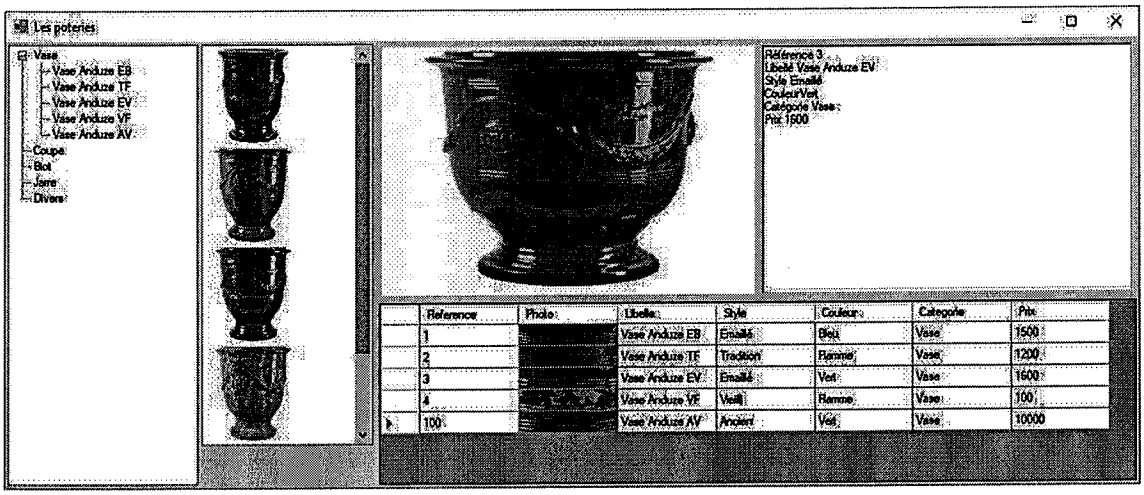
Licence Informatique - L3 **Lundi 6 janvier 2020**

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

Une boutique de poteries a mis en place un outil numérique pour permettre à ses clients de rechercher et visualiser les poteries disponibles dans la boutique. La boutique vend des poteries contemporaines et des poteries anciennes. Les poteries appartiennent à différentes collections : Vase, Coupe, Biot, Jarre ou Autre. Chaque poterie est référencée, a un libellé, un style, une couleur, un prix et une photo. Une poterie ancienne possède en complément une datation.
Un prototype a été développé en langage C# sous forme d'une application Windows Forms dont l'interface est la suivante.



Cette interface comporte plusieurs zones :

- La 1ère zone à gauche contient un arborescence (TreeView nommé Arbre) qui permet de présenter les poteries selon leur collection avec leur libellé
- La 2ème zone contient une liste d'images de toutes les poteries de la boutique (ListBox nommé ListePhotos)
- La 3ème zone en bas à droite permet d'afficher les informations des poteries sous forme d'une grille (DataGridView nommée Grille)
- La 4ème zone en haut à droite comporte à gauche la photo de la poterie sélectionnée et à droite son détail textuel. Elle contient à gauche un composant PictureBox nommé Photo et à droite un composant RichTextBox nommé Edition.
  La sélection d'une poterie peut se faire soit en cliquant sur le libellé de la poterie dans l'arborescence, soit en cliquant dans une ligne dans la grille ou soit en cliquant dans la liste d'images.

Dans un 1er temps, l'application comporte 2 classes qui modélisent les poteries et une application principale qui gère une liste de poteries.

- Les classes « Poterie » et « PoterieAncienne » sont partiellement décrites dans l'annexe 1.
- La classe principale appelée « **Form1** » gère l'ensemble des poteries à l'aide d'une liste. L'annexe 2 donne une description partielle de cette classe.

