TD 1 - Classes et Objets

Programmation Orientée Objet

Objectif

— Comprendre ce qu'est une classe ;
— Comprendre ce qu'est un objet ;
— Comprendre la différence entre une classe et un objet ;
— Manipuler un programme objet.

1 Classe HelloWorld

1) Ecrivez un programme C++ affichant "Hello World !".
2) Ecrivez une classe `HelloWorld` dont une méthode affiche le message "Hello World !". Donnez le code du programme principal appelant la méthode de la classe `HelloWorld`.

2 Classe Personne - Premiers pas

1) Créez une classe Personne. `Cette` classe comportera les informations suivantes stockées sous la forme de chaînes de caractères : le `nom` et le `prénom`. Cette classe aura comme opération : `afficher`, `saisir` et `raz`.
2) Ecrire un petit programme d'essai qui affecte tout d'abord les valeurs aux différents champs d'une telle classe, les affiche avant de leur appliquer la fonction `raz`.
2) Ecrire un constructeur et un destructeur pour cette classe. Le constructeur devra renseigner les champs de `personne`. Ecrire un programme principal créant un objet instance de la classe `Personne`.
3) On désire ajouter des accesseurs aux membres privés de la classe. En quoi cela consiste-t-il ? Quel est l'intérêt de cela ? Implémentez des accesseurs sur les membres privées de `Personne`.

3 Classe Voiture - Approche composants

Jusqu'à présent, nous avons réalisé les classes dans un seul fichier source commun avec le programme principal. Cette approche est contraire à l'esprit de la POO. On y préfère implémenter une classe sous la forme d'un composant logiciel, c'est à dire en décrivant la structure de la classe dans un fichier d'entête (`.hpp`) et le code source des fonctions membres dans un fichier de source(`.cpp`). Enfin le programme principal sera implémenté dans un programme source à part. Faire cela pour la classe `Voiture`, en réalisant les fichiers `Voiture.hpp`, `Voiture.cpp` `MainVoiture.cpp`. Attention au problème de la double inclusion !

On considère qu'une voiture est caractérisée par sa marque, son modèle. De plus, elle peut contenir un lien vers ses passagers. Les passagers sont des instances de la classe Personne. Ceux-ci sont au maximum au nombre de 5. Il convient donc de créer également une fonction d'ajout de passagers à la voiture ainsi qu'une méthode `afficherPassager()` affichant le nom de chaque passager. On surchagera pour cela l'opérateur d'affectation de la classe Personne.

Créer les classes et les méthodes permettant la gestion de cette classe `Voiture`.

4 Ensemble

Définir une classe `Ensemble` pour manipuler des ensembles d' "elements" (`#define elements int` dans un premier temps). On utilisera l'allocation dynamique pour manipuler des ensembles de taille quelconque.

1) Donnez la définition de la classe dans un premier temps sans son implémentation. On considère pour le moment que la classe Ensemble ne contient que les opérations `appartient`, `card` et `inserer`.

2) L'utilisation de l'allocation dynamique implique d'initialiser les données membres de la classe lors de sa création. Pour cela, il vous faut modifier le constructeur par défaut, ainsi que le destructeur.

3) Ajouter une méthode `affiche()` pour la classe.

4) Surchargez la méthode `affiche()` de manière à ce qu'elle puisse prendre comme paramètre le rang d'un élément et afficher cet élément.

5) Ajouter un programme principal utilisant la classe `Ensemble`. Créez deux éléments. Donnez leur une valeur. Ajouter les à l'ensemble. Affichez.

6) Modifier l'ensemble des programmes afin de manipuler un ensemble de personnes.