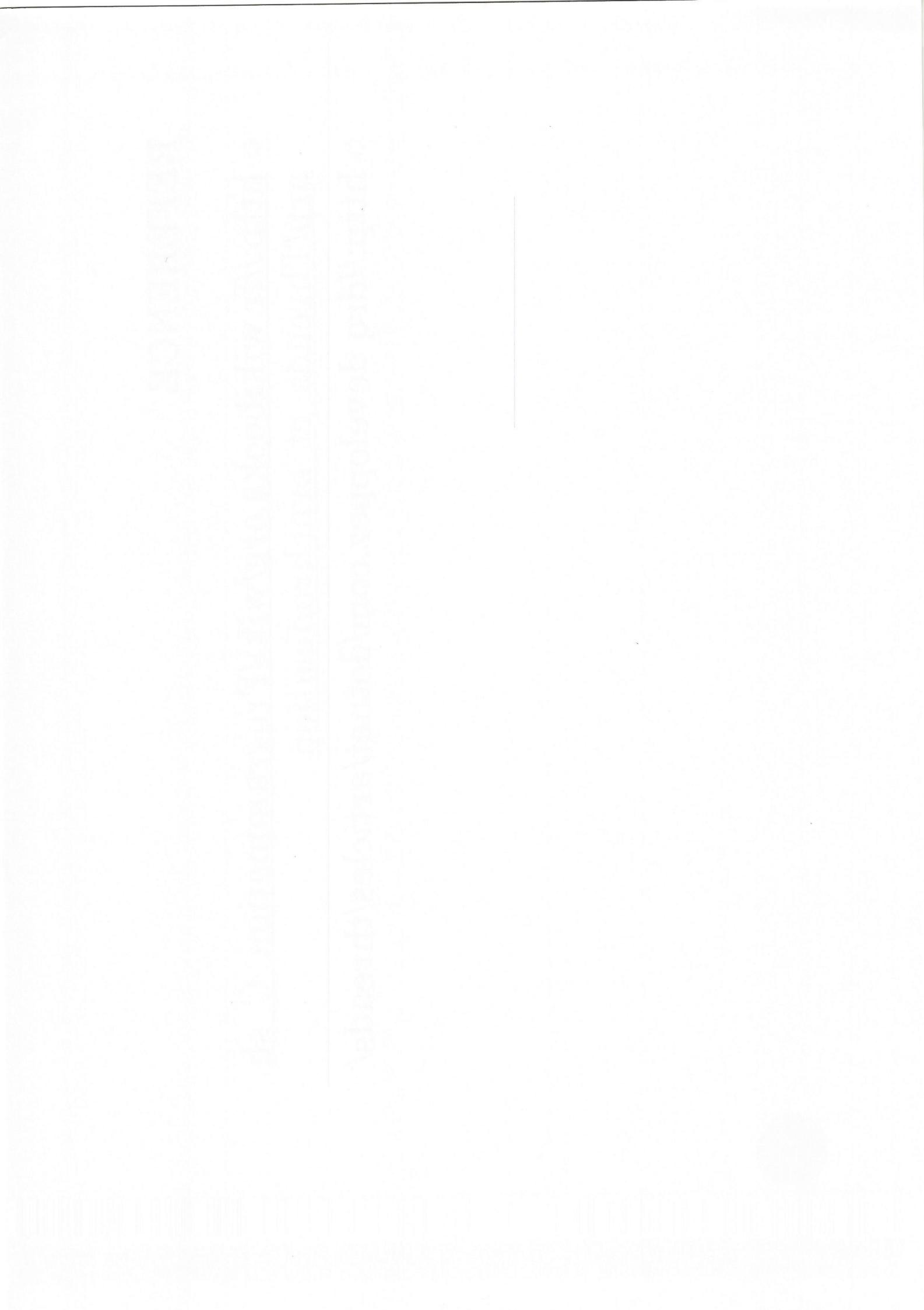


## REFERENCE

- o [http://fr.wikibooks.org/wiki/Programmation C sharp/Threads et synchronisation](http://fr.wikibooks.org/wiki/Programmation_C_sharp/Threads_et_synchronisation)
- o <http://drq.developpez.com/dotnet/articles/threads/>



Health Care Policy and the Politics of Health Care Policy  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

Health Politics, Policy and Law  
Edited by Michael J. Krasner and Daniel P. Gitterman

```
176        }
177        private void radioButton_interet_9_CheckedChanged(object      ↵
178            sender, EventArgs e)
179        {
180            if (radioButton_interet_9.Checked == true)
181            {
182                emprunt1.TauxInteretAnnuel = 0.09f;
183                MiseAJourIHM();
184            }
185        }
186        #endregion
187
188        private void button_ok_Click(object sender, EventArgs e)
189        {
190            if (textBox_capital_emprunte.Text == "0")
191            {
192                errorProvider_capitalVide.SetError
193                    (textBox_capital_emprunte, "Le Champ Capital doit      ↵
194                     être supérieur à 0 veuillez saisir un nombre entier      ↵
195                     de 10 chiffres maximum");
196            }
197            else
198            {
199                errorProvider_capitalVide.SetError
200                    (textBox_capital_emprunte, "");
201            }
202        }
203        private void Reinitialiser()
204        {
205            listBox_periodicite_rembs.SelectedIndex = 0;
206            radioButton_interet_7.Checked = true;
207            emprunt1.DureeEmpruntMois = 1;
208        }
209    }
210 }
```

```
131     }
132     //choix dans la listbox de la périodicité
133     private void listBox_periodicite_rembts_SelectedIndexChanged(object sender, EventArgs e)
134     {
135         if (listBox_periodicite_rembts.SelectedIndex == 0)
136         {
137             emprunt1.PeriodiciteRemboursement =
138                 Emprunt.EnumPeriodicite.Mensuel;
139         }
140         if (listBox_periodicite_rembts.SelectedIndex == 1)
141         {
142             emprunt1.PeriodiciteRemboursement =
143                 Emprunt.EnumPeriodicite.Bimestriel;
144         }
145         if (listBox_periodicite_rembts.SelectedIndex == 2)
146         {
147             emprunt1.PeriodiciteRemboursement =
148                 Emprunt.EnumPeriodicite.Trimestriel;
149         }
150         if (listBox_periodicite_rembts.SelectedIndex == 3)
151         {
152             emprunt1.PeriodiciteRemboursement =
153                 Emprunt.EnumPeriodicite.Semestriel;
154         }
155
156         MiseAJourIHM();
157     }
158     #region Bouton_Taux
159     //boutons de choix du taux annuel
160     private void radioButton_interet_7_CheckedChanged(object sender, EventArgs e)
161     {
162         if (radioButton_interet_7.Checked == true)
163         {
164             emprunt1.TauxInteretAnnuel = 0.07f;
165             MiseAJourIHM();
166         }
167     }
168     private void radioButton_interet_8_CheckedChanged(object sender, EventArgs e)
169     {
170         if (radioButton_interet_8.Checked == true)
171         {
172             emprunt1.TauxInteretAnnuel = 0.08f;
173             MiseAJourIHM();
174         }
175     }
```

