnage johnDoe;` , écrire l'instruction suivante :

`std::cout << "age du personnage: " << johnDoe._age << '\n';`
Est-ce que ce code compile ? Pourquoi ?

2. Déclarez l'**accesseur** `unsigned int age();` dans la classe `Personnage` (dans le fichier `Personnage.h`) puis définissez-le dans `Personnage.cpp`. C'est un accesseur, il doit donc être avec un **accès publique**. Puis dans votre `main`, modifiez la ligne précédente comme suit :
   `std::cout << "age du personnage: " << johnDoe.age() << '\n';`
3. Toujours après la création de `johnDoe`, écrire la ligne suivante :
   `johnDoe._age = 40;` Est-ce que ce code compile ? Pourquoi ?
4. Déclarez le **modificateur/mutateur** `void changeAge(unsigned int nouvelAge);` dans la classe `Personnage` (dans le fichier `Personnage.h`) puis définissez-le dans `Personnage.cpp`. Il doit également être avec un **accès publique**. Puis dans votre `main`, modifiez la ligne précédente comme suit :
   ```
   std::cout << "age du personnage: " << johnDoe.age() << '\n';
   johnDoe.changeAge(40);
   std::cout << "age du personnage: " << johnDoe.age() << '\n';
   ```
5. De façon similaire, mettez en œuvre les accesseurs et les modificateurs pour les attributs nom et prénom et testez-les. Voiçi un exemple de test :
   ```
   Personnage spiderMan;
   spiderMan.changeAge(15);
   spiderMan.changeNom("Parker");
   spiderMan.changePrenom("Peter");
   std::cout << "Nom: " << spiderman.nom() << "Prénom: " << spiderman.prenom()
   << " Age: " << spiderman.age() << '\n';
   ```

5 Surcharge de constructeur

Pour le moment, nous avons défini un constructeur (celui par défaut). Maintenant, nous allons en définir deux autres. Autrement dit, on va **surcharger** le premier constructeur.

1. Ajoutez un nouveau constructeur à la classe `Personnage` qui prend comme paramètres `string prenom`, `string nom`. L'attribut d'âge sera initialisé par défaut. Testez-le dans le `main`. Par exemple :
   ```
   Personnage ironMan("Tony","Stark");
   std::cout << "Nom: " << ironMan.nom() << "Prénom: " << ironMan.prenom()
   << '\n';
   ```

2. Ajoutez un nouveau constructeur à la classe `Personnage` qui prend comme paramètres `string prenom`, `string nom` et `int age`. Testez-le de façon similaire.

3. Écrire le constructeur par copie `Personnage(const Personnage & pers)`. Le constructeur par copie doit initialiser les attributs de nouvel objet avec une copie de tous les attributs l'objet `pers`. Utilisez-le dans le `main` pour le tester. Par exemple,

```
Personnage georgeEn1985("Georges","MacFly",47);
Personnage georgesEn1955(georgesEn1985); // constructeur par copie
georgesEn1955.changeAge(47-(1985-1955));
std::cout << "En 1985, Nom: " << georgeEn1985.nom()
 << "Prénom: " << georgeEn1985.prenom()
 << georgeEn1985.age() << '\n';
std::cout << "En 1955, Nom: " << georgeEn1955.nom()
 << "Prénom: " << georgeEn1955.prenom()
 << georgeEn1955.age() << '\n';
```

6 Allocation dynamique d'objets `Personnage`

Jusqu'à présent, les personnages ont été construits en déclarant une variable de type personnage avec d'éventuels paramètres (en fonction des constructeurs disponibles), comme par exemple :

```
Personnage spiderMan("Peter","Parker",15);
```

C'est une allocation statique. L'objet `spiderMan` sera détruit automatiquement à la fin de la fonction/méthode où il a été déclaré.

