Direction Recherche et Ingénierie de la Formation

Examen de passage à la 2ème année

Session Juillet 2017 (Correction)

***Filière** : Techniques de Développement Informatique* — ***Épreuve** : Synthèse*
***Niveau**: TS* — ***Variante** : V2*
***Durée** : 5 heures* — ***Barème** : / 120pts*

❖ Partie I : Théorie (40 pts)

➢ Dossier 1: L'essentiel en technologies de l'information (14 pts)

▪ Exercice 1: Conversion numérique

NB: la calculatrice est strictement interdite.

Remplir le tableau suivant:

Décimal	Binaire	Octal	Hexadécimal
804	**1100100100**	**1444**	**324**
496	**111110000**	**760**	**1F0**
279	**100010111**	**427**	**117**
2561	**101000000001**	**5001**	**A01**

▪ Exercice 2: Algèbre de Boole

Soit la fonction logique suivante:

$$F(g,h,k) = ghk + \bar{g}hk + \bar{g}\bar{h}k + gh\bar{k}$$

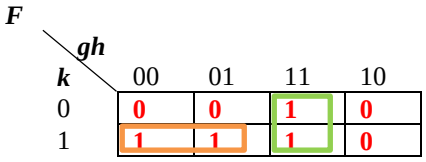
1- Simplifier analytiquement la fonction logique F.

$$F(g,h,k) = gh + \bar{g}k$$

2- Construire la table de vérité

g	h	k	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

3- Simplifier avec la méthode de Karnaugh la fonction logique ***F***.



F	gh	00	01	11	10
k					
0		0	0	1	0
1		1	1	1	0

$$F(g,h,k) = gh + \bar{g}k$$

➢ Dossier 2: Programmation structurée

Réinsertion du minimum d'un tableau

Il s'agit de réinsérer le minimum d'un tableau déjà rempli par 10 réels, dans une position saisie par l'utilisateur.

Exemple:

```
Tableau: | 6 | -2 | 9 | 0 | 2 | 3 | 1.1 | 1 | 2 | 3 |
Position: 5
Nouveau tableau: | 6 | -2 | 9 | 0 | -2 | 2 | 3 | 1.1 | 1 | 2 | 3 |
```

1- Écrire une fonction qui retourne le minimum d'un tableau de taille quelconque passé en paramètre.
2- Écrire une procédure qui insère un élément dans un tableau de taille quelconque passé en paramètre, en passant l'élément à insérer et sa position par paramètres.
3- Utiliser la fonction et la procédure pour réinsérer le minimum d'un tableau déjà rempli par 10 réels, dans une position saisie par l'utilisateur.

```
Tableau b : Réel [ 11 ] <- [ 6 , -2 , 9 , 0 , 2 , 3 , 1.1 , 1 , 2 , 3 ]
Variable c : Entier
Variable e : Réel
Début
     Saisir c
     e <- Mini ( b , 11 )
     InsérerA ( b , 11 , c , e )
Fin
Fonction Mini ( f : Réel [ 11 ] ; g : Entier ) : Réel
Variable h : Réel
Variable i : Entier
Début
   h <- f [ 1 ]
   Pour i <- 2 à g
        Si f [ i ] < h Alors
           h <- f [ i ]
        FinSi
   FinPour
   Retourner h
Fin
Procédure InsérerA ( R f : Réel [ 11 ] ; g , h : Entier ; i : Réel )
Variable j : Entier
Début
   Si h <= 11 Et h >=1 Alors
      Pour j <- g à h + 1 Pas -1
           f [ j ] <- f [ j - 1 ]
```

```
        FinPour
        f [ h ] <- i
    FinSi
Fin
```

➢ Dossier 3: Analyse et conception orientée objet (13 pts)