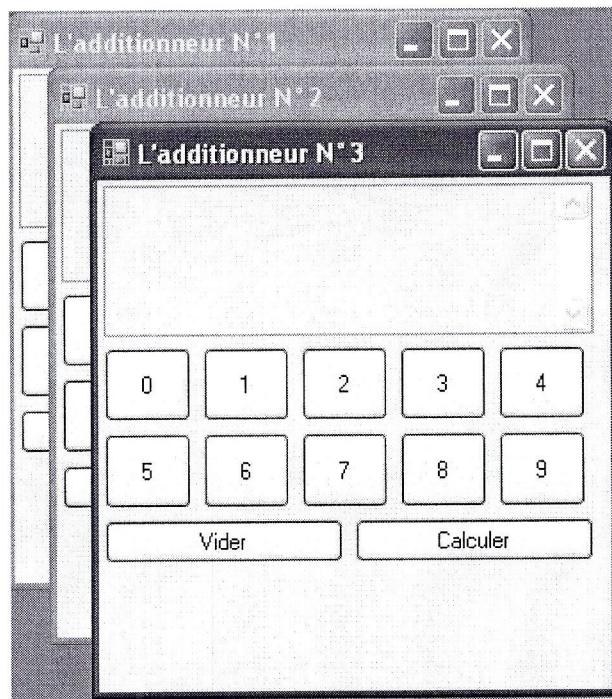
```
91     #region programme
92     //saisie le nom et contrôle la saisie du nom
93     private void textBox_nom_TextChanged(object sender, EventArgs e)
94     {
95         if (!Controles.Controle_Saisie_Nom(textBox_nom.Text))
96         {
97             errorProvider_nom.SetError(textBox_nom, "Saisir
98             uniquement des lettres (30 maximum sauf si c'est nom
99             composés), si c'est un nom composé 15 lettres avant un
100             '-' puis suivi d'un maximum 15 lettres.");
101         }
102         else
103         {
104             emprunt1.Nom = textBox_nom.Text;
105             errorProvider_nom.SetError(textBox_nom, "");
106         }
107     }
108
109     // saisie capital emprunté et contrôle la saisie
110     private void textBox_capital_emprunte_TextChanged(object
111         sender, EventArgs e)
112     {
113         if (!Controles.Controle_Saisie_Capital_Emptrunt
114             (textBox_capital_emprunte.Text))
115         {
116             errorProvider_capital.SetError
117                 (textBox_capital_emprunte, "Saisir uniquement des
118                 chiffres maximum 10");
119         }
120     }
121     // ScrollBar pour determiner le nombre de mois de l'emprunt en
122     // fonction de la périodicité
123     private void hScrollBar_duree_ValueChanged(object sender,
124         EventArgs e)
125     {
126         // la scrollbar
127         HScrollBar bar = (HScrollBar)sender;
128
129         bar.Value = Math.Max(1, bar.Value / bar.SmallChange) *
130             bar.SmallChange;
131         emprunt1.DureeEmpruntMois = hScrollBar_duree.Value;
132
133         MiseAJourIHM();
134     }
```

```
45         {
46             listBox_periodicite_rembs.SelectedIndex = 1;
47         }
48     }
49     else if (emprunt1.PeriodiciteRemboursement == Emprunt.EnumPeriodicite.Trimestriel)
50     {
51         listBox_periodicite_rembs.SelectedIndex = 2;
52     }
53 }
54 else if (emprunt1.PeriodiciteRemboursement == Emprunt.EnumPeriodicite.Semestriel)
55 {
56     listBox_periodicite_rembs.SelectedIndex = 3;
57 }
58 else if (emprunt1.PeriodiciteRemboursement == Emprunt.EnumPeriodicite.Annuelle)
59 {
60     listBox_periodicite_rembs.SelectedIndex = 4;
61 }
62 int val = (int)emprunt1.PeriodiciteRemboursement;
63 hScrollBar_duree.Minimum = val;
64 hScrollBar_duree.Maximum = 300 + (val - 1);
65 hScrollBar_duree.LargeChange = val;
66 hScrollBar_duree.SmallChange = val;
67 hScrollBar_duree.Value = emprunt1.DureeEmpruntMois;
68 if (label_Duree_en_mois.Text != hScrollBar_duree.Value.ToString())
69 {
70     label_Duree_en_mois.Text =
71         hScrollBar_duree.Value.ToString();
72 }
73 //choix taux
74 if (emprunt1.TauxInteretAnnuel == 0.07f)
75 {
76     radioButton_interet_7.Checked = true;
77 }
78 else if (emprunt1.TauxInteretAnnuel == 0.08f)
79 {
80     radioButton_interet_8.Checked = true;
81 }
82 else if (emprunt1.TauxInteretAnnuel == 0.09f)
83 {
84     radioButton_interet_9.Checked = true;
85 }
86
87 this.label_montant_remb_periodicite.Text = Math.Round
88     (emprunt1.Calcul_Remboursement(), 2) + "€".ToString();
89 this.label_nb_remboursements.Text =
90     emprunt1.Calcul_Nombre_Mensualite().ToString();
91 }
92 #endregion
```

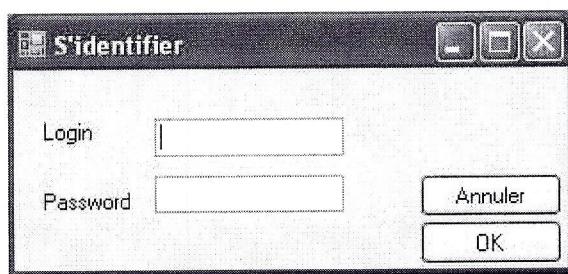
```
1  using ClassEmprunt;
2  using ClassLibrary_Controles;
3  using System;
4  using System.Collections.Generic;
5  using System.ComponentModel;
6  using System.Data;
7  using System.Drawing;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11 using System.Windows.Forms;
12
13
14 namespace WinFormsSynthese
15 {
16     public partial class FormEmprunt : Form
17     {
18         private Emprunt emprunt1;
19
20         public FormEmprunt()
21         {
22             InitializeComponent();
23             emprunt1 = new Emprunt("", 0, 1,
24             Emprunt.EnumPeriodicite.Mensuel, 0.07F);
25             listBox_periodicite_rembxs.Items.AddRange(new string[]
26             { "Mensuel", "Bimestriel", "Trimestriel", "Semestriel",
27               "Annuelle" });
28             MiseAJourIHM();
29         }
30
31         #region IHM
32         private void MiseAJourIHM()
33         {
34             //saisie capital
35             textBox_capital_emprunte.Text =
36                 emprunt1.CapitalEmprunte.ToString();
37
38             //choix duree scrollbar
39             emprunt1.DureeEmpruntMois = Math.Max
40                 (emprunt1.DureeEmpruntMois, (int)
41                 emprunt1.PeriodiciteRemboursement);
42
43             label_montant_remb_periodicite.Text =
44                 emprunt1.DureeEmpruntMois.ToString();
45
46             //choix periodicite
47             if (emprunt1.PeriodiciteRemboursement ==
48                 Emprunt.EnumPeriodicite.Mensuel)
49             {
50                 listBox_periodicite_rembxs.SelectedIndex = 0;
51             }
52             else if (emprunt1.PeriodiciteRemboursement ==
53                 Emprunt.EnumPeriodicite.Bimestriel)
```



Vous pouvez également numéroter les feuilles de l'additionneur.



