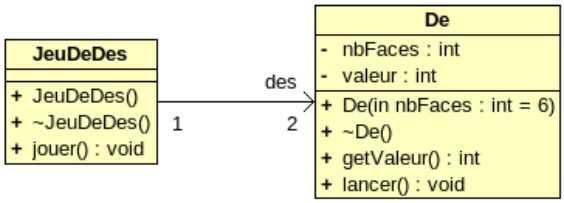
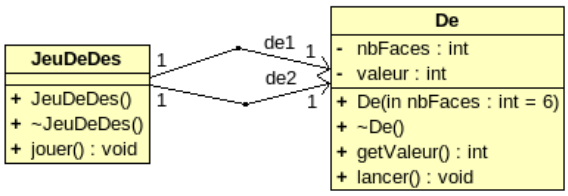
Génération de code C++ avec bouml

Association



Dans la classe JeuDeDes :

```
De * des[2];
```

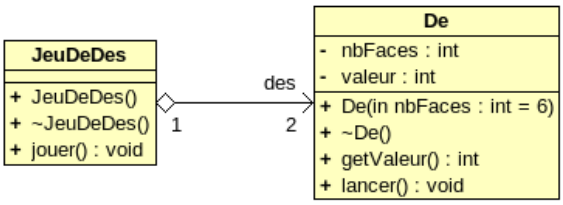


Dans la classe JeuDeDes :

```
De * de1;
De * de2;
```

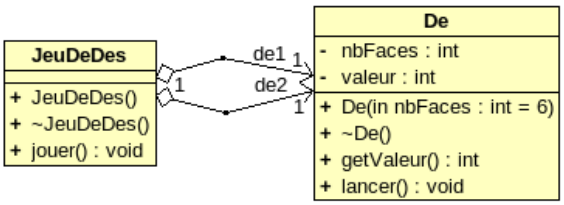
Remarque : le sens de navigation unidirectionnel implique que le JeuDedes doit connaître les deux dés mais la réciproque n'est pas vrai.

Agrégation



Dans la classe JeuDeDes :

```
De * des[2];
```

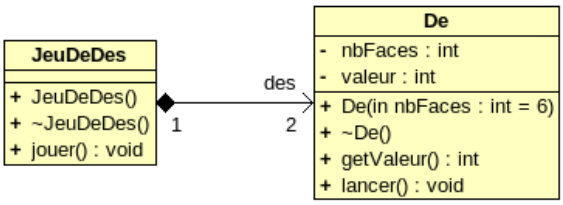


Dans la classe JeuDeDes :

```
De * de1;
De * de2;
```

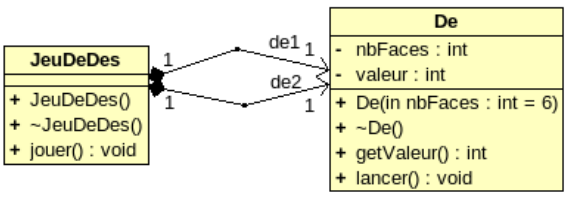
Remarque : pour les cardinalités de type 2..*, bouml n'utilise pas les conteneurs de la STL (*Standard Template Library*). On remarque aussi que bouml « code » de la même manière une association et une agrégation.

Composition



Dans la classe JeuDeDes :

```
De des[2];
```

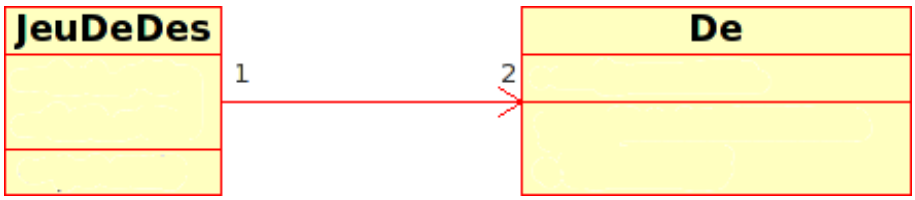


Dans la classe JeuDeDes :

```
De de1;
De de2;
```

Remarque : Les objets De n'existent que si l'objet JeuDeDes existe : c'est donc une composition. Lorsque l'objet JeuDeDes sera détruit, les objets De le seront aussi. L'objet JeuDeDes est donc responsable de la création et la destruction des objets De associés (composés donc). Le code généré n'est pas le mêmeque pour une association ou une agrégation.

Association (idem pour l'agrégation)



Dans le fichier JeuDeDes.h :

```
private:
       vector<De*> m_desVector;

public:
  void addDes ( De * add_object );
  void removeDes ( De * remove_object );
  vector<De *> getDesList ( );
```

Dans le fichier JeuDeDes.cpp :

```
/**
 * Add a Des object to the m_desVector List
 */
void JeuDeDes::addDes ( De * add_object ) {
  m_desVector.push_back(add_object);
}

/**
 * Remove a Des object from m_desVector List
 */
void JeuDeDes::removeDes ( De * remove_object ) {
  int i, size = m_desVector.size();
  for ( i = 0; i < size; ++i) {
       De * item = m_desVector.at(i);
       if(item == remove_object) {
              vector<De *>::iterator it = m_desVector.begin() + i;
              m_desVector.erase(it);
              return;
       }
   }
}

/**
 * Get the list of Des objects held by m_desVector
 * @return vector<De *> list of Des objects held by m_desVector
 */
vector<De *> JeuDeDes::getDesList ( ) {
  return m_desVector;
}
```

Remarque : pour les cardinalités de type 2..*, umbrello utilise les conteneurs de la STL (*Standard Template Library*) et génère le code complet de leur utilisation. Ceci est donc un bon exemple d'utilisation du type vector de la STL.