Gestion de la sélection nationale de football

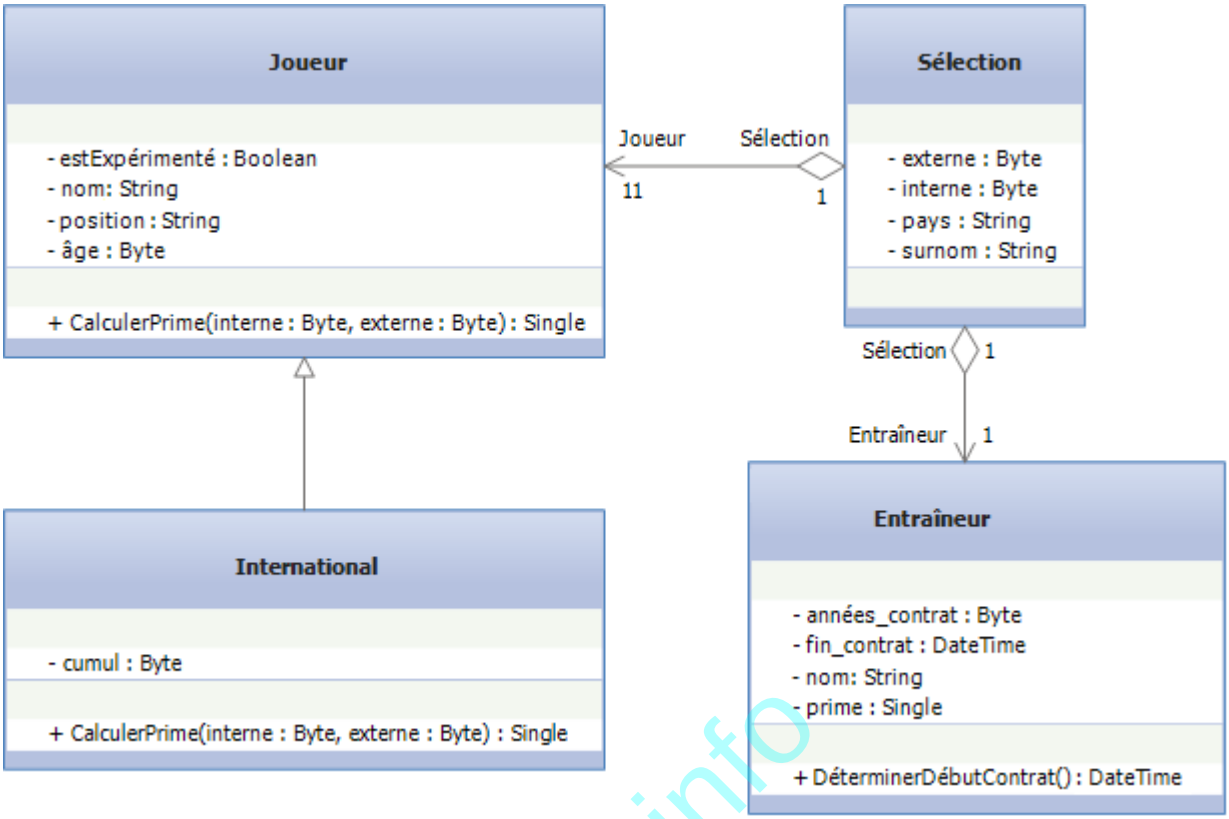
La fédération nationale de football souhaite gérer la sélection nationale. La sélection nationale a le nom de son pays qu'elle représente, son surnom, le nombre de matchs joués à l'intérieur du pays, et le nombre de matchs joués à l'étranger pendant l'année en cours.

La sélection nationale est formée autour de onze joueurs (excluant ceux en réserve). Chaque joueur a son nom complet, son âge, sa position sur le terrain, et une indication sur le fait qu'il est un joueur qui a une expérience avec la sélection nationale ou non. Chaque joueur touche une prime annuelle calculée automatiquement à partir du nombre de matchs joués par la sélection à l'intérieur du pays, du nombre de matchs joués à l'étranger pendant l'année en cours, et du fait qu'il est expérimenté ou non.

Un joueur international est un joueur qui joue dans un club de football à l'étranger. Pour ce joueur, il faut tenir en compte du cumul d'années qu'il a passé en jouant à l'étranger. Le calcul de la prime annuelle du joueur international inclut le cumul d'années à l'étranger.