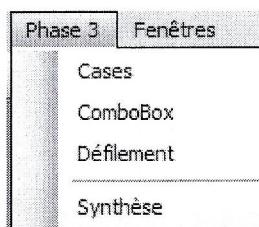
S'il vous reste du temps (et du courage !), vous pouvez créer une nouvelle feuille qui permettra à l'utilisateur de s'identifier : Vous mettrez en place une règle de validation (par exemple l'identification pourrait être bonne, si le login saisi est égal au mot de passe).



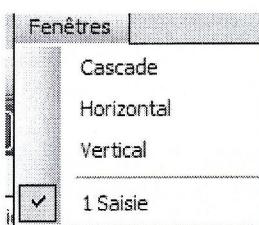
Un click sur le bouton OK validera la saisie et rendra accessible les menus de l'application.  
Une piste possible :: Rendez public (en utilisant les propriétés get/set de C#) les menus de la feuille MDI, de manière à ce que leur propriété **enabled** soit accessible par cette feuille de login.

Les **menus de phase** présentent les titres des différents exercices réalisés pendant cette séance de formation.

**Exemple :** Phase 3



Le menu **Fenêtres** présente les différentes façons d'arranger les fenêtres dans un environnement MDI, et la liste des fenêtres ouvertes.



La barre d'outils présente :

- Un **ToolStripButton** permettant de s'identifier
- Un **ToolStripSplitButton** reprenant les items du menu Phase 3.

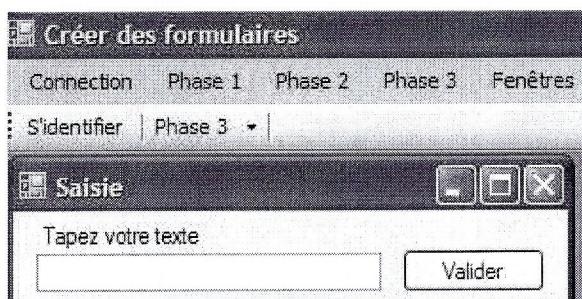
La barre d'état présente :

- Un **ToolStripStatusLabel** initialisé avec la date du jour
- Un **ToolStripStatusLabel** indiquant la dernière opération effectuée.

A l'affichage de la feuille, seuls le menu **Connection** et le bouton **S'identifier** de la barre d'outils sont accessibles : L'utilisateur doit s'identifier pour travailler sur l'application.

## IV.2 COMMUNICATION ENTRE FEUILLES

Dissocier la saisie du texte de la feuille **Cases à cocher et boutons radios** en créant une feuille de saisie qui transmettra en paramètre le texte saisi dans la TextBox au label de la feuille codée dans la phase 3.



## IV PHASE 4 : MENUS, BARRES D'OUTILS ET D'ETAT

### IV.1 MENUS, BARRES D'OUTILS, BARRES D'ETAT

#### IV.1.1 Objectifs

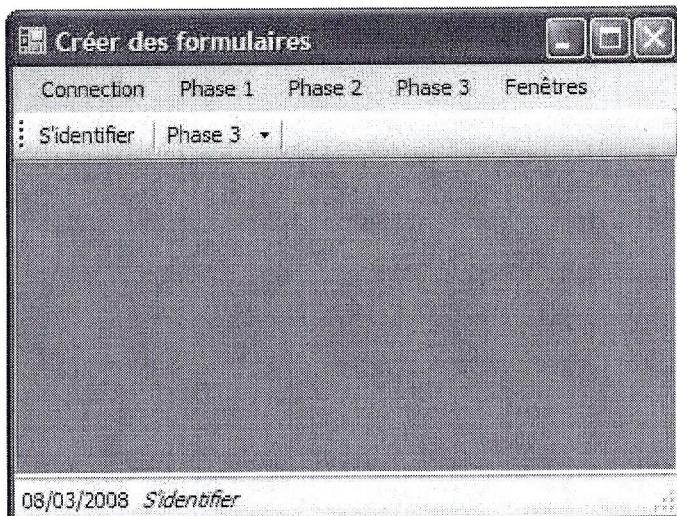
Utiliser les contrôles de menus

#### IV.1.2 But du traitement

Il s'agit de rassembler sous une feuille Menu, par exemple par phase, tous les exercices réalisés pendant cette séance de formation.

On ajoutera également une barre d'outils et une barre d'état.

Chaque click sur une option de menu ou dans la barre d'outil appelle la feuille concernée. A l'initialisation, seul le menu Connection et le bouton s'identifier de la barre d'outils sont accessibles.



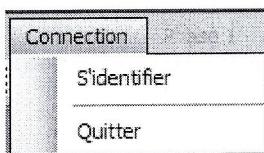
#### IV.1.3 Les traitements

Si tous vos exercices ont été réalisés dans des projets différents, démarrer un nouveau projet de nom Menu.

Renommez le formulaire Form1 en frmMenu, et positionnez ce formulaire comme conteneur MDI.

Importez chacune des feuilles des exercices réalisés, en faisant attention de renommer l'espace de nom sur toutes les feuilles d'extension .cs et .designer.cs.

