Travail pratique 2 (20 % +10%)

Date de remise 25 Octobre

Numéro 1 (20 points)

Réalisez un programme en C++ qui comporte la classe incomplète suivante :

```
class Etudiant
{
private:
string codePerm; // exemple "TREJ15028009"
int nbCafe; // nombre de tasses de café consommées par jour
public:
// constructeurs, méthodes à écrire

char getSexe()const { return . . . }
int getAge() const
{
      return 105 - atoi(codePerm.substr(8, 2).c_str());
}
. . .
};
```

Complétez la classe, en réalisant des constructeurs et autres fonctions membres appropriées afin de satisfaire aux exigences suivantes :

1. Dans la première *démonstration* (fonction **`demo1()`**) du fonctionnement de cette classe, on instancie (construit) deux objets de la classe **`Etudiant`** :
   - **`etud1`** : avec code permanent "`TREJ15028009`" est né le 15 février 1980. Il consomme 3 tasses de café par jour.
   - **`etud2`** : avec code permanent "`CHAN27568503`" est née le 27 juin 1985. Elle consomme 1 tasse de café par jour (valeur **par défaut**).

   On affiche les informations de ces deux étudiants comme suit :

```
Informations de TREJ15028009
- sexe              : masculin
- age a 2005        : 25 an(s)
- cafe consomme     : 3 tasse(s) par jour

Informations de CHAN27568503
- sexe              : feminin
- age a 2005        : 20 an(s)
- cafe consomme     : 1 tasse(s) par jour
```

   Déterminez et affichez les informations de l'étudiant qui consomme plus de café entre **`etud1`** et **`etud2`**.

Cette première démonstration permet aussi d'afficher la consommation de café originale de l'objet **`etud1`**, de réduire cette consommation à trois tasses de moins et de réafficher la consommation après la réduction.

Réaffichez les informations de l'étudiant qui consomme plus de café entre **`etud1`** et **`etud2`**.

2. Dans la deuxième *démonstration* (fonction **`demo2()`**) du fonctionnement de cette classe, on déclare et initialise un tableau de 5 étudiants, par exemple :

```
Etudiant etud[] = {
      Etudiant("TREJ15028009", 2),
      Etudiant("CHAN27568503"),
      Etudiant("ARCP02067001", 5),
      Etudiant("LAFJ31628104", 0),
      Etudiant("TREM01128607")
};
int nbEtud = sizeof(etud) / sizeof(Etudiant);
```

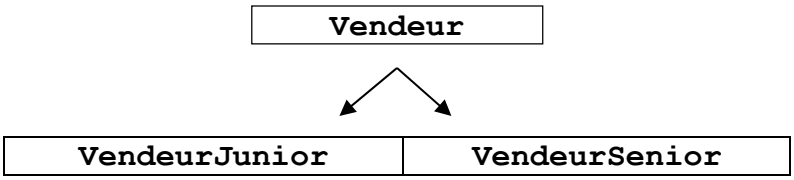
   a) affichez le contenu de ce tableau.
   b) supprimez le deuxième et le troisième étudiant puis affichez le contenu du tableau.
   c) déterminez et affichez les informations de l'étudiant le plus âgé dans le tableau restant.

Critères de correction

Classe **`Etudiant`**, pointeur **`this`**, méthodes **`get`**, **`set`**, ...	6 points
Bon fonctionnement	8 points
Présentation, qualité, etc.	6 points

Numéro 2 (10 points) Héritage simple

Voici la hiérarchie simple des classes pour des vendeurs que vous allez réaliser:



Réalisez la classe **`Vendeur`**, abstraite, contenant:

- Le numéro d'assurance sociale (NAS), une chaîne de 11 caractères *(ex.: 123 456 789)*.
- Le sexe du vendeur *(F ou M)*.
- Un constructeur approprié.
- Une méthode publique abstraite *(virtuelle pure)* calculant et retournant le salaire hebdomadaire. *Cette méthode sera concrétisée dans les sous classes.*
- La surcharge de l'opérateur d'affichage que voici:

```
friend ostream & operator << (ostream & out, const Vendeur & v)
{
v.afficher(out);
return out;
}
```

- Une méthode d'affichage virtuelle protégée *(protected)* dont voici la signature:

```
virtual void afficher(ostream &) const;
```

Cette méthode doit afficher le NAS, le sexe, ainsi que le salaire hebdomadaire.

Réalisez la sous classe **`VendeurJunior`**, dérivant de la classe **`Vendeur`**, contenant:

- Un champ privé pour le montant des ventes.
- Un constructeur approprié.
- La redéfinition de la méthode protégée `afficher` pour afficher également le montant des ventes.
- La concrétisation de la méthode de calcul du salaire hebdomadaire héritée de la classe `Vendeur`, sachant que le salaire d'un vendeur junior est calculé comme suit:

  Salaire de base de 400$ + 20% des ventes (en commission)

Réalisez la sous classe **`VendeurSenior`**, dérivant de la classe **`Vendeur`**, contenant:

- Un champ privé pour le nombre d'années d'ancienneté.
- Un constructeur approprié.
- La redéfinition de la méthode protégée `afficher` pour afficher également le nombre d'années d'ancienneté.

- La concrétisation de la méthode de calcul du salaire hebdomadaire héritée de la classe `Vendeur`, sachant que le salaire d'un vendeur senior est calculé comme suit:

  Salaire de base de 800$ + 35$ par année d'ancienneté

Vous disposez du fichier de données "**`vendeurs.txt`**" contenant les informations de vendeurs:

```
J 123 433 234 M 1560
S 534 234 125 M 12
...
```

C'est à dire, un homme vendeur junior (J) dont le NAS est "123 433 234" qui a vendu pour 1560$, et un homme vendeur senior (S) dont le NAS est "534 234 125" qui a 12 années d'ancienneté.

Lisez le fichier et remplissez un tableau (*maximum 20 vendeurs*) de pointeurs sur des objets `Vendeur` (c'est à dire un `Vendeur*[]`, dont les éléments pointent sur des `VendeurJunior` ou `VendeurSenior`).

1. Affichez les éléments du tableau à l'aide d'un petit patron de fonction une fois qu'il est rempli. *Réalisez si vous voulez la fonction suivante:*
   ```
   ostream & operator << (ostream & out, const Vendeur * v);
   ```

2. Triez les vendeurs du tableau en ordre croissant de leur salaire hebdomadaire.

3. Affichez le vendeur le moins bien payé et le vendeur le mieux payé du tableau *(qui devraient ici correspondre au premier et au dernier élément du tableau).*

Critères de correction

Classe abstraite `Vendeur`	3 points
Sous classes `VendeurJunior` et `VendeurSenior`	3 points
Lecture du fichier, remplissage du tableau de `Vendeur*`, autres tâches	3 points
Fonctionnement, qualité, présentation, etc.	1 point