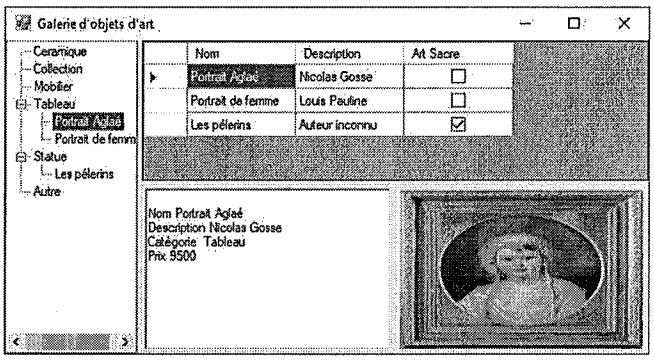
Licence Informatique **Mardi 15 juin 2021**

Conception et développement avancé d'applications **Examen - Durée 2h**
Documents CM, TD et TP autorisés

Dans le cadre du développement d'une application de vente en ligne d'objets d'art, il a été décidé de créer un prototype pour l'avant-vente du projet. Ce prototype a été développé en Windows Forms (Framework .net), en langage C#, et sous Visual Studio 2019.
L'application comporte 3 classes :
- une classe « ObjetArt » qui décrit de façon simplifiée un objet d'art avec le nom de l'objet (supposé unique), sa description, un prix et une photo.
- une classe « ObjetArtSacré » qui décrit de façon simplifiée un objet d'art qui est un objet d'art mais qui comporte en plus, une description de son histoire (religieuse).
- la classe principale nommée « Form1 » qui comporte une liste de tous les objets d'art vendus par la galerie.
L'interface de l'application est la suivante :



L'application comporte
- à gauche, une description des noms des objets d'art classés sous forme arborescente (nommé « ArbreCat » de type TreeView) selon leur catégorie,
- en haut à droite une grille (nommée « Grille » de type DataGridView) qui représente la liste des objets d'art (avec uniquement leur nom, leur description et s'il s'agit d'objet d'art sacré ou non), e
- en bas une zone d'édition (nommée « Edition » de type RichTextBox) et une composant (nommé « ImageObjet » de type PictureBox) qui permettent d'afficher les informations complètes et la photo de l'objet sélectionné soit dans la grille, soit dans l'arborescence.

Exercice 1 : C# (5 points)

Les classes «ObjetArt » et « ObjetArtSacre » - Annexe 1
L'annexe 1 décrit partiellement la classe « ObjetArt ». En utilisant cette annexe :
1. **(2 pts)** Expliquer la portion de code suivante : « `public class ObjetArt : IComparable<ObjetArt>` » et compléter la classe si besoin pour que cette indication soit correcte ou expliquer si ce n'est pas nécessaire.
2. **(3 pts)** Donner le code complet de la classe « ObjetArtSacre » qui dérive de la classe « ObjetArt ». Expliquer les éléments décrits dans votre classe.

Exercice 2 : Application Windows Forms (12 points)

La classe « Form1 » - Annexe 2
Note : Seules les méthodes proposées dans le code des annexes sont à utiliser. Si vous jugez nécessaire d'utiliser d'autres méthodes, décrivez leur code complet en C#.
L'annexe 2 décrit partiellement la classe principale nommée « Form1 ». L'ensemble des objets d'art sont gérés à l'aide d'une liste nommée « *lc* » de type « List<ObjetArt> ». Les photos des objets sont gérées en ressources dans le projet.
Le constructeur de la classe est la suivant :

```
public Form1()
 {
     InitializeComponent();
     lc = new List<ObjetArt>();
     InitListeObjetArts();
     InitArbre();
     InitGrille();
 }
```

3. **(1.5 pts)** Indiquer sous une forme au choix, les composants qui permettraient de structurer l'interface de l'application comme décrite ci-dessus.
4. **(2 pts)** La méthode « *InitListeObjetArts* » permet de remplir la liste des objets à des fins de tests de l'application. Donner le code de cette méthode permettant l'ajout d'un objet d'art dont la catégorie est « tableau », et un objet d'art sacré dont la catégorie est « statue » avec des valeurs aux choix.

```
public void InitListeObjetArts(){ à compléter}
```