#### Le menu Connection



- Un click sur **S'identifier**, enverra dans un premier temps un message de bienvenue, et permettra l'accessibilité à tous les menus de l'application.
- **Quitter** permettra de quitter l'application, après confirmation par une boîte de dialogue.

```
47         capitalEmprunte = _capitalEmprunte;
48         dureeEmpruntMois = _dureeEmpruntMois;
49         periodiciteRemboursement = _periodiciteRemboursement;
50         tauxInteretAnnuel = _tauxInteretAnnuel;
51     }
52     public override string ToString()
53     {
54         return "Emprunt" + nom + " " + capitalEmprunte + " " +
55             dureeEmpruntMois + " " + periodiciteRemboursement + " " +
56             tauxInteretAnnuel + " ";
57     }
58
59     // calcul mensualité emprunt
60     public double Calcul_Remboursement()
61     {
62         double remboursement = (double)capitalEmprunte *
63             (Calcul_taux_Annuel_Periodicite() / (1- Math.Pow((1+
64             Calcul_taux_Annuel_Periodicite(), -Calcul_Nombre_Mensualite(
65             ()))));
66         return remboursement;
67     }
68     // Calcul du taux Annuel
69     public double Calcul_taux_Annuel_Periodicite()
70     {
71         double tauxAnnuelPeriodicite = tauxInteretAnnuel * (int)
72             periodiciteRemboursement / 12;
73         return tauxAnnuelPeriodicite;
74     }
75     // Calcul du nombre de remboursement en fonction de la duree du
76     // credit et de la periodicité des remboursements
77     public int Calcul_Nombre_Mensualite()
78     {
79         int nbMensualite = dureeEmpruntMois / (int)
80             periodiciteRemboursement;
81         return nbMensualite;
82     }
83 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Net.NetworkInformation;
5  using System.Reflection.Metadata.Ecma335;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace ClassEmprunt
10 {
11     public class Emprunt
12     {
13         string nom;
14         public string Nom { get => nom; set => nom = value; }
15
16         int capitalEmprunte;
17         public int CapitalEmprunte { get => capitalEmprunte; set => capitalEmprunte = value; }
18
19         int dureeEmpruntMois;
20         public int DureeEmpruntMois { get => dureeEmpruntMois; set => dureeEmpruntMois = value; }
21
22         public enum EnumPeriodicite : ushort
23         {
24             Mensuel = 1,
25             Bimestriel = 2,
26             Trimestriel = 3,
27             Semestriel = 6,
28             Annuelle = 12,
29         }
30         EnumPeriodicite periodiciteRemboursement;
31         public EnumPeriodicite PeriodiciteRemboursement { get => periodiciteRemboursement; /*set => periodiciteRemboursement = value;*/ }
32
33         float tauxInteretAnnuel;
34         public float TauxInteretAnnuel { get => tauxInteretAnnuel; /*set => tauxInteret = Value;*/ }
35
36         /// <summary>
37         /// Constructeur classic
38         /// </summary>
39         /// <param name="nom"></param>
40         /// <param name="nom"></param>
41         /// <param name="dureeEmpruntMois"></param>
42         /// <param name="periodiciteRemboursement"></param>
43         /// <param name="tauxInteret"></param>
44         public Emprunt(string _nom, int _capitalEmprunte, int _dureeEmpruntMois, EnumPeriodicite _periodiciteRemboursement, float _tauxInteretAnnuel)
45         {
46             nom = _nom;
```

La durée en mois nombre du remboursement, donné par l'ascenseur doit s'afficher en clair ; si pour des remboursements mensuels, ce chiffre évolue de un en un, pour des remboursements trimestriels, par exemple, il évoluera de trois en trois.

Dès le chargement de la feuille, positionner le curseur sur le « nom », le taux d'intérêt sélectionné doit être « 7% », la durée en mois de remboursement doit être égale à « 1 », ainsi que le nombre de remboursements en Euros, et la sélection doit être « Mensuelle » dans la périodicité de remboursement.

A chaque changement de capital, de taux de périodicité et de durée de remboursement, ou de périodicité, le montant du remboursement sera réinitialisé.

Sur le clic du bouton OK, afficher un message d'erreur si le capital n'est pas saisi.

Dans la zone *capital emprunté*, il ne peut y avoir que des chiffres, on limite à 10 chiffres sans décimale cadrés à droite. Le champ Nom s'il est saisi, ne doit comporter que des caractères alphabétiques.

Un contrôle d'erreur peut être positionné en cas de saisie incorrecte.

On fait grossir le curseur de la barre horizontale en fonction de la périodicité.

Vous avez en votre possession l'exécutable du programme réalisé. Faites le *tourner* et .... *retourner*.

Etudiez tout d'abord l'interaction entre la liste Périodicité de remboursement et la scrollBar Durée du remboursement :

- Quel est l'effet de la sélection d'un item de la liste sur la ScrollBar ? : quelles sont les propriétés affectées ?
- Comment évolue le label qui contient la durée en mois lorsque le curseur de la ScrollBar est actionné ? Comment évolue-t-il lorsqu'on change la périodicité ?
- A quel moment est calculé le nombre de remboursements ?

Mettez en place les contrôles, le calcul du taux d'intérêt, puis le calcul du montant du remboursement.

Controlez vos résultats par rapport aux résultats donné par le programme.

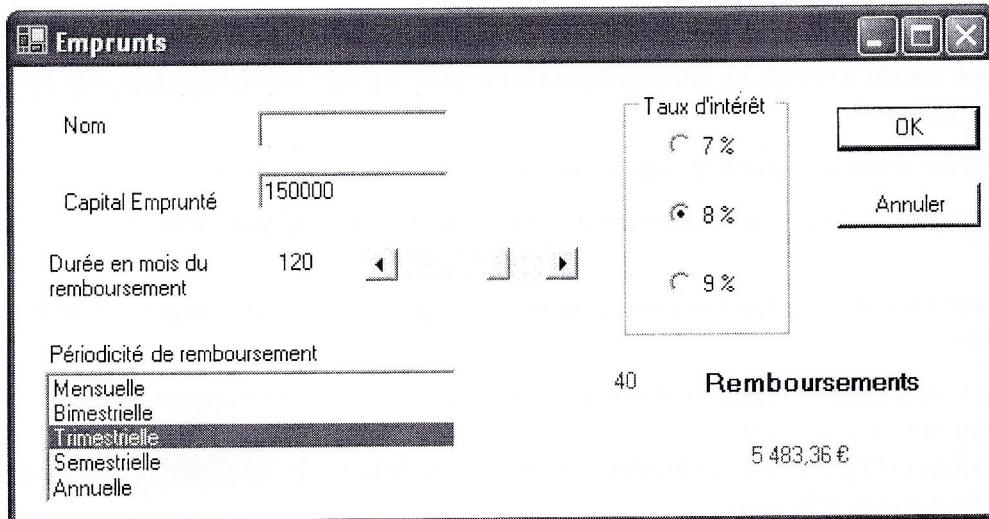
## III.5 SYNTHESE

### III.5.1 Objectifs

Savoir utiliser propriétés et méthodes des contrôles de base dans une application.

### III.5.2 But du traitement

Nous devons réaliser un petit programme avec une seule fenêtre pour calculer le montant périodique d'un prêt en fonction de son taux, de la périodicité de remboursement, de la durée du prêt et bien sûr de la somme empruntée.



### III.5.3 Les traitements

La formule de calcul est la suivante :

$$\text{Montant de remboursement} = K * \left( t / (1 - (1 + t)^{-n}) \right)$$

Où      K      : est le capital emprunté,  
t      : est le taux annuel,  
n      : est le nombre de remboursements.

#### **Attention**

Le taux donné par les boutons radio est un taux annuel, il faudra le diviser par 12 si l'on veut un taux mensuel.

Il peut y avoir des remboursements mensuels, bimestriels, trimestriels, semestriels ou annuels.

Le nom de la personne est saisi de manière facultative, par contre toutes les autres zones sont obligatoires. Il faut contrôler leur présence et leur validité.

### **Les propriétés utilisées :**

- Les propriétés Minimum, Maximum , SmallChange, LargeChange et Value du contrôle HsScrollBar.  
**Attention :** La valeur maximale pouvant être atteinte à l'exécution équivaut à la valeur de la propriété Maximum moins la valeur de la propriété LargeChange plus 1.
- Les propriétés Minimum, Maximum et Value du contrôle NumericUpDown.

Cet exercice peut être fait de la même façon avec des contrôles **TrackBar**.

