EFREI
L2 **TP noté C++**

18 Mai 2016

1 Notions de base

L'objectif de cet exercice est de concevoir une classe `Personne` qui contient le nom ainsi que la date de naissance d'une personne, et qui puisse être utilisée avec le programme suivant :

```
void main() {
  // Philippe Durant, ne le 10/01/1979
  Personne phil("Philippe Durant", 10, 1, 1979);
  // recupere le nom complet de M. Durant
  std::string nom = phil.getNom();
  cout << "Nom = " << nom << endl;
  // recupere l'age de Philippe Durant
  int age = phil.getAge(); // recupere l'age de philippe
  cout << "Age = "<< age << endl;
}
```

1. Ecrivez la déclaration et l'implémentation de la classe `Personne` pour qu'elle fonctionne avec le programme suivant :

   ```
   void main()
   {
     Personne phil( "Philippe Durant" ) ;
     std::string nom = phil.getNom() ;
     cout << "nom : " << nom << endl ;
     phil.setNom( "Alphonse Dupont" ) ;
   }
   ```

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

2. On suppose qu'une personne a 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`).

   Complétez ensuite la classe `Personne` avec un attribut `ddn` (date de naissance) de type pointeur vers `Date` (`Date*`) pour représenter la date de naissance. Modifiez le constructeur de la classe `Personne` puis ajouter la méthode `getAge` pour qu'elle fonctionne avec le programme donné en début d'exercice.

   Est-il nécessaire d'implémenter un destructeur pour la classe `Personne` ? Pourquoi ? Si oui que doit-il faire ? Si oui, faites-le !

Note : on supposera que la classe `Date` dispose d'un constructeur par défaut qui initialise l'instance avec la date du jour (on n'écrira pas le code de ce constructeur).

3. 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. 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( "Elodie Dupond", 2, 5, 1979 );
  Marier(phil,elo);
}
```

   Proposez une solution pour résoudre ce problème.

5. 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 ne le 10/2/1977
  Personne albert( "Albert Gontrand", 24, 8, 1907 ) ;
  // Albert Gontrand ne le 24/8/1907
  albert = philippe ; // on tente une reincarnation
}
```

   Que peut-on faire pour résoudre ce problème ? Faites les modifications nécessaires au programme.

2 Héritage et méthodes virtuelles

Héritage

1. Créez une classe `Forme2D` caractérisée par sa position dans l'espace plan et sa couleur (un entier). Ecrivez les constructeurs et destructeurs adaptés si nécessaire. Ecrivez une méthode `affiche()` qui inscrit à l'écran les caractéristiques de l'objet (position, couleur).

2. Créez la classe Rectangle qui dérive de `Forme2D` et qui est caractérisée par sa `largeur` et sa `hauteur`. Redéfinissez la méthode `affiche()` pour cette classe en réutilisant la méthode de la classe de base.
3. Créez la classe `Cercle` qui dérive également de `Forme2D` mais qui est caractérisée, elle, par son `rayon`. Redéfinissez également la méthode `affiche()`.
4. Ecrivez un programme principal qui utilise ces trois classes.

Méthodes virtuelles

1. On souhaite maintenant que toute instance dérivée de `Forme2D` retourne sa surface grâce à une méthode `double surface()` ; Le calcul d'une surface n'étant pas défini pour une forme indéterminée, quelle solution peut-on adopter pour l'ajout de cette méthode dans la classe `Forme2D` ? Que peut-on dire alors de la classe `Forme2D` ?
2. Implémentez la méthode `surface()` pour les classes `Rectangle` et `Cercle`.
3. Créez une classe `Ecran` qui contient un ensemble de pointeurs vers des objets dérivés de `Forme2D` (par exemple sous la forme d'un tableau). Cette classe dispose d'une méthode qui permet d'attacher ou de détacher une `Forme2D` donnée par son adresse ainsi que des méthodes `affiche()` - dont le rôle consiste à afficher chacun des objets enregistrés - et la méthode `surface()` qui donne la surface totale occupée par les formes à l'écran. N'oubliez pas d'écrire les constructeur et destructeur nécessaires. **Note** : L'affichage des éléments de l'ensemble doit être effectué sous la forme `cout<<*e<<endl;` pour chaque élément `e`.
4. Quelle sera la sortie standard lors de l'exécution du programme suivant :

```
int main()
{
   Rectangle r(1.0, 1.0, 0, 2.0, 4.0) ;
   Cercle c(-1.0, -1.0, 256, 1.0) ;
   Ecran ecran() ; // l'ecran peut contenir 5 formes au max
   ecran.attache(&r)
   ecran.attache(&c) ;
   ecran.affiche() ;
   cout << ecran.surface() << endl ;
}
```

Méthodes virtuelles et membres statiques

Modifiez l'implémentation de la classe `Forme2D` de façon à ce que l'ensemble des formes soit stocké dans un tableau membre statique de la classe, plutôt qu'une instance de la classe `Ecran`. Ajoutez les méthodes statiques nécessaires

Bon travail