

## Implémentation d'un éditeur de graphes en C#

### Etape 2 : Implémentation d'un graphe

*Définition : Un **graphe** est un couple  $(V, E)$ , où  $V$  est un ensemble de nœuds (ou sommets) du graphe, et  $E$  un ensemble d'arêtes reliant 2 sommets de  $V$ .*

*Définition : L'**encapsulation** consiste à cacher l'information contenue dans un objet et de ne proposer que des méthodes de manipulation de cet objet. Ainsi les propriétés et axiomes associés aux informations contenues dans l'objet seront assurés/validés par les méthodes de l'objet et ne seront plus de la responsabilité de l'utilisateur extérieur. L'utilisateur extérieur ne pourra pas modifier directement l'information et risquer de mettre en péril les axiomes et les propriétés comportementales de l'objet.*

#### 2.1. La classe Noeud

Cette classe définit les objets de type Noeud de votre graphe. Elle contient les informations (des *attributs*) relatives à un nœud et propose des *méthodes* permettant de manipuler un objet de cette classe (principalement l'affichage dans notre cas).

Ajoutez une nouvelle classe Noeud à votre projet en utilisant le menu **Projet / Ajouter une classe**.

Afin d'accéder aux classes de dessin et de formulaire sans devoir préciser le chemin complet de la classe, ajoutez les 2 lignes suivantes dans l'en-tête de votre fichier :

```
using System.Drawing;  
using System.Windows.Forms;
```

Ajoutez de nouveaux champs à votre classe : un champ centre de type Point qui contiendra les coordonnées du nœud, un champ taille de type Size pour préciser la taille d'affichage du nœud et un champ couleur de type Color déterminant la couleur d'affichage du nœud.

Modifiez le constructeur de la classe Nœud (c'est-à-dire la méthode public Noeud()) afin qu'il prenne en paramètres les valeurs initiales des champs centre, taille et couleur.

Ecrivez la méthode public void dessiner(PaintEventArgs e) qui affiche le nœud à l'aide des méthodes fournies par l'objet de type Graphics fourni dans le paramètre e.

#### 2.2. La classe Arete

Une arête relie deux nœuds du graphe à l'aide d'un segment de droite. Aussi, un objet de type Arete doit encapsuler deux références sur les nœuds qu'il relie, ainsi que, pour l'affichage, l'épaisseur et la couleur du segment.

De la même manière que pour la classe Nœud, écrivez la classe Arete, comportant les champs noeud1 et noeud2 de type Noeud, epaisseur et couleur. Modifiez le constructeur de la classe et écrivez la méthode dessiner.

#### 2.3. La classe Graphe

Créez une nouvelle classe Graphe. Cette classe encapsule deux listes d'objets : celle recensant les nœuds et celle recensant les arêtes du graphe. On utilisera pour cela la structure de données System.Collections.Generic.List<>. On prendra soin, dans l'écriture du code de la classe, de masquer autant que possible l'existence des classes Noeud et Arete à l'utilisateur de la classe.

Définissez ensuite les méthodes de base pour modifier un graphe (c'est-à-dire les opérations d'ajout et de suppression de nœuds et d'arêtes), ainsi que la méthode dessiner qui affiche itérativement l'ensemble des nœuds et des arêtes du graphe.

### Etape 3 : Modification de la classe VueGraphe

Modifiez la classe VueGraphe afin qu'elle encapsule un objet de type Graphe.

Redéfinissez la méthode appelée par la plateforme pour redessiner le formulaire : protected override void OnPaint(PaintEventArgs e) pour qu'elle affiche le graphe encapsulé. N'oubliez pas d'appeler la méthode OnPaint de la classe parent à l'aide de base.OnPaint(e);

On veut qu'un nœud soit ajouté à la position de la souris lorsque l'utilisateur clique dans le formulaire. Pour cela, ajoutez un gestionnaire d'événements traitant l'appui sur le bouton gauche de la souris dans le formulaire (MouseDown).

Faites en sorte que l'on ne puisse créer un nœud sur un nœud déjà existant. Pour cela ajoutez une méthode public Object contient(Point) dans la classe Graphe et des méthodes public bool contient(Point) dans les classes Noeud et Arete.

L'utilisateur doit pouvoir créer une arête en cliquant sur un nœud existant et en déplaçant le curseur jusqu'à la deuxième arête avant de relâcher le bouton gauche de la souris (opération de type *drag'n drop*). Pour cela, vous devez gérer les événements MouseDown, MouseUp et MouseMove.

Vous modifierez ensuite la méthode OnPaint, pour qu'un segment soit tracé entre le 1er nœud sélectionné et la position courante du curseur lors d'une opération d'ajout d'arête.

Gérez la possibilité pour l'utilisateur de déplacer les nœuds à l'aide d'un clic droit de la souris.