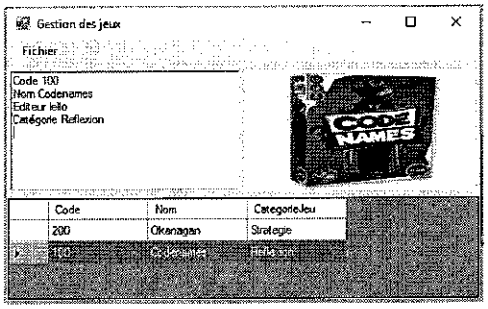
Licence Informatique – L3
Conception et développement avancé d'applications
Documents CM, TD et TP autorisés

Lundi 8 janvier 2018
Examen - Durée 2h

Exercice 1 : Application Windows Forms (13 points)

Un magasin de jeu souhaite développer une application permettant de gérer les produits en vente. Dans un premier temps un étudiant en stage a développé une application dont l'interface est la suivante :



Cette application est un prototype permettant de tester quelques fonctionnalités qui seront incluses dans l'application finale.
L'interface comporte un menu « Fichier » avec 2 options « Ouvrir », « Sauvegarder » qui permettent de charger les informations des jeux à partir d'une base de données dans l'application dans une liste, ou de sauvegarder ces informations à partir de la liste dans la base de données.
L'interface est composée de 3 « zones » :

- La 1ère zone en bas contient une Grille (DataGridView appelée Grille) qui donne la liste des produits qui sont des jeux avec uniquement le code, le nom et la catégorie du jeu.
- La 2ème zone en haut à droite permet d'afficher la photo du jeu qui a été sélectionné dans la grille (avec un clic souris en début de ligne), elle comporte un composant de type PictureBox appelé Photo.
- La 3ème zone en haut à gauche affiche les informations complètes du jeu avec son code, son nom et sa catégorie mais également son éditeur. Elle comporte un composant de type RichTextBox appelé Edition.

Dans un 1er temps, l'application ne comporte que 2 classes :

- La classe « **Jeu** » qui modélise un jeu avec un code, un nom, une catégorie (sélectionnée dans une liste énumérée), un éditeur et une photo. L'annexe 2 donne une description partielle de cette classe.
- La classe principale appelée « **Form1** » qui gère l'ensemble des jeux à l'aide d'une liste. L'annexe 3 donne une description partielle de cette classe.

La classe « Jeu » - Annexe 2

1. **(1,5 points)** Compléter la classe « Jeu» pour qu'elle implémente l'interface **« IComparable »** en expliquant le code ajouté. La comparaison des jeux est faite sur leur nom.
2. **(1 point)** Donner un ensemble d'instructions (qui pourraient être dans le programme principal d'une application console) montrant l'usage de cette méthode sur des jeux.

La classe « Form1 » - Annexe 3

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#.

La liste des jeux est gérée dans une liste générique de type List<Jeu>. Des attributs liés à la gestion de la base de données sont également déclarés dans la classe.
Lors du lancement de l'application, 3 méthodes « InitConnect », « InitListeBD » et « InitGrille » sont appelées dans le constructeur (cf annexe 3). La méthode « InitConnect » réalise la connexion à la base de données Oracle pour un utilisateur nommé « util100 ».
La base de données décrite en **Annexe 1** comporte les informations sur les jeux dans un table jeu. Dans la base de données, un jeu est décrit par un code en entier (integer), et les autres colonnes : nom, editeur, categorie et nomPhoto sont gérées en chaînes de caractères (varchar). Le nom de la photo (nomPhoto) correspond au nom de la ressource image de la photo, stockée dans les ressources du projet de l'application. La catégorie correspond à un des champs du type énuméré des catégories.

3. **(4,5 points)** Donner le code C# de la méthode « InitListeBD » en commentant les instructions, qui permet de :
    a. Remplir la table « dt » de type DataTable (déclarée en attribut) avec les informations de la table « Jeu » de la base de données
    b. Parcourir cette table, et pour chaque ligne de la table
        i. Récupérer les différentes informations du jeu, en convertissant les types le cas échéant pour les adapter aux attributs de la classe « Jeu ».
        ii. Créer un nouveau jeu
        iii. Ajouter ce jeu à la liste des jeux

    On rappelle que pour obtenir une image appelée « img » de type « Image », avec le nom d'une ressource d'image donnée en chaîne de caractères appelée « nom », on peut utiliser l'instruction suivante : *Image img = Properties .Resources.ResourceManager.GetObject(nom) ;*

4. **(2 points)** Donner le code C# de la méthode « **InitGrille** » qui remplit la grille avec les informations qui sont dans la liste de jeu (sans utiliser la table DataTable) en masquant les colonnes correspondant à l'éditeur et à la photo. La propriété « Visible » de la colonne considérée peut être utilisée pour masquer la colonne.
5. **(4 points)** Un clic dans n'importe quelle case de la grille permet l'affichage de la photo du jeu sélectionné dans le composant de type « PictureBox » appelé « Photo » de l'interface principale (en haut à droite), et l'affichage de ses informations détaillées dans la zone d'édition.
    a. Donner le code C# du gestionnaire d'événement qui réalise ces affichages.
    private void Grille_CellClick(object sender, DataGridViewCellEventArgs e)
    b. Expliquer brièvement le paramètre « e » de type DataGridViewCelleEventArg de ce gestionnaire et donner le nom de la classe mère de cette classe. Citer 2 autres classes dérivées de cette classe mère en indiquant les spécificités de ces classes dérivées.