1. Créez un nouveau Personnage en utilisant l'allocation dynamique (utilisation de `new`)
2. Changer le nom du personnage et afficher-le à l'écran.
3. Tout objet alloué dynamiquement par `new` doit être systématiquement détruit quand il devient inutile avec `delete`. Insérez l'instruction de destruction de votre personnage à l'endroit que vous jugez utile.

IMPORTANT : Notez bien que si vous créez un objet avec `new` dans une fonction ou une méthode quelconque, il ne sera pas automatiquement détruit à la fin de l'exécution de cette fonction contrairement à une allocation statique. Tout objet créé par `new` doit systématiquement et explicitement être détruit à l'aide d'un `delete`, soit dans la fonction qu'il l'a créé, soit dans une autre fonction utilisée ultérieurement (par exemple dans un destructeur). Si vous omettez d'utiliser le `delete`, votre programme produira des fuites de mémoire et avoir un comportement instable (ou pas) et il deviendra très difficile à corriger.

7 Enumérations

1. Créez une énumération `Genre` dans le fichier `Personnage.h` qui prend pour valeur `MASCULIN` ou `FEMININ` (voir fiche du C).

2. Ajoutez à la classe `Personnage` un nouvel attribut stockant le genre du personnage.
3. Modifiez les 2 constructeurs pour prendre en compte ce nouvel attribut
4. Mettez à jour le message des constructeurs pour gérer l'accord de `né` en fonction du genre (sans écriture inclusive).

8 Mise en œuvre de méthodes

1. Ajouter une méthode `std::string identite()` qui retourne la chaîne de caractère `Salut, je m'appelle [prénom] [nom] et j'ai [age] ans.`
2. Utiliser la méthode `identite` depuis le `main`.
3. Ajoutez une méthode `bool estVieux()` qui retourne vraie si le personnage a plus de 80 ans.
4. Ajoutez une méthode `bool estPlusVieuxQue(Personnage * pers)` qui retourne vraie si le personnage courant est plus agé que le personnage `pers`.
5. Testez ces nouvelles méthodes dans le `main`.

9 La classe Histoire

Nous allons maintenant écrire une classe `Histoire` dont l'objectif est de recenser les personnages de cette histoire.

1. Créer les fichiers `Histoire.h` et `Histoire.cpp`.
2. La classe `Histoire` devra stocker le titre de l'histoire (par exemple `retour vers le futur`, `martine à la plage`, `starwars`...) et des pointeurs sur des objets `Personnage`. Pour stocker ces pointeurs on utilisera un tableau de type `vector<Personnage *>`. Écrire les attributs de la classe `Histoire`.
3. Écrire une méthode privée `void stockePersonnage(Personnage * p)` qui stocke le personnage dans l'histoire.
4. Écrire une méthode publique `Personnage * creerPersonnage(nom,prenom,age,genre);` qui créé un personnage (allocation dynamique) et le stocke dans l'histoire.
5. Écrire un constructeur par défaut (titre vide, pas de personnage).
6. Écrire une méthode publique `long nombreDePersonnages()` qui retourne le nombre de personnages dans l'histoire.
7. Écrire les accesseurs/modificateurs pour le titre de l'histoire.
8. Écrire la méthode publique `afficherPersonnages(int age)` qui affiche le prénom et le nom de chaque personnage de l'histoire dont l'âge est inférieur à `age`.
9. Écrire une méthode privée `effacerTousLesPersonnages()` qui détruit tous les personnages de l'histoire.

10. Ecrire le destructeur de `Histoire`. Ne pas oublier de détruire explicitement tous les personnages de l'histoire qui ont été alloués dynamiquement par `creerPersonnage()`.
11. Dans le `main`, créer une histoire avec le titre et les personnages de votre choix. Puis afficher le titre et la liste de vos personnages à l'écran.
12. Écrire un constructeur par copie pour la classe `Histoire`.

Par exemple :

```
Titre de l'histoire: Retour vers le futur
Personnage 1: Marty McFly
Personnage 2: Emmett Brown
Personnage 3: George McFly
```