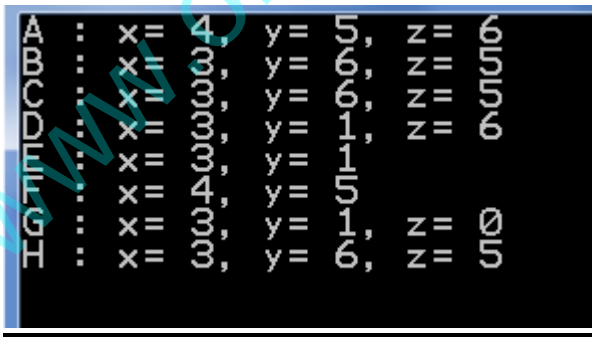
La sélection est entraînée par un entraîneur. L'entraîneur a son nom complet, la date de fin de son contrat avec la sélection, le nombre d'années du contrat, et sa prime annuelle. La date de début du contrat peut être déterminée automatiquement.

Établir le diagramme de classes permettant de gérer les données de la sélection nationale. Toutes les variables d'instance doivent être privées, et toutes les méthodes doivent être publiques.



❖ Partie II: Pratique (80 pts)

➢ Dossier 1: Programmation structurée (15 pts)



➢ Dossier 2: Programmation événementielle et orientée objet (65 pts)

Développement d'une application pour la gestion d'une équipe de football

La fédération nationale de football souhaite gérer la sélection nationale. L'analyse de l'application mène au développement des classes suivantes (toutes les classes doivent être **publiques**, toutes les variables d'instance doivent être **privées**, et toutes les méthodes doivent être **publiques**):

1- Classe "Joueur":

a- Écrire la classe "**Joueur**" caractérisée par son nom complet, son âge, sa position sur le terrain, et s'il est expérimenté (vrai) ou non (faux).

b- Ajouter tous les accesseurs

c- Ajouter à cette classe deux constructeurs: un constructeur par défaut, et un constructeur d'initialisation avec tous les paramètres.

d- Écrire une méthode polymorphe "**CalculerPrime(interne,externe)**" qui admet en paramètre les nombres des matchs joués par la sélection nationale pendant l'année en cours, et qui retourne la prime annuelle du joueur suivant la relation:

$$Prime = \begin{cases} 20000\ MAD\ par\ match\ \text{à}\ l'intérieur; \\ 30000\ MAD\ par\ match\ \text{à}\ l'extérieur; \\ +50\%\ de\ la\ prime\ du\ joueur\ principal, si\ le\ joueur\ est\ expérimenté. \end{cases}$$

```
using System;
namespace Foot
{
    [Serializable]
    public class Joueur
    {
        private string nom,position;
        private byte âge;
        private bool estExpérimenté;
        public string Nom
        {
            get
            {
                return nom;
            }
            set
            {
                nom = value;
            }
        }
        public byte Âge
        {
            get
            {
                return âge;
            }
            set
            {
                âge = value;
            }
        }
        public string Position
        {
            get
            {
                return position;
            }
            set
            {
                position = value;
            }
        }
        public bool EstExpérimenté
        {
            get
            {
                return estExpérimenté;
            }
            set
            {
                estExpérimenté = value;
            }
```

```
        }
        public Joueur(){}
        public Joueur(string nom, string position, byte âge, bool estExpérimenté)
        {
            this.nom = nom;
            this.position = position;
            this.âge = âge;
            this.estExpérimenté = estExpérimenté;
        }
        public virtual float CalculerPrime(byte interne, byte externe)
        {
            float prime = 20000 * interne + 30000 * externe;
            return prime = estExpérimenté ? 1.5F * prime : prime;
        }
    }
}
```

2- Classe "International":

a- Écrire la classe "**International**" qui hérite de la classe "**Joueur**". Le joueur international est caractérisé par le cumul d'années qu'il a passé en jouant à l'étranger.

b- Ajouter à cette classe un constructeur d'initialisation avec tous les paramètres. Le cumul d'années doit être inférieur à l'âge du joueur; sinon lever une exception

c- Réécrire la méthode polymorphe "**CalculerPrime(interne,externe)**" sachant que:

$$Prime_{International} = Prime_{Joueur} + 5000\ MAD\ x\ cumul\ d'années\ à\ l'étranger.$$

```
using System;
namespace Foot
{
    public class International : Joueur
    {
        private byte cumul;
        public International(string nom, string position, byte âge, bool estExpérimenté, byte cumul)
            : base(nom, position, âge, estExpérimenté)
        {
            if (âge < cumul) throw new Exception("Erreur cumul!");
            this.cumul = cumul;
        }
        public override float CalculerPrime(byte interne, byte externe)
        {
            return CalculerPrime(interne, externe) + 5000 * cumul;
        }
    }
}
```