Exercice 2 : Application APS.NET MVC (7 points)

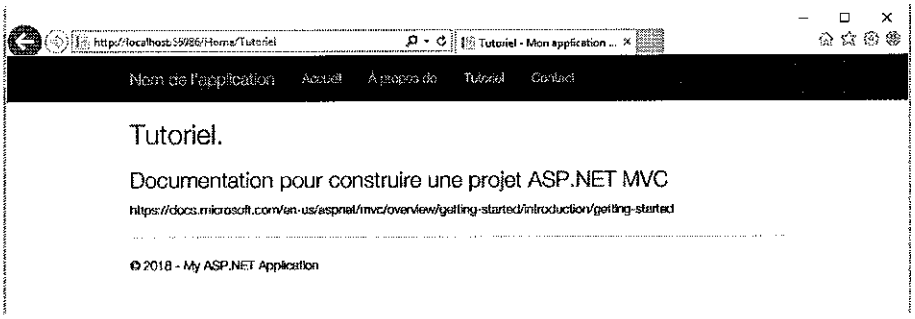
Un stagiaire est chargé de développer une application Web ASP.net selon le modèle MVC. L'environnement Visual Studio 2015, permet la génération d'un squelette d'application qui comporte une barre de menu avec 3 options : « Accueil » « A propos de » et « Contacts ». Dans une première étape, il étudie le squelette généré de l'application.

Les annexes 4, 5 et 6 donnent le code des fichiers « Layout.cshmtl », « HomeController.cs » et « About.cshtml » générés dans ce squelette.

1. **(3,5 points)** Expliquer le modèle d'architecture « Modèle – Vue – Controller » en détaillant le fonctionnement de la gestion du clic sur l'option « A propos de » 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.

Puis il décide de mettre en place une nouvelle option dans la barre de menu, nommée « Tutoriel » qui ouvre une fenêtre et affiche les informations ci-dessous, où « Tutoriel » est le titre de la page, « Documentation ... » est un message passé par le biais de la variable « ViewBag » du contrôleur à la vue et l'adresse est le contenu de la page. Le pied de page n'a pas été modifié. L'action correspondant au clic sur le bouton tutoriel est géré dans le contrôleur « Home ».

2. **(3,5 points)** Expliquer clairement les étapes à suivre pour mettre en place cette option et donner le code permettant sa mise en œuvre dans la barre de menu. Vous pouvez à votre convenance compléter sur les annexes et/ou écrire sur votre copie.



ANNEXE 1 – Base de Données des jeux

```
CREATE TABLE Jeu (
code INTEGER PRIMARY KEY,
nom VARCHAR(20) NOT NULL,
editeur VARCHAR(50),
categorie VARCHAR(50),
nomPhoto VARCHAR(50)
);

INSERT INTO Jeu Values (200, 'Okanagan','Matagot', 'Strategie', 'Okanagan');
INSERT INTO Jeu Values (100, 'Codenames','Iello', 'Reflexion', 'Codenames');
```

ANNEXE 2 – Classe « Jeu »

```
  public enum Categorie { Ambiance, Strategie, Reflexion, Autre }
    public class Jeu
    {
        private int code;
        private string nom;
        private string editeur;
        private Categorie cat;
        private Image photo;

       public Jeu(int c, string n, string e, Categorie ca, Image p)
        {
            nom = n;
            editeur = e;
            cat = ca;
            photo = p;
            code = c;
        }
        public int Code { get { return code; } }
        public string Nom { get { return nom; } set { nom = value; } }
        public string Editeur { get { return editeur; } }
        public Categorie CategorieJeu { get { return cat; } }
        public Image Photo { get { return photo; } set { photo = value; } }

        public override string ToString()
        {
            string s = "";
            s += "Code " + code + "\n";
            s += "Nom " + nom + "\n";
            s += "Editeur " + editeur + "\n";
            s += "Catégorie " + Enum.Format(typeof(Categorie), cat, "g") + "\n";
            return s;
        }
}
```

ANNEXE 3 – Classe « Form1»

```
public partial class Form1 : Form
    {
        private List<Jeu> liste;
        private OracleConnection con;
        private DbProviderFactory dbpf;
        private DbDataAdapter oDA;
        private DataSet oDS;
        private DataTable dt;

        public Form1()
        {
            InitializeComponent();
            Photo.SizeMode = PictureBoxSizeMode.StretchImage;
            liste = new List<Jeu>();
            InitConnect();
            InitListeBD();
            InitGrille();
        }

        public Jeu GetJeu(int c)
        {
            foreach (Jeu j in liste)
                if (j.Code == c) return j;
            return null;
        }

        private void InitConnect()
        {
            try
            {
                con = new OracleConnection();
                con.ConnectionString = "User Id=util100;Password=util100;
                       Data Source=//ufrsciencestech.u-bourgogne.fr:25559/ense2017";
                con.Open();
                dbpf = DbProviderFactories.GetFactory("Oracle.DataAccess.Client");
            }
            catch (Exception ec) {Edition.Text += ec.Message; }
        }

         public void InitListeBD(){
              // A compléter Question 3  }

        public void InitGrille()
        {     // A compléter Question 4  }

        private void Grille_CellClick(object sender, DataGridViewCellEventArgs e)
        {     // A compléter Question 5  }

    }
```

ANNEXE 4 – 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 5 – 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 6 – 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>
```