Exercice 1 : C# (4 points)

Les classes « Poterie » et « PoterieAncienne » - Annexe 1

1. **(1 pt)** Donner le code du constructeur avec paramètre de la classe « **PoterieAncienne** » qui initialise tous les attributs excepté la photo qui a une valeur par défaut gérée en ressource (« Properties.Resources.PotDefaut »).
2. **(1,5 pts)** On souhaite compléter la classe « Poterie » pour permettre la comparaison de deux poteries, qui sont considérées comme égales si elles ont la même référence et la même collection.
   a. Expliquer l'intérêt d'avoir une telle méthode.
   b. Donner en C# une méthode de comparaison en expliquant votre choix et votre code.
3. **(1,5 pts)** La modélisation des informations à l'aide d'une hiérarchie de classe vous paraît-elle justifiée ? Argumentez votre réponse (que cela soit oui ou non).

Exercice 2 : Application Windows Forms (13 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, décrire leur code complet en C#.

4. **(1,5 pts)** L'ensemble des poteries est géré à l'aide d'une liste dans la classe principale « Form1 ». Ce choix est-il justifié ? Une autre modélisation aurait-elle été possible ? Expliquer les avantages et inconvénients du choix proposé et des autres possibilités le cas échéant.

5. **(1,5 pts)** Expliquer **quels composants conteneurs** ont été utilisés selon vous, pour construire l'interface décrite ci-dessus. Indiquer brièvement leur organisation sous une forme au choix (phrases ou schéma).

Lors du lancement de l'application, les méthodes d'initialisation « InitArbre » et « InitGrille » sont appelées dans le constructeur pour « remplir » les composants (Arbre et Grille). Une méthode « InitListe » permet de créer des poteries pour tester l'application.

L'annexe 2 décrit partiellement la classe « Form1 ». En utilisant cette annexe, répondre aux questions suivantes.

6. **(1,5 pts)** Donner la partie de code de la méthode « InitListe » qui permet de créer et d'ajouter dans la liste, la poterie ancienne présentée sur l'interface ci-dessus. La photo de cette poterie est en ressource dans le projet et s'appelle « Vase5 ».
7. **(2 pts)** Initialisation de la liste des photos.
   a. Expliquer les différentes étapes nécessaires pour initialiser la liste des photos des poteries.
   b. Donner les parties de code correspondantes en indiquant précisément où ces portions de code, méthodes ou gestionnaires d'événement doivent être décrits.
8. **(2 pts)** Donner le code de la méthode « InitArbre » qui remplit le composant de type TreeView nommé Arbre, avec les libellés des poteries classées par collection. On conservera pour chaque nœud le libellé, et la référence de la poterie dans la propriété « Name » du nœud.
9. **(2 pts)** Expliquer la méthode « InitGrille » qui remplit la grille avec les informations des poteries. Indiquer clairement, compte tenu du code fourni, à partir de quelles informations cette grille est créée et justifier également l'ordre de description des différentes colonnes. Indiquer également pourquoi la colonne de datation n'apparaît pas pour la poterie ancienne.

Le clic dans une cellule de la grille produit l'affichage des informations textuelles de la poterie dans la zone d'édition (de type RichTextBox nommée Edition) et de sa photo dans le PictureBox nommé Photo.

10. **(2,5 pts)**
    a. Lors de l'ajout de cet évènement le code ci-dessous a été généré (sous Visual Studio). Expliquer ce code
       « this.Grille.CellClick += new DataGridViewCellEventHandler(this.Grille_CellClick)»
    b. L'entête du gestionnaire correspondant a également été généré. Expliquer ses paramètres.
       « private void Grille_CellClick(object sender, DataGridViewCellEventArgs e)»
    c. Donner le code complet du gestionnaire.

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.

Cet environnement permet la génération d'un squelette d'application qui comporte une barre de menu avec 3 options : « Accueil » « A propos de » et « Contacts ».

Les annexes 3, 4 et 5 donnent le code des fichiers « Layout.cshmtl », « HomeController.cs » et « About.cshtml » générés dans ce squelette.

