Examen Licence Informatique - L3

Programmation par objets 2 (ULIN 606)

Session 2 - juin 2007

Tous documents autorisés

1 Généricité paramétrique

Soit la classe partielle `Armoire` suivante dans une version spécifique destinée à ranger des vêtements.

```
class Armoire{
private:
vector<Vetement*> contenu;
public:
Armoire();
virtual bool contient(Vetement* v);
};

Armoire::Armoire(){}
bool Armoire::contient(Vetement* v)
{return find(contenu.begin(),contenu.end(),v)!=contenu.end();}
```

Question 1.1

- *Expliquez pourquoi le constructeur de cette classe est vide.*
- *Que se passerait-il si la classe `Vetement` avait des sous-classes et si le vecteur était ainsi déclaré : `vector<Vetement> contenu` ?*
- *Proposez une version générique de la classe `Armoire` paramétrée par le type des objets rangés à l'intérieur (.h et .cc)*
- *Ecrivez le destructeur pour cette classe paramétrée.*
- *Ecrivez une méthode d'ajout d'un élément dans une armoire (paramétrée).*
- *Supposez que la classe `Vaisselle` existe. Ecrivez un programme `main` contenant une instanciation d'armoire contenant de la vaisselle et appelez les méthodes `ajout` et `contient`.*

2 Héritage multiple

Question 2.1 *Faites un schéma de la hiérarchie d'héritage correspondant au programme ci-dessous. Dans ce schéma il vous est demandé de faire figurer les liens d'héritage (y compris d'éventuels liens transitifs si l'exemple le veut), mais également l'ordre de déclaration des superclasses de chaque classe afin de faciliter l'application de l'algorithme.*

```
class Vin{
 private:
  string couleur;
  string nom;
 public:
  Vin(){cout << "Vin " << endl;}
  virtual ~Vin(){}
  virtual void copieCaracteristiques(string&)const{cout <<  "copie caracteristiques " << endl;}
};

class Vin_Invention : virtual public Vin{
 public:
 Vin_Invention(){cout << "Vin_Invention " ;saisieCaracteristiques(cin);}
 virtual ~Vin_Invention(){}
 virtual void saisieCaracteristiques(istream& is){cout <<  "saisie caracteristiques " << endl;}
};
```

```
class Vin_Supervise : virtual public Vin{
 private:
static string Historique;
 public:
 Vin_Supervise(){cout << "Vin_Supervise " ;copieCaracteristiques(Historique);}
 virtual ~Vin_Supervise(){}
};

string Vin_Supervise::Historique="-";

 class Bourgogne  : virtual public Vin_Invention, virtual public Vin_Supervise {
 public:
 Bourgogne(){cout << "Bourgogne " << endl;}
 virtual ~Bourgogne(){}
 };

int main(){
 cout << " ------------------------ " << endl;
  Bourgogne instanceOfBourgogne;
  cout << endl;
}
```

Question 2.2 *Montrez ce qu'affiche ce programme en expliquant l'algorithme que vous avez déroulé pour connaître l'ordre d'appel des constructeurs. Le graphe des liens d'héritage peut contenir des arcs de transitivité qui peuvent être empruntés lorsque vous appliquez l'algorithme.*

Question 2.3 *Que changeriez-vous à ce programme pour faire de l'héritage « répété » ? Quel est alors le schéma mémoire de* `instanceOfBourgogne` *et qu'en pensez-vous ?*

3 Cave vinicole

Nous étudions quelques éléments simplifiés de modélisation et de programmation pour un système de gestion de cave vinicole.

Les *cépages* sont décrits par un nom, une région et le fait d'être agréés pour leur utilisation dans les vins de qualité AOC. Ils disposent de deux méthodes :

- `string couleur()`, abstraite dans le cas général, qui retourne la couleur du cépage,
- `string toString()` retournant une chaîne contenant le nom, la région et la couleur du cépage.

Les *cépages* sont classés dans deux catégories suivant leur couleur :

- *cépages rouges* décrits en plus par leur taux en tanins, qui est ajouté à la chaîne retournée par la méthode `toString`,
- *cépages blancs*.

