Activité n°3 – *Gestion des commerciaux (version application Windows)*

Cette activité est la suite de l'activité n°2. Vous devez donc reprendre l'application *Commerciaux* terminée.

1. Préparation de l'application

En vous aidant du code de l'application Commerciaux de l'activité n°1, vous allez créer un nouveau projet *GestionCommerciaux* avec interface graphique. Nous allons entièrement développer une nouvelle solution pour ce projet même s'il aurait été possible que notre solution de l'activité n°1 contienne plusieurs projets.

Travail à réaliser

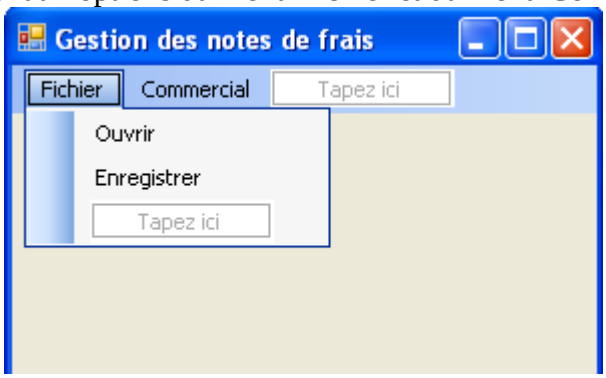
- Créer un nouveau projet (GestionCommerciaux) de type Application Windows Forms ;
- Créer les différentes classes de l'activité précédente et intégrer le code développé.

2. L'application WinForm GestionCommerciaux

L'application que nous allons réaliser contiendra plusieurs formulaires.

a. Le formulaire principal

Ce formulaire permet d'accéder aux options du menu Fichier et du menu Commercial.



Travail à réaliser

- Construire ce formulaire en ajoutant un composant de type MenuStrip et ajouter les options du menu.
- Ce formulaire est le formulaire principal et contient les autres à l'aide du mécanisme de fenêtre MDI. Dans la fenêtre de propriété du formulaire, indiquer que ce formulaire est un conteneur MDI.

L'option *Ouvrir* lance une boîte de dialogue permettant d'accéder à un fichier; Pour l'instant .

Travail à réaliser

- Ajouter deux attributs privés dans le code du formulaire.

```
private ServiceCommercial leService;
private string nomFichier;
```

- Dans l'événement clic du menu *Fichier* > *Ouvrir*, écrire le code qui :
  - ouvre la boite de dialogue,
  - récupère le fichier sélectionné par l'utilisateur,
  - appelle la méthode qui charge le ServiceCommercial à partir du fichier.

- Dans l'événement clic du menu *Fichier* > *Enregistrer*, écrire le code qui permet de sauvegarder le ServiceCommercial dans le fichier.

Les options du menu *Commercial* ouvre chacune un formulaire différent. C'est l'événement clic de chaque sous-menu qui appellera le nouveau formulaire.
Chaque formulaire a besoin du ServiceCommercial construit à partir du fichier dans le formulaire principal. Le formulaire principal passera cet objet à l'aide du constructeur de chacun des nouveaux formulaires :

```
private void ajouterToolStripMenuItem_Click(object sender, EventArgs e)
{
    FrmAjoutCommercial f = new FrmAjoutCommercial(leService);
    f.MdiParent = this;
    f.Show();
}
```

Commentaire du code : La première ligne crée un nouveau formulaire (d'ajout d'un commercial)
La seconde ligne indique que son formulaire parent est le formulaire courant.
La troisième ligne montre le formulaire.

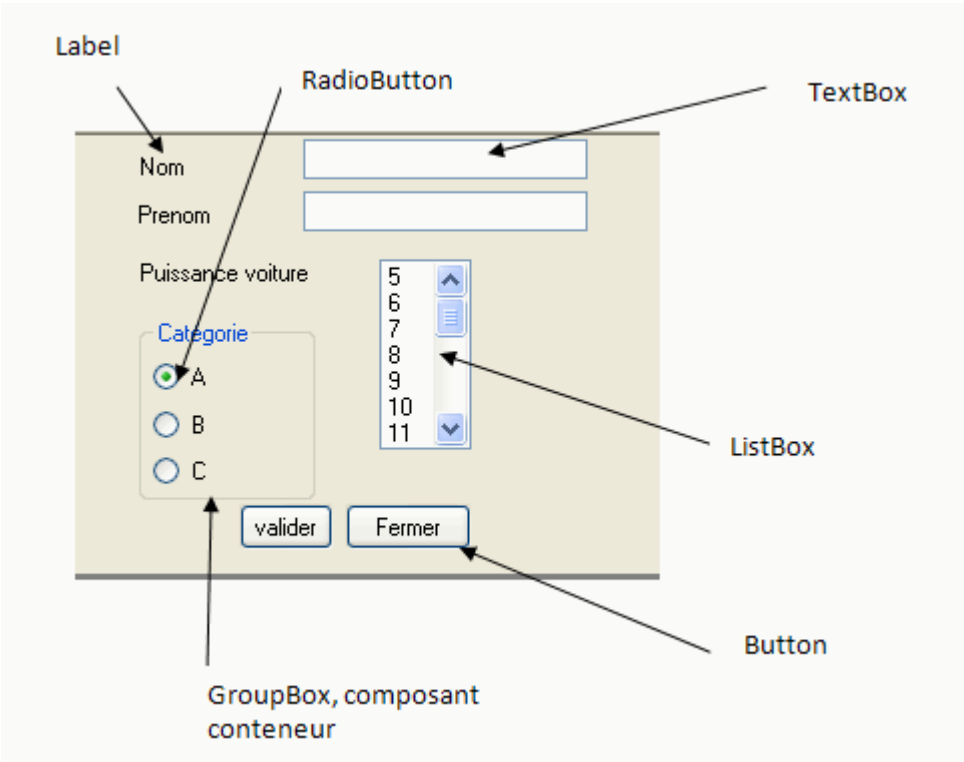
Ceci ne peut se faire bien sûr que si un formulaire nommé ici *FrmAjoutCommercial* a été créé en conception et que son constructeur a été modifié. Chaque classe formulaire contiendra un attribut privé de type ServiceCommercial.