3- Classe "Entraîneur":

a- Écrire la classe "**Entraîneur**" caractérisée par son nom complet, sa date de fin de son contrat avec l'équipe, le nombre d'années du contrat, et sa prime annuelle.

b- Ajouter à cette classe un constructeur d'initialisation avec tous les paramètres. La prime annuelle doit être entre 300000 MAD et 600000 MAD; sinon lever une exception.

c- Écrire une méthode "**DéterminerDébutContrat()**" qui retourne la date de début du contrat.

```
using System;
namespace Foot
{
    [Serializable]
    public class Entraîneur
```

```
    {
        private string nom;
        private DateTime fin_contrat;
        private byte années_contrat;
        private float prime;
        public Entraîneur(string nom, DateTime fin_contrat, byte années_contrat, float prime)
        {
            this.nom = nom;
            this.fin_contrat = fin_contrat;
            this.années_contrat = années_contrat;
            if (prime < 300000 || prime > 600000) throw new Exception("Erreur prime!");
            this.prime = prime;
        }
        public DateTime DéterminerDébutContrat()
        {
            return fin_contrat.AddYears(-années_contrat);
        }
    }
}
```

4- Classe "Sélection":

a- Écrire la classe "**Sélection**" caractérisée le nom de son pays qu'elle représente, son surnom, le nombre de matchs joués à l'intérieur du pays, et le nombre de matchs joués à l'étranger pendant l'année en cours, la liste de ses joueurs, et son entraîneur.

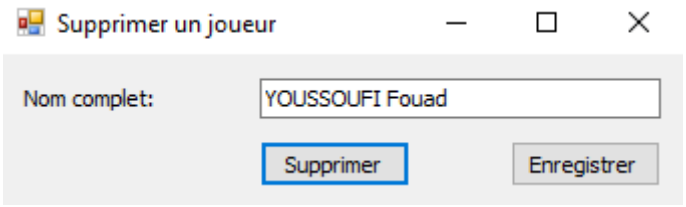
b- Ajouter un accesseur à lecture seule pour la liste des joueurs

c- Ajouter à cette classe deux constructeurs: un constructeur sans paramètres, et un constructeur d'initialisation avec tous les paramètres sauf la liste des joueurs. Les deux constructeurs doivent instancier la liste des joueurs avec une capacité de 11.

```
using System;
using System.Collections.Generic;
namespace Foot
{
    [Serializable]
    public class Sélection
    {
        private string pays, surnom;
        private byte interne, externe;
        private List<Joueur> joueurs;
        private Entraîneur entraîneur;
        public List<Joueur> Joueurs
        {
            get
            {
                return joueurs;
            }
        }
        public Sélection()
        {
            joueurs = new List<Joueur>(11);
        }
        public Sélection(string pays, string surnom, byte joués, byte gagnés, Entraîneur entraîneur)
            : this()
        {
            this.pays = pays;
            this.surnom = surnom;
            this.interne = joués;
            this.externe = gagnés;
            this.entraîneur = entraîneur;
        }    }}
```

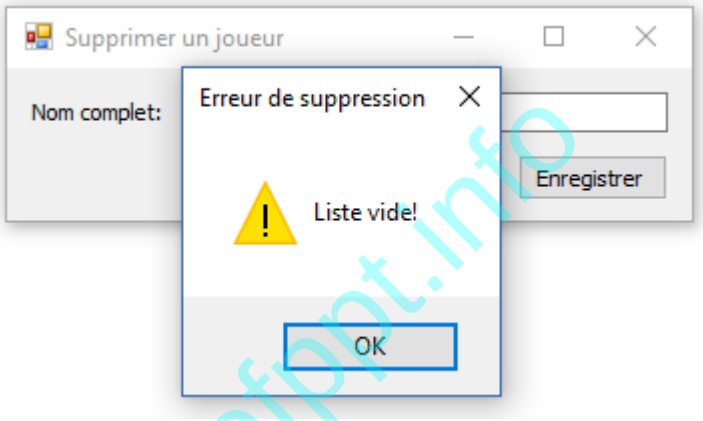
5- Formulaire de suppression des joueurs:

NB: donner uniquement le code à mettre à l'intérieur des méthodes événementielles. L'entête de ces méthodes événementielles n'est pas demandé.

