Exercice 3 (Classe afficheur LCD)

OBJECTIF :

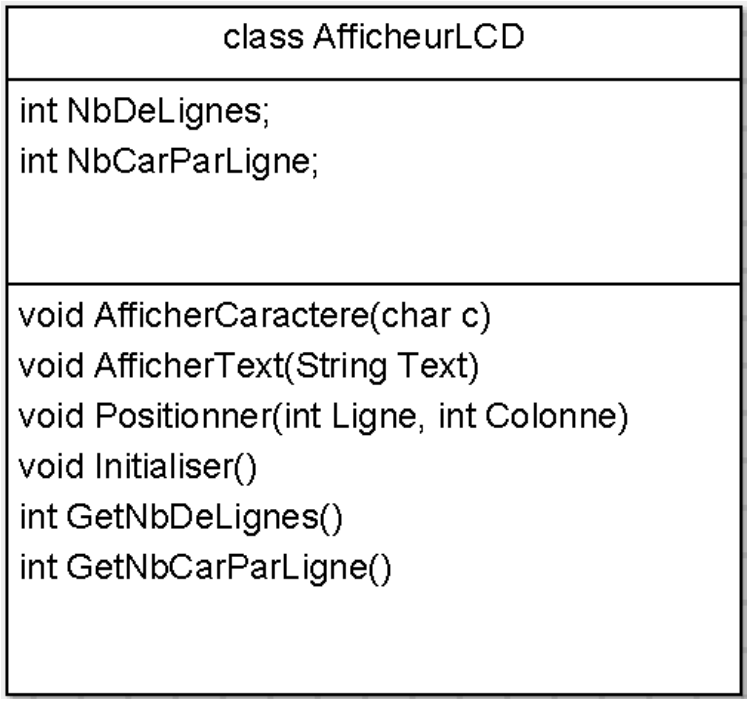
A l'issue de la réalisation de cet exercice, les étudiants doivent être capables de réaliser une classe et de l'utiliser.
Ceci dans le cadre d'une application console avec le Visual Studio 2015.

PREPARATION DU PROJET

Sous K:\ES\Maitres-Eleves\SLO\Modules\SL228_POBJ\CoursCpp\Exercices\Ex3 vous disposez d'une base de modèle UML (AfficheurLCD_Ex3 - Base.zargo). Il faut évoluer cette base pour obtenir le diagramme ci-dessous.

Le projet comprend 3 fichiers : le fichier programme principal que nous nommerons Ex3.cpp (à créer manuellement) et les 2 fichiers AfficheurLCD.h et AfficheurLCD.cpp.

La base est obtenue par la génération de Code ArgoUML du modèle ci-dessous:



Remarque : pour le projet Ex3 dans les fichiers sources vous devez avoir Ex3.cpp et AfficheurLCD.cpp. Dans les fichiers d'en-tête vous devez avoir AfficheurLCD.h.

DONNEES DU PROBLEME

Il s'agit d'ajouter la simulation de l'afficheur LCD en allouant dynamiquement un tableur de n lignes de x caractères en relation avec les paramètres fournis. Allocation à placer dans le constructeur.

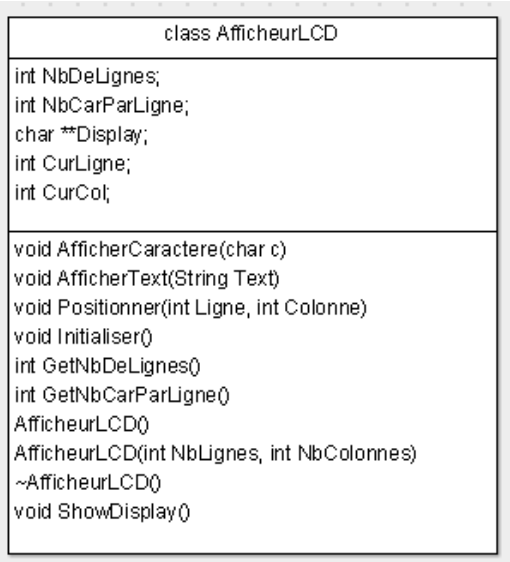
Afin de tester la gestion de la matrice simulant l'afficheur LCD, il faut modifier les fonctions d'affichage pour qu'elles remplissent la matrice au lieu d'afficher directement. Il devient nécessaire de tenir compte de la position en cours.