La couleur doit être stockée sous forme d'une variable `static` dans les classes représentant les cépages rouges et les cépages blancs. Chacune de ces deux classes contient sa propre implémentation de la méthode `string couleur()`.

Un vin est un assemblage de plusieurs cépages, il a une dénomination et une qualité (AOC, AOVDQS, pays, table). On dispose pour l'instant pour les vins de deux méthodes (on va en ajouter) :

- `string toString()` retournant une chaîne contenant le nom et la liste des cépages,
- `ajoute(c :Cepage)` qui ajoute un cépage à un vin sans effectuer de contrôle.

La figure 1 vous propose un schéma possible pour ces classes que nous vous invitons à respecter pour le reste de l'examen.

Question 3.1 (Premiers éléments des classes)
Utilisez la classe `vector` *de la librairie STL pour implémenter les collections qui vous seront nécessaires dans le code C++.*

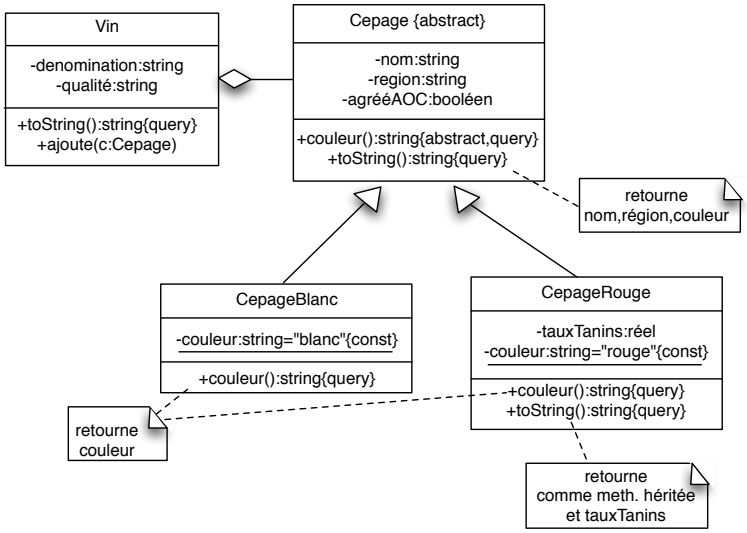

FIG. 1 – Une solution de conception pour la cave vinicole

Ecrivez partiellement en C++ les classes représentant les cépages (`Cepage`,`CepageRouge` et `CepageBlanc`) et les vins. Ecrivez le code du fichier .h (header) et du fichier .cc (implémentation).

N'écrivez dans cette question QUE :

- *l'entête des classes (dont les noms, liens de spécialisation, etc.),*
- *les attributs,*
- *les constructeurs,*
- *les destructeurs.*

Pour la suite, vous supposerez que les accesseurs existent.

Question 3.2 (couleur)

Ecrivez, pour les classes représentant les vins, la méthode `couleur`.

Question 3.3 (contient)

Ecrivez, pour la classe représentant les vins, une méthode `contient` qui retourne vrai si un cépage qui lui est passé en paramètre appartient à un vin, faux sinon.

Question 3.4 (saisie des cépages)

Ecrivez, pour les classes représentant les cépages, une méthode `saisie(istream &is)` qui saisit les caractéristiques d'un cépage (tous les attributs d'une instance) sur un flot d'entrée quelconque `is`. Proposez ensuite une surcharge de l'opérateur d'extraction dans un flot pour saisir les cépages.

Question 3.5 (saisie d'un vin)

Nous étudions, pour la classe représentant les vins, une méthode `saisie(istream &is)` qui saisit les caractéristiques d'un vin sur un flot quelconque `is`.

(a) Proposez des classe d'exceptions pour représenter deux des problèmes pouvant survenir lors de la saisie des cépages d'un vin, et qui se produisent (1) lorsque le vin est un AOC, mais que le cépage c saisi n'est pas agréé pour les AOC, (2) lorsque le cépage est déjà contenu dans le vin.

(b) Ecrivez la méthode `saisie` en prévoyant le(s) signalement(s) d'exception approprié(s).

(c) Ecrivez également un petit programme qui se préoccupe de capturer les exceptions liées à une saisie.