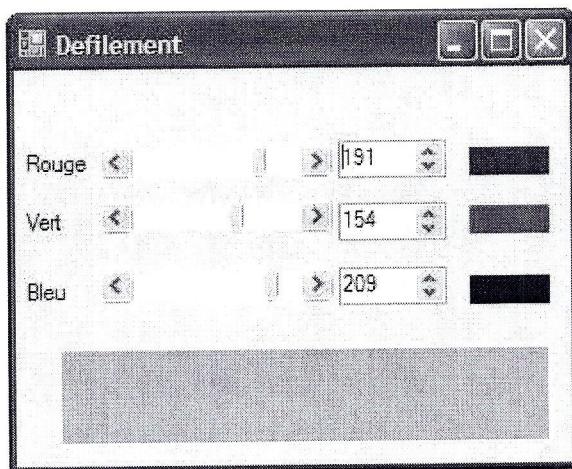
## III.4 LES COMPOSANTS DE DEFILEMENT

### III.4.1 Objectifs

Savoir utiliser les propriétés et méthodes des composants de défilement.

### III.4.2 But du traitement

Chaque barre de défilement (HScrollBar) permet de faire varier la couleur de chaque fondamentale, Rouge, Vert ou Bleu obtenant la couleur de l'étiquette positionnée en fin de ligne. La couleur résultant du mélange des trois est affichée à l'aide d'une quatrième étiquette au bas de la feuille.



### III.4.3 Les traitements

Chaque barre de défilement (HScrollBar) permet de faire varier la couleur de chaque fondamentale, Rouge, Vert ou Bleu obtenant la couleur de l'étiquette positionnée en fin de ligne.

La couleur résultant du mélange des trois est affichée à l'aide d'une étiquette au bas de la feuille.

L'utilisateur peut également obtenir la couleur désirée en positionnant dans les NumericUpDown de chaque ligne la valeur souhaitée.

#### Les évènements utilisés :

- L'évènement Scroll du contrôle HsScrollBar
- L'évènement ValueChanged du contrôle HsScrollBar
- L'évènement ValueChanged du contrôle NumericUpDown

#### Les méthodes utilisées :

- La méthode `Color.FromArgb(red, green, blue)`  
Où `red, green, blue` sont des valeurs comprises entre 0 et 255.



```
87         EventArgs e)
88     {
89         if (TextBoxSaisieNom.Text != "" && !Controles.NomAVerifier >
90             (TextBoxSaisieNom.Text))
91         {
92             errorProvider_nouvel_element.SetError(TextBoxSaisieNom, >
93                 "Veuillez-saisir uniquement des lettres et 1 '-' >
94                 pour les prenoms composés");
95         }
96     }
97
98     private void textBox_index_element_TextChanged(object sender, >
99         EventArgs e)
100    {
101        if (textBox_index_element.Text != "" && ! >
102            Controles.IndexElementAVerifier >
103            (textBox_index_element.Text))
104        {
105            errorProvider_index_element.SetError >
106                (textBox_index_element, "veuillez saisir un numero >
107                 entre 0 et " + (listBox_liste_de_nom.Items.Count - >
108                 1));
109        }
110
111        #endregion
112    }
113 }
114
```

```
        entre 0 et index count -1");
46    }
47    }
48
49    private void button_ajout_liste_Click(object sender, EventArgs e)
50    {
51        if (Controles.NomAVerifier(textBoxSaisieNom.Text))
52        {
53            if (listBox_liste_de_nom.FindStringExact
54                (textBoxSaisieNom.Text) == -1) // FindStringExact
55                compare ne nom saisie et les noms dans la liste
56            {
57                listBox_liste_de_nom.Items.Add
58                    (textBoxSaisieNom.Text);
59                textBox_items_count.Text =
60                    listBox_liste_de_nom.Items.Count.ToString();
61                textBoxSaisieNom.Clear();
62                textBoxSaisieNom.Focus();
63                errorProvider_nouvel_element.Clear();
64            }
65        }
66    }
67
68    private void button_vider_liste_Click(object sender, EventArgs e)
69    {
70        listBox_liste_de_nom.Items.Clear();
71        textBox_items_count.Text =
72            listBox_liste_de_nom.Items.Count.ToString();
73        textBox_text.Clear();
74        textBox_selected_items.Clear();
75    }
76
77    #endregion
78
79    #region mesTextBoxs
80
81    private void listBox_liste_de_nom_SelectedIndexChanged(object
82        sender, EventArgs e)
83    {
84        textBox_selected_items.Text =
85            listBox_liste_de_nom.SelectedIndex.ToString();
86        textBox_text.Text = listBox_liste_de_nom.Text;
87    }
88
89    private void textBox_saisie_liste_TextChanged(object sender,
```

```
1  using ClassLibraryControle;
2  using System;
3  using System.Collections;
4  using System.Collections.Generic;
5  using System.ComponentModel;
6  using System.Data;
7  using System.Drawing;
8  using System.Linq;
9  using System.Text;
10 using System.Text.RegularExpressions;
11 using System.Threading.Tasks;
12 using System.Windows.Forms;
13
14 namespace WinFormsOperationBaseListBox
15 {
16     public partial class FormLes_listesEtLeurPropriete : Form
17     {
18         public FormLes_listesEtLeurPropriete()
19         {
20             InitializeComponent();
21         }
22
23         #region mesBoutons
24
25         private void button_selectionner_Click(object sender, EventArgs e)
26         {
27             if (Controles.IndexElementAVerifier
28                 (textBox_index_element.Text))
29             {
30                 int idSelectionne = int.Parse
31                     (textBox_index_element.Text);
32                 if (idSelectionne < listBox_liste_de_nom.Items.Count)
33                 {
34                     listBox_liste_de_nom.SelectedIndex = idSelectionne;
35                     textBox_selected_items.Text =
36                         textBox_index_element.Text;
37                     textBox_text.Text = listBox_liste_de_nom.Text;
38                     textBox_index_element.Clear();
39                     errorProvider_index_element.Clear();
40                 }
41                 else
42                 {
43                     errorProvider_index_element.SetError
44                         (textBox_index_element, "veuillez saisir un numero
entre 0 et " + (listBox_liste_de_nom.Items.Count -
1));
45                 }
46             }
47             else
48             {
49                 errorProvider_index_element.SetError
50                     (textBox_index_element, "veuillez saisir un numero
entre 0 et " + (listBox_liste_de_nom.Items.Count -
1));
51             }
52         }
53     }
54 }
```



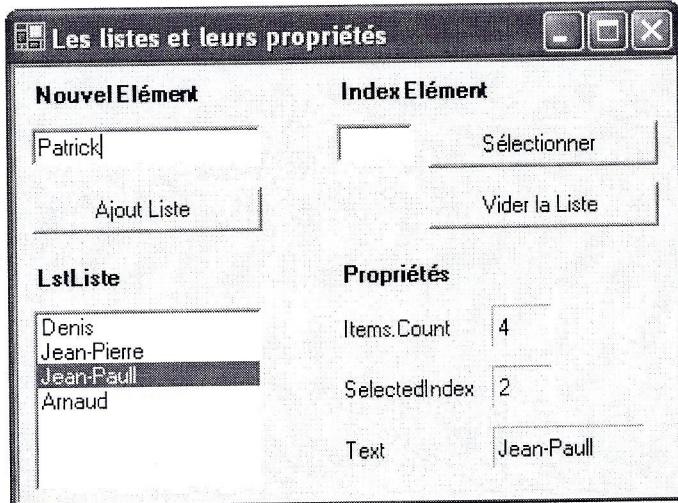
## III.2 OPERATIONS DE BASE SUR LES LISTBOX

### III.2.1 Objectifs

Etre en mesure de réaliser les opérations principales concernant les contrôles de type **ListBox**.