Travail à réaliser

- Ajouter les trois formulaires qui dans un premier temps ne contiendront rien. Les trois formulaires doivent permettre :
  - de créer un nouveau commercial.
  - d'ajouter une nouvelle note de frais.
  - de visualiser les notes de frais des différents commerciaux.
- Tester et vérifier que chaque formulaire s'ouvre bien à partir des menus.

b. Le formulaire d'ajout d'un nouveau commercial

Ce formulaire devra permettre d'ajouter un commercial à notre service commercial. Il aura la forme suivante :



La première chose à faire après le dépôt des composants sur le formulaire est de modifier la propriété Name, en respectant par exemple les règles suivantes :

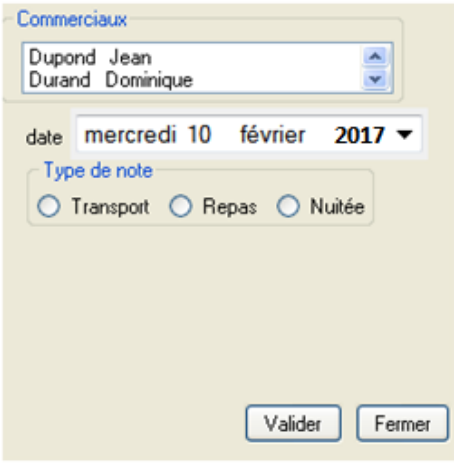
Composant	Préfixe	Exemple
Formulaire	frm	frmAjoutCommercial
Button	btn	btnValider
TextBox	txt	txtNom
Label	lbl	lblNom
ListBox	lst	lstPuissance
GroupBox	grBox	grBoxCategorie
RadioBox	rdBtn	rdBtnCategorie
CheckBox	chk	chkVille
DateTimePicker	dtp	dtpDate
ComboBox	cbBox	cbBoxNom

Ceci est un exemple de normalisation, à vous de vous faire votre propre charte de programmation. Il n'est par contre pas nécessaire de renommer les composants qui ne seront jamais utilisés dans le code comme c'est souvent le cas des Label.

Travail à réaliser

- Compléter le formulaire. Le chargement des valeurs "en dur" dans la ListeBox se fait à l'ouverture du formulaire (événement Load) en utilisant la méthode Add de la propriété Items. Le bouton valider teste si le RadioButton est coché et crée un nouveau commercial en l'ajoutant au service commercial.
- Tester que le nouveau commercial est bien enregistré dans le fichier sérialisé.

c. Le formulaire d'ajout de notes de frais



La liste des commerciaux est remplie à partir de tous les commerciaux du service commercial.

```
private void FrmAjoutNote_Load(object sender, EventArgs e)
{
    for (int i = 0; i < leService.nbCommerciaux(); i++)
    {
        Commercial c = leService.getCommercial(i);
            ... //Vous pouvez utiliser un foreach
```

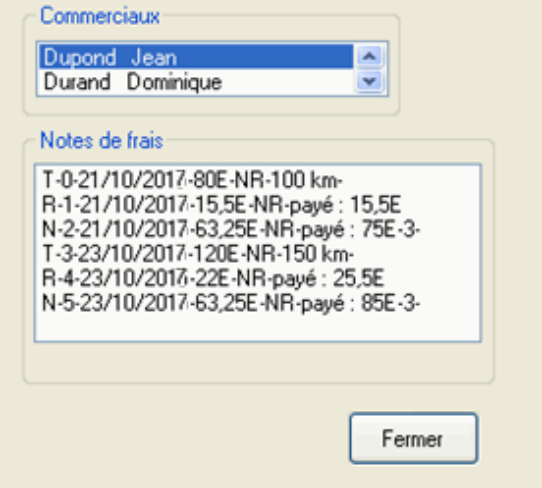
La sélection d'un radioButton fait apparaître les champs nécessaires à la saisie de la note de frais concernée comme dans l'exemple ci-dessous:

Le bouton Valider ajoute au commercial sélectionné la note de frais (Transport, midi ou nuitée) correspondante. Pour récupérer le commercial, il faudra extraire le nom et le prénom à partir de la sélection de la liste (SelectedItem).

Travail à réaliser

- Construire le formulaire.
- Imaginer un test pour vérifier que la note de frais a bien été ajoutée au bon commercial.

d. Le formulaire de visualisation des notes de frais



La gestion de la liste des commerciaux est identique au formulaire précédent. Lorsque l'on clique sur un commercial, ses notes apparaissent (événement *selectedIndexChange*d ou ...). Il faudra utiliser la méthode *ToString()* écrite dans l'application console.

Travail à réaliser

- Construire le formulaire.
- Tester.