5. **(2.5 pts)** La méthode « *InitArbre* » permet la création et le remplissage de l'arborescence de description des objets d'art de la galerie classés selon leur catégorie. Donner le code de cette méthode.
6. **(2.5 pts)** La méthode « *InitGrille* » permet de créer les colonnes et remplir la grille (*de type DataGridView et nommée Grille*) avec certaines informations des objets : son nom, sa description et s'il s'agit d'un objet d'art sacré ou non.
    a. Expliquer la partie de code décrite dans l'annexe 2 pour cette méthode.
    b. Compléter cette méthode pour prendre en compte la 3ème colonne et remplir la grille.
7. **(2 pts)** Le clic sur le nom d'un objet d'art dans l'arborescence permet l'affichage de ses informations complètes dans la zone d'édition et de sa photo sur le composant nommé « ImageObjet » de type « PictureImage ». Donner le code de ce gestionnaire d'évènement.
   `private void ArbreCat_AfterSelect(object sender, TreeViewEventArgs e)) { à completer}`
8. **(1.5 pt)** Le clic sur une case de la grille réalise les mêmes affichages que la sélection dans l'arborescence. Donner le code de ce gestionnaire d'évènement.
   `private void Grille_CellContentClick(object sender, DataGridViewCellEventArgs e) { à compléter}`

Exercice 3 : Application ASP.NET MVC (3 points)

Une 2ème version du prototype a été implémentée sous forme d'application Web ASP.net selon le modèle MVC en utilisant l'environnement Visual Studio 2019.

Expliquer le principe du modèle d'architecture « Modèle – Vue – Controller » et comment il est mis en œuvre dans les applications Web ASP.net dans cet environnement.

ANNEXE 1 – Classe « ObjetArt » et type énuméré « Catégorie »

```
public enum Categorie { Ceramique, Collection, Mobilier, Tableau, Statue, Autre }
```

```
public class ObjetArt : IComparable<ObjetArt>
    {   private string nom;
        private string description;
        private Categorie cat;
        private float prix;
        private Image photo;

        public ObjetArt()
        {
            this.nom = "";
            this.description = "";
            this.cat = Categorie.Autre;
            this.prix = 0;
        }

        public ObjetArt(string n, string d, Categorie ty, float p)
        {
            this.nom = n;
            this.description = d;
            this.cat = ty;
            this.prix = p;
        }

        public Image Photo
        { get { return this.photo; } set { this.photo = value; } }
        public string Nom
        { get { return this.nom; } set { this.nom = value; } }

        public string Description
        { get { return this.description; } set { this.description = value; } }

        public Categorie Cat { get { return this.cat; } set { this.cat = value; } }

        public float Prix
        { get { return this.prix; } set { this.prix = value; } }

        public string catS
        {
            get { return Enum.Format(typeof(Categorie), this.cat, "g"); }
            set { this.cat = (Categorie)Enum.Parse(typeof(Categorie), value, false); }
        }

        public override string ToString()
        {
            string s = "";
            s += "\nNom " + this.Nom;
            s += "\nDescription " + this.description;
            s += "\nCatégorie  " + this.catS;
            s += "\nPrix " + this.prix;
            return s;
        }

    . . .
```

ANNEXE 2 – Classe « Form1»

```
public partial class Form1 : Form
{
    private List<ObjetArt> lc;

     public Form1()
    {
        InitializeComponent();
        lc = new List<ObjetArt> ();
        InitListeObjetArts();
        InitArbre();
        InitGrille();
    }

   public void InitListeObjetArts()
    { // A COMPLETER }

    public void InitGrille()
    {
            DataGridViewTextBoxColumn Nom = new DataGridViewTextBoxColumn();
            DataGridViewTextBoxColumn Description = new DataGridViewTextBoxColumn();
            Nom.HeaderText = "Nom";
            Nom.Name = "Nom";
            Nom.ReadOnly = true;

            Description.HeaderText = "Description";
            Description.Name = "Description";
            Description.ReadOnly = true;

           Grille.Columns.AddRange(new DataGridViewColumn[] { Nom, Description});

   // A COMPLETER / MODIFIER
          . . .
   }

     private void ArbreCat_AfterSelect(object sender, TreeViewEventArgs e)
   { // A COMPLETER }

    private void Grille_CellContentClick(object sender, DataGridViewCellEventArgs e)
    { // A COMPLETER }

   . . .
}
```

ANNEXE 3 – Application Web ASP.net - modèle MVC
Fichier « _Layout.cshtml » du squelette d'application

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Mon application ASP.NET</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Nom de l'application", "Index", "Home", new { area = "" },
new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Accueil", "Index", "Home")</li>
                    <li>@Html.ActionLink("À propos de", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

ANNEXE 3 – Application Web ASP.net - modèle MVC
Fichier « _Layout.cshtml » du squelette d'application

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Mon application ASP.NET</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Nom de l'application", "Index", "Home", new { area = "" },
new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Accueil", "Index", "Home")</li>
                    <li>@Html.ActionLink("À propos de", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```