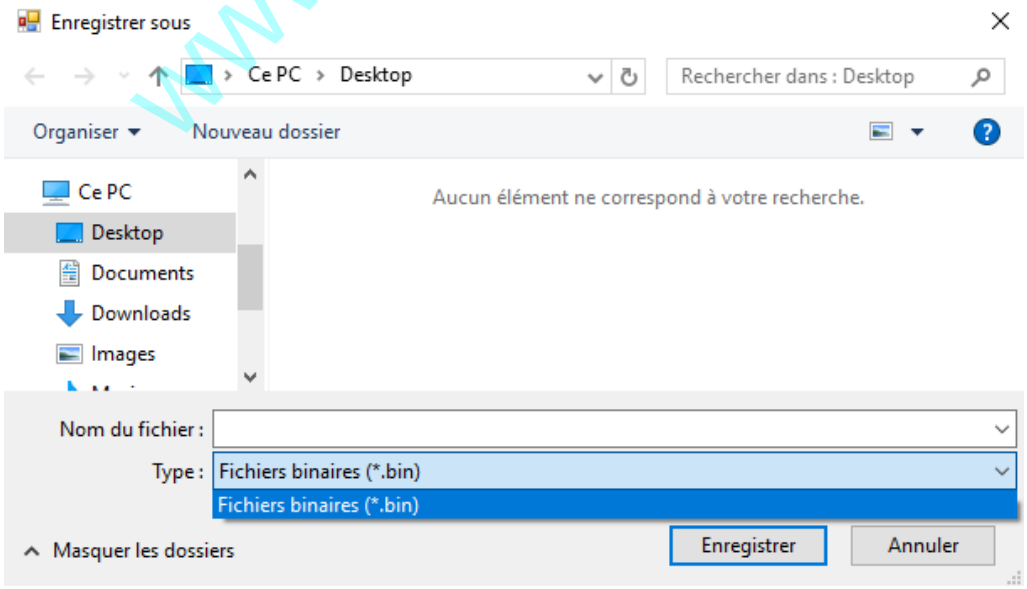


Soit la sélection "sélection1" une instance de la classe "Sélection", contenant une liste remplie de joueurs. Le formulaire précédent permet de supprimer un joueur de la sélection.

a- Écrire le code du bouton "Supprimer" permettant de supprimer un joueur de la liste des joueurs de "sélection1". Lever une exception si la liste des joueurs est déjà vide. Capturer l'exception dans une boite de message d'avertissement.



b- Écrire le code du bouton "Enregistrer" permettant d'enregistrer la sélection dans un fichier binaire, en utilisant une boîte de dialogue de sauvegarde de fichiers.



```
using System;
using System.Windows.Forms;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
namespace Foot
{
    public partial class Form1 : Form
    {
        private Sélection sélection1;
        public Form1()
        {
            InitializeComponent();
            sélection1 = new Sélection();
            sélection1.Joueurs.Add(new Joueur("ALAOUI Ahmed", "Guardien de but", 25, false));
            sélection1.Joueurs.Add(new Joueur("ALAMI Mohamed", "Défenseur", 30, true));
            sélection1.Joueurs.Add(new Joueur("OUAZZANI Ali", "Aile droit", 20, false));
            sélection1.Joueurs.Add(new Joueur("SAFI Ayoub", "Aile gauche", 22, true));
            sélection1.Joueurs.Add(new Joueur("YOUSSOUFI Fouad", "Attaquant", 27, true));
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                if (sélection1.Joueurs.Count == 0) throw new Exception("Liste vide!");
                for (int a = 0; a < sélection1.Joueurs.Count; a++)
                    if (sélection1.Joueurs[a].Nom_complet == textBox1.Text)
                    {
                        sélection1.Joueurs.RemoveAt(a);break;
                    }
            }
            catch (Exception a)
            {
             MessageBox.Show(a.Message, "Erreur de suppression", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
        private void button2_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();
            saveFileDialog1.Filter = "Fichiers binaires (*.bin)|*.bin";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                Stream flux = File.Create(saveFileDialog1.FileName);
                BinaryFormatter format = new BinaryFormatter();
                format.Serialize(flux, sélection1);
                flux.Close();
            }
        }
    }
}
```