1. (1,5 pts) Expliquer le modèle d'architecture « Modèle – Vue – Controller » en détaillant le fonctionnement de la gestion du clic sur l'option « Contacts » du squelette de l'application.
   On précisera clairement sur cet exemple les parties de code concernées, la façon dont les événements sont déclenchés, les informations passées, etc.
   Vous pouvez utiliser les annexes pour les annoter si vous le souhaitez et les rendre avec votre copie.
2. (1,5 pts) Indiquer quelles seraient les étapes nécessaires pour mettre en place une nouvelle fonctionnalité dans cette architecture comme par exemple la visualisation de la liste des poteries (sans nécessairement donner le code complet).

ANNEXE 1 – Classes « Poterie » et « PoterieAncienne »

```
public enum Collection { Vase, Coupe, Biot, Jarre, Divers };
```

```
public class Poterie
    {  private int reference;
       private string libelle;
       private string style;
       private String couleur;
       private Collection cat;
      private float prix;
      private Image photo;
       public Poterie()
            { this.Reference = -1; this.Libelle = "Inconnu"; this.Style = "";
                this.Categorie = Collection.Divers;
                this.Couleur = "";
                this.Photo = Properties.Resources.PotDefaut;
       }
       public Poterie(int r, string l, string s, string c, Collection ca, float p)
            {   this.Reference = r; this.Libelle = l; this.Style = s; this.Couleur = c;
                this.Categorie = ca;
                this.Prix = p;
                this.Photo = Properties.Resources.PotDefaut;
            }
       public int Reference{ get { return this.reference; } set { this.reference = value; } }
       public Image Photo{ get { return this.photo; } set { this.photo = value; } }
       public string Libelle{ get { return this.libelle; } set { this.libelle = value; } }
       public string Style{ get { return this.style; } set { this.style = value; } }
       public string Couleur{ get { return this.couleur; } set { this.couleur = value; } }
       public Collection Categorie { get { return this.cat; } set { this.cat = value; } }
       public float Prix{ get { return this.prix; } set { this.prix = value; } }

       public override string ToString()
            {
                string s = "";
                s += "Référence " + this.Reference;
                s += "\nLibellé " + this.Libelle;
                s += "\nStyle " + this.Style;
                s += "\nCouleur" + this.Couleur;
                s += "\nCatégorie " + this.Categorie.ToString();
                s += "\nPrix " + this.Prix + "\n";
                return s;
            }
       . . .
}
```

```
public class PoterieAncienne : Poterie
    {   private DateTime date;
        public DateTime Datation { get { return this.date; } set { this.date = value; } }

        public PoterieAncienne() : base()
        {     this.date = new DateTime();
        }

         . . .

        public override string ToString()
        {     string s = base.ToString();
              s += "\nDatation : " + this.date;
               return s;
        }
       }
```

}

ANNEXE 2 – Classe « Form1»

```
public partial class Form1 : Form
{
    private List<Oeuvre> listePot;
    private ImageList listPh;

    public Form1()
    {
        InitializeComponent();
 this.listePot = new List<Poterie>();
  this.InitListe(); // initialisation de la liste des poteries (listePot)

 // A COMPLETER
   . . .

  this.InitArbre(); // initialization de arborescence
       this.InitGrille(); // initialisation de la grille

    }

    public void InitGrille()
    {   // A EXPLIQUER
        Grille.DataSource = listePot;
    }
          Poterie rechPoterie
    public ~~Oeuvre~~ ~~rechOeuvre~~(int r)   (Poterie o in listPot)
    {   foreach (~~Oeuvre~~ o in ~~listo~~)
            if (o.Reference == r) return o;
          return null;
    }

    public void InitListe()
    {   // A COMPLETER      }


    public void InitArbre()
    {   // A COMPLETER      }


    private void Grille_CellClick(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 4 – Application Web ASP.net - modèle MVC
Fichier « HomeController.cs » du squelette d'application

```
public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }
    }
```

ANNEXE 5 – Application Web ASP.net - modèle MVC
Fichier « About.cshtml » du squelette d'application

```
@{
    ViewBag.Title = "About";
}
<h2>@ViewBag.Title.</h2>
<h3>@ViewBag.Message</h3>

<p>Use this area to provide additional information.</p>
```