### III.2.2 But du traitement

Visualiser les propriétés de la ListBox dans son état vide, lors de l'ajout ou la sélection d'un élément



### III.2.3 Les traitements

Lors du lancement de l'application, la liste **LstListe** est vide. Le peuplement de la liste s'effectue en saisissant des valeurs dans la TextBox **Nouvel Elément** puis en pressant le bouton **Ajout Liste**. Une fois un nouvel élément ajouté, le focus se repositionne sur **Nouvel Elément** qui est remise à blanc.

Lorsque l'utilisateur sélectionne un élément, les propriétés affectées s'affichent à droite. Il est également possible de sélectionner un élément en saisissant son N° dans la TextBox **Index Elément** et en pressant le bouton **Sélectionner**.

Enfin, le bouton **Vider la liste** comme son nom le laisse supposer remet la liste à blanc.

#### Si vous avez du temps

Faites en sorte qu'un utilisateur maladroit ne puisse pas provoquer une erreur d'exécution de votre application. Exemple : s'il saisit un N° d'élément qui n'existe pas ou sous forme de texte.

Assurez-vous également qu'une valeur ne puisse pas être ajoutée deux fois dans la liste.

- La coche de la case à cocher **Couleur du fond** fait apparaître le groupe d'options **Fond**. Un élément coché dans ce groupe d'options modifie la couleur de fond du label (propriété BackColor).
- La coche de la case à cocher **Couleur des caractères** fait apparaître le groupe d'options **Caractères**. Un élément coché dans ce groupe d'options modifie la couleur de la police du texte recopié (propriété ForeColor).
- La coche de la case à cocher **Casse** fait apparaître le groupe d'options **Casse**. Un élément coché dans ce groupe d'options fera passer en minuscules/majuscules le texte recopié.

La décoche de chaque case à cocher provoquera la disparition du groupe d'options correspondant.

On utilisera la structure `System.Drawing.Color`.

### III PHASE 3 : LES DIFFERENTS OBJETS GRAPHIQUES

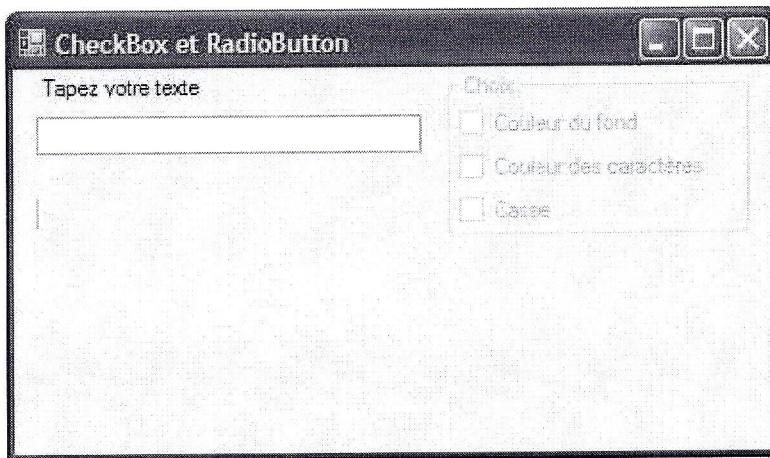
#### III.1 CHECK BOX ET BOUTONS RADIOS

##### III.1.1 Objectif

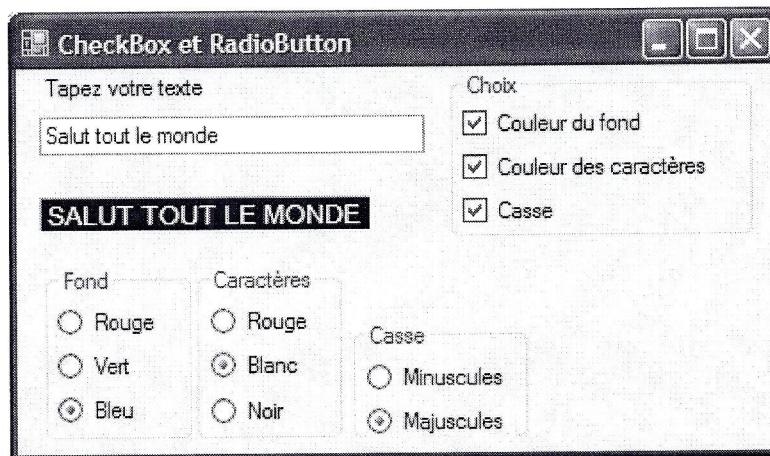
Savoir gérer des cases à cocher et groupes de boutons radios.

##### III.1.2 But du traitement

Au lancement du programme, la feuille se présente de la façon suivante:  
Seule est accessible, la zone de texte où l'utilisateur peut saisir son texte



Puis en cours de programme, en fonction de l'option de choix sélectionnée, on pourra faire varier la couleur de fond, la couleur des caractères et la casse.



##### III.1.3 Les traitements

La feuille initiale permet uniquement de saisir un texte, recopié caractères à caractères dans un label.

Au premier caractère entré, le groupe de choix est rendu accessible et devient inaccessible si le caractère est supprimé.

## II PHASE 2 : LA VALIDATION DE LA SAISIE

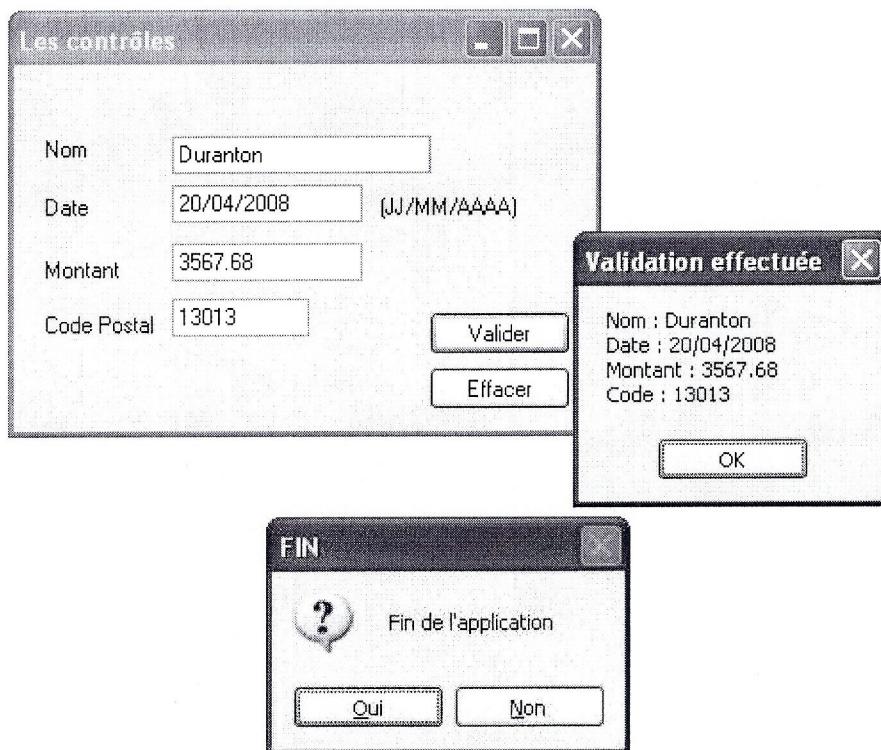
### II.1 LES CONTROLES DE SAISIE

#### II.1.1 Objectif

Savoir gérer la validité d'une zone de saisie alphabétique, numérique ou date.

