TP de C++ n°5

Composition & Agrégation

L'objectif de ce TP est de concevoir et programmer une classe nommée `Personne` qui permette de créer, modifier et relier entre eux des objets représentant des individus. Ces individus sont dotés d'un nom, d'un prénom et d'une date de naissance.

1. Gestion du nom (relation de composition basique)

Déclarez et implémentez une classe `Personne` pour pouvoir faire fonctionner le programme suivant :

```
void main()

{
	Personne phil( "Philippe Durant" ) ;

	std::string nom = phil.getNom() ;

	cout << " Saisissez un nouveau nom pour " << nom << " : " ;

	std::string nouveauNom = "";

	cin >> nouveauNom;

	phil.setNom( nouveauNom ) ;

}
```

Qu'advient-il de l'attribut utilisé pour représenter le nom de la personne lorsqu'on supprime une instance de cette classe ?

2. Gestion de l'age (relation de composition)

On suppose qu'une personne est dotée d'une date de naissance, de laquelle on peut déduire l'âge à tout moment. Créez une classe `Date` qui encapsule les informations de base relatives à une date (numéro du jour dans le mois, numéro du mois et année) et qui permet d'y accéder via les accesseurs et les mutateurs nécessaires (`get/setJour`, `get/setMois`, `get/setAnnee`). La classe Date disposera également d'un constructeur par défaut qui initialise l'instance à la date du jour.

Complétez ainsi la classe `Personne` avec un attribut nommé `dateNaissance` (date de naissance) de type `Date`. Modifiez le constructeur de la classe `Personne` pour autoriser la création d'une instance à partir du nom, du prénom et de la date de naissance comme le montre le programme principal ci-dessous. Ajoutez également la méthode `getAge()`. qui retoune l'âge (en nombre d'années) de l'individu.

Est-il nécessaire d'implémenter un destructeur pour la classe `Personne` à ce stade ?

```
void main() {

      // Nouvelle personne : Philippe Durant, né le 10/01/1979
      Personne phil("Philippe Durant", 10, 1, 1979);

      int age = phil.getAge(); // récupère l'âge de philippe

      cout << phil. getNom() << " a " << age << " ans. " << endl;

}
```

3. Gestion du conjoint (relation d'agrégation)

On suppose qu'une personne peut être mariée à une autre, mais que cette relation est susceptible d'être rompue (en cas de divorce ou de décès du conjoint par exemple).

Ajoutez les attributs et méthodes nécessaires pour pouvoir marier deux instances existantes de la classe `Personne` et éventuellement leur permettre de changer de conjoint ou de redevenir célibataires !

Doit-on supprimer l'attribut conjoint dans le destructeur lorsqu'une instance de la classe `Personne` est supprimée ? Expliquez en quoi c'est différent de ce qui est fait pour l'instance de la classe `Date` (qui représente la date de naissance).

4. Passage de paramètres

Que se passe-t'il lorsqu'on transmet par valeur un objet de type `Personne` à une fonction ?

Par exemple :

```
void Marier( Personne a, Personne b)
{

      a.epouse(b);

      b.epouse(a)

}

void main() {

      Personne phil( "Philippe Durant", 10, 2, 1978 );

      Personne elo( "Élodie Dupond", 2, 5, 1979 );

      Marier(phil,elo);

}
```

Proposez une solution pour résoudre ce problème.

5. Problème de « réincarnation » (allocations/initialisations implicites et copie)

Que se passe-t'il quand on cherche à affecter un objet de type `Personne` à un autre objet de même type ?

Par exemple :

```
void main() {
      Personne phil( "Philippe Durant", 10, 2, 1977 ) ;

      // Philippe Durant né le 10/2/1977

      Personne albert( "Albert Gontrand", 24, 8, 1907 ) ;

      // Albert Gontrand né le 24/8/1907

      albert = philippe ; // on tente une réincarnation

}
```

Que peut-on faire pour résoudre ce problème ? Faites les modifications nécessaires au programme et/ou à la classe `Personne`.