TP C++ : 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.

Exercice : Gestion du nom (relation de composition basique)

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

```
int main(void)
{
    Personne phil("Philippe","Durand");
    string nom = phil.getNom();
    cout << "Saisissez un nouveau nom pour " << nom << " : " ;
    string nouveauNom;
    cin >> nouveauNom;
    phil.setNom(nouveauNom);

    return 0;

}
```

- Gestion de l'âge (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 du jour de naissance de l'individu. 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 retourne l'âge (en nombre d'années) de l'individu.

```
int main(void)
{
    // 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;
    return 0;

}
```

Gestion du temps en C++ :

- https://en.cppreference.com/w/cpp/chrono/c/localtime
- https://en.cppreference.com/w/cpp/chrono/c/tm

- 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 (méthode : epouse) deux instances existantes de la classe **`Personne`** et éventuellement leur permettre de changer de conjoint (méthode : changerConjoint) ou de redevenir célibataires (méthode : seSepare). Rajouter également une métode getEtat() permettant de savoir si la personne est en couple ou célibataire! 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).

- 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);
}


int main(void)
{
   Personne phil( "Philippe", "Durant", 10, 2, 1978 );
   Personne elo( "Élodie", "Dupond", 2, 5, 1979 );
   Marier(phil,elo);
   return 0;

}
```

Proposez une solution pour résoudre ce problème.

Exemple de programme principal

```
int main()
{
    cout << "Agregation et Composition!" << endl;

    Personne phil("Philippe", "DURANT", 10,2,1978 );
    Personne laure("Laure", "BLANQUART", 10,8,1977 );
    Personne autre("charmante", "Inconnu", 12,7,1977 );
    Personne  *louis;
    louis = new Personne("Louis", "BROCHE",1,1,2000);


    cout << phil.getPrenom() << " " << phil.getNom() << " est " << phil.getEtat() << endl;
    cout << laure.getPrenom() << " " << laure.getNom() << " est " << laure.getEtat() << endl;

    cout << "Mariage !!" << endl;
    Marier(phil,laure);

    cout << phil.getPrenom() << " " << phil.getNom() << " est " << phil.getEtat() << endl;
    cout << laure.getPrenom() << " " << laure.getNom() << " est " << laure.getEtat() << endl;

    cout << "Separation !!" << endl;
    phil.seSepare();

    cout << phil.getPrenom() << " " << phil.getNom() << " est " << phil.getEtat() << endl;
    cout << laure.getPrenom() << " " << laure.getNom() << " est " << laure.getEtat() << endl;

    cout << "Mariage !!" << endl;
    Marier(autre,*louis);
    cout << louis->getPrenom() << " " << louis->getNom() << " est " << louis->getEtat() << endl;
    cout << autre.getPrenom() << " " << autre.getNom() << " est " << autre.getEtat() << endl;


    cout << "Separation !!" << endl;
    louis->seSepare();

    cout << louis->getPrenom() << " " << louis->getNom() << " est " << louis->getEtat() << endl;
    cout << autre.getPrenom() << " " << autre.getNom() << " est " << autre.getEtat() << endl;

    // Changement de conjoint
   cout << "Louis  : Changement de conjoint  avec autre !!" << endl;
    louis->changerConjoint(&laure);

    cout << louis->getPrenom() << " " << louis->getNom() << " est " << louis->getEtat() << endl;
    cout << laure.getPrenom() << " " << laure.getNom() << " est " << laure.getEtat() << endl;
    cout << autre.getPrenom() << " " << autre.getNom() << " est " << autre.getEtat() << endl;


    return 0;
}
```