#### II.1.2 But du traitement

Gérer la saisie dans une feuille comportant un champ alphabétique, un champ numérique entier, un champ comportant une date et un champ monétaire (voir feuille ci-dessous).



#### II.1.3 Les traitements

La feuille n'est pas redimensionnable.

Les 4 champs sont **obligatoires** :

- txtNom.Text n'accepte que des caractères alphabétiques, champ de longueur 30 maxi.
- txtDate.Text accepte, bien sur une date valide, supérieure à la date du jour.
- txtMontant.Text accepte chiffres et séparateur décimal, et refuse les nombres négatifs.
- txtCP.Text n'accepte que 5 caractères numériques.

Le bouton « Effacer » efface les zones de saisie.

Le bouton « Valider» valide la saisie, et envoie un message de bonne fin.  
La croix sert à quitter l'application : Envoyez une MessageBox de confirmation de fin d'application, qui permet de revenir dans le traitement.

Si une saisie n'est pas correcte, il faut envoyer un signal sonore, afficher un message d'erreur, positionner le focus sur le champ en erreur, bref, avertir l'utilisateur.

Plusieurs méthodologies différentes sont à tester à travers la réalisation de ce TP :

1. On peut contrôler tous les champs sur le click du bouton Valider. Réaliser ce premier scénario en choisissant les **bonnes** méthodes de contrôle pour chaque champ.
2. On peut également contrôler la saisie champ après champ, le bouton Valider étant clickable pour effectuer les « traitements métier ». Il faut alors choisir les **bons** évènements, peut-être aussi les **bons** contrôles graphiques .....

**Quels traitements sur quels évènements préconisez-vous ?**



## I PHASE 1 : PREMIERS PAS

### I.1 L'ADDITIONNEUR

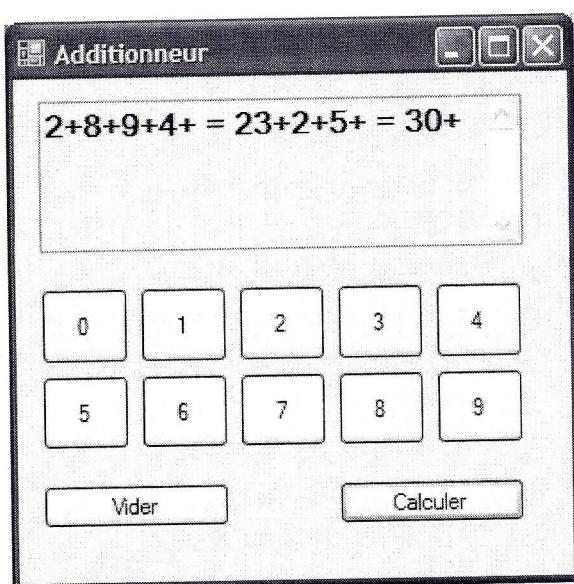
#### I.1.1 Objectif

Prendre en main l'environnement de développement.

S'initier à la programmation événementielle.

#### I.1.2 But du traitement

Programmer un additionneur par simple click sur les boutons de l'interface.



#### I.1.3 Les traitements

L'utilisateur ne peut additionner que des chiffres.

A chaque clic sur un chiffre, le chiffre suivi de " + " est ajouté à la TextBox (multiligne et avec scrollbars).

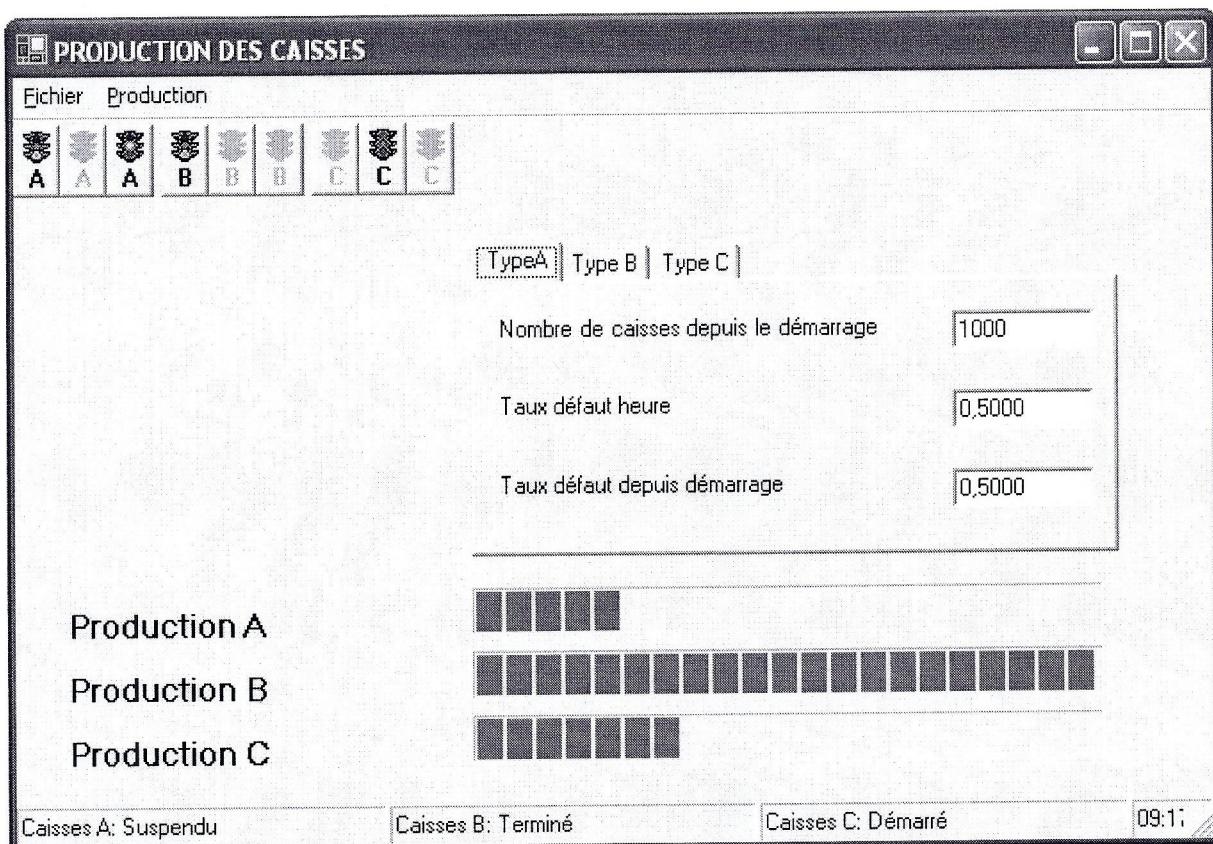
En cliquant sur le bouton **Calculer**, le résultat final de l'addition est affiché (suivi de " + " lui aussi).