Il est nécessaire d'ajouter une nouvelle méthode void **ShowDisplay**() qui affiche le contenu des n lignes.

Pour pouvoir allouer la matrice il faut ajouter un attribut privé :
char **Display.

Pour pouvoir gérer la position courante de l'affichage il faut ajouter deux attribut privé :
int CurLigne // ligne en cours 1 à NbLigne.
int CurCol // colonne en cours de 1 à NbCarParLigne.

NOUVELLE SITUATION DU MODELE UML



DETAIL ALLOCATION DE LA MATRICE

Au niveau du .h la matrice est déclarée **char **Display.**

Au niveau du constructeur il est nécessaire d'allouer ligne par ligne de la manière suivante:

```
Display = new char*[NbDeLignes];
for (int i=0;i < NbDeLignes;i++)
{
    Display[i]=new char[NbCarParLigne+1];
}
```

Remarque : il faut un char supplémentaire pour la terminaison de la chaine.

DETAIL METHODE INITIALISER

Pour obtenir une bonne simulation de l'affichage il faut initialiser les n lignes de x colonnes avec le caractère '.' et positionner l'affichage en 1,1.

DETAIL METHODE AFFICHERTEXT

L'objectif est de placer le Texte à la position courante. Il faut gérer le dépassement du nombre de caractères par ligne et mettre à jour CurCol.

Pour passer du string à un tableau de char il faut procéder ainsi :

```
const char *ptr1 = 0;
ptr1= Text.data ( );
```

DETAIL METHODE AFFICHERCARACTERE

L'objectif est de placer le caractère à la position courante. Il faut gérer le dépassement du nombre de caractères par ligne et mettre à jour CurCol (+1 pour chaque caractère)

DETAIL METHODE SHOWDISPLAY

Affichage de l'ensemble de la matrice avec ajout d'une ligne vide.

DETAIL METHODE POSITIONNER

Mettre à jour CurLigne et CurCol sans dépasser les maximums.

DETAIL DESTRUCTEUR

Le destructeur doit maintenant supprimer la mémoire allouée. Il faut procéder à l'inverse de la construction.

ACTION DANS LE PROGRAMME PRINCIPAL ET RESULTAT

Voici un exemple d'action dans le programme principal.

```
AfficheurLCD MonLCD(4, 20);

// utilisation de l'objet MonLCD
MonLCD.Initialiser();      // Initialisation de MonLCD
MonLCD.ShowDisplay();

MonLCD.Positionner(1,1);
MonLCD.AfficherText("ETML !");
MonLCD.Positionner(2,1);
MonLCD.AfficherText("Exercice 3 !");
MonLCD.Positionner(3,1);
MonLCD.AfficherText("Ligne 3 !");
MonLCD.Positionner(4,1);
MonLCD.AfficherText("Ligne 4 !");
MonLCD.ShowDisplay();

MonLCD.Positionner(2,18);
MonLCD.AfficherCaractere('*');
MonLCD.AfficherCaractere('*');
MonLCD.AfficherCaractere('*');
MonLCD.AfficherCaractere('*');
MonLCD.ShowDisplay();

MonLCD.Positionner(3,1);
MonLCD.AfficherText("Ecole ");
MonLCD.AfficherText("Superieure ");
MonLCD.AfficherText("Lausanne");
MonLCD.AfficherText("Bug");
MonLCD.ShowDisplay();

cout << "Mon LCD a " << MonLCD.GetNbDeLignes() << " lignes  et "
     << MonLCD.GetNbCarParLigne() << " caracteres par ligne" << endl;
```

Exemple de résultat :