En cliquant sur le bouton **Vider**, on vide la TextBox et remet à zéro le compteur. Le contenu de la TextBox ne peut pas être modifié autrement que par les boutons.

On quitte le programme en cliquant sur la croix de fermeture de fenêtre.

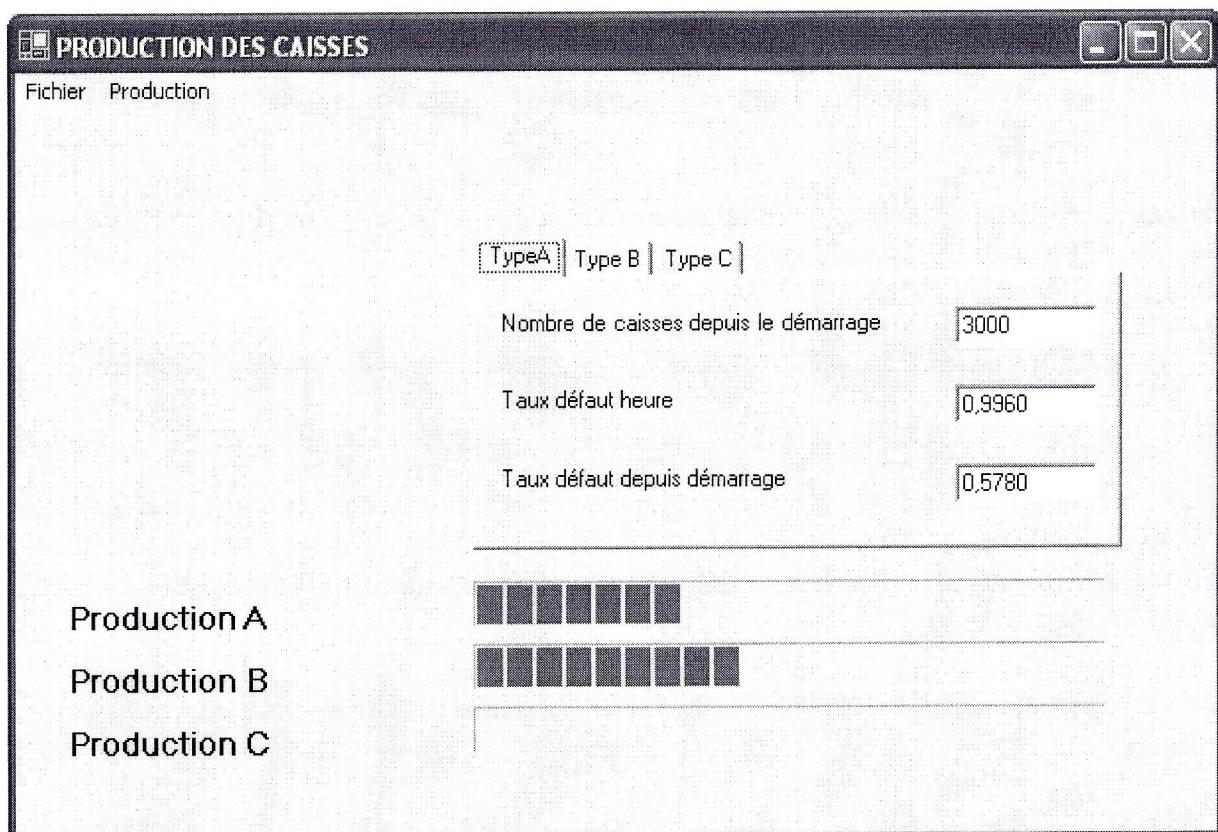


- Une barre d'état présentant l'état de la production de chaque type de caisse (Démarré, Suspendu, Redémarré ou Terminé) ainsi que l'heure courante.



## Exercice Tout Embal : Evènementiel et Thread

La société TOUTEMBAL désire suivre sa production de caisses en bois. La société produit des caisses de types A, B et C.  
L'évolution de la production doit être suivie en temps réel.



Pour chaque type de caisses sont affichés:

- -Le nombre de caisses produites depuis le démarrage
  - -Le taux de défaut de la dernière heure (avec 4 décimales)
  - -Le taux de défaut cumulé depuis le démarrage de la production (avec 4 décimales)
- Les informations seront présentées sous la forme d'onglets (un onglet par type)

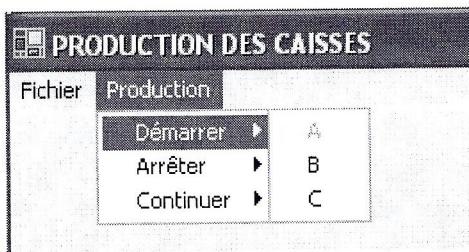
Les productions horaires sont les suivantes:

- Caisse A: 1000 caisses/heure
- Caisse B: 5000 caisses/heure
- Caisse C: 10000 caisses/heure

Une barre de menus présentera 2 options décomposées comme suit:

- Fichier  
    → Quitter termine l'application

→ Production  
    Ce menu présente les fonctions de démarrage, suspension (Arrêter) et reprise (Continuer) des productions de chaque type de caisses



La production des différents types de caisses est planifiée de façon personnalisée (**démarrage** (avec réinitialisation de la production précédente), **arrêt** (=suspension), **reprise après arrêt**).

La progression de la production des caisses de types A, B et C est représentée dans 3 barres de production. Le nombre de caisses à produire pour chaque type est le suivant:

- **Caisse de type A:** 10000
- **Caisse de type B:** 25000
- **Caisse de type C:** 120000

Dans le menu Production, les options seront rendues opérantes ou inopérantes en fonction du contexte. Par exemple, on ne pourra pas arrêter ou continuer une production qui n'a pas démarré, on ne pourra pas continuer une production qui n'a pas été arrêtée ou qui n'a pas démarré ...

Lorsqu'une production est terminée pour un type donné, un message d'information sera affiché.

#### Exercice 1 :

Développer cette application en faisant attention de séparer l'IHM des classes métiers (class Production et éventuellement class Caisse).

Le rythme de la production sera donné par le Timer des controls de la boîte à outils. Un évènement sera implémenté afin de mettre à jour l'IHM en fonction de la cadence de la production.

#### Exercice 2 :

La classe Production aura son propre Thread lui permettant de donner le rythme à la production

#### Exercice 3: Barre d'outils, barre d'état.

Afin de rendre plus conviviale l'interface de l'application, ajouter:

- Une barre d'outils présentant les fonctions du menu production