30/47

15,0

BEKKALI Amine
M1 - 2014

Examen C#

Christian Khoury
Durée 2h
Documents Interdits

- **<u>Répondre succinctement dans les espaces entre les questions et non ailleurs.</u>**

Exercice 1 [15 pts]

1. Donner un exemple d'héritage de classe et comment un constructeur de classe dérivée utilise le constructeur de base (3 pts)

3

```
class A: B
{
    public A(): base()
    {
        // le code
    }
}
```

2. Donner un exemple d'utilisation de lien dynamique de méthode en C#. Expliquer en quoi cela pourrait servir à travers un exemple. (5 pts)

5

La réimplémentation de méthode d'une classe fille (avec le mot clef override) est un exemple d'utilisation de lien dynamique

Cela est utile si par exemple on veut définir un comportement spécifique à une classe, mais dont une partie est néanmoins partagée par la classe mère

Ex: dans l'exo 2: les instruments sont tous joué, mais certains soufflent et d'autrent pincent.

3. Est-ce le comportement de base en C# ? Comparer cela à Java et C++. (3 pts)

4. Quel est l'inconvénient majeur des liens dynamiques ? (2 pts)

0

5. Intérêt des références vs pointeurs et vice versa ? (2 pts)

0

une référence permet de manipuler des objets et d'en change leur contenu, on manipule un objet référencé par son adresse memoire

un pointeur va stocker l'adresse, il faut donc manipuler son contenue

?

Exercice 2 [14 pts]

Tout instrument de musique peut être joué. Pour les instruments à vent, le musicien souffle alors que les instruments à cordes son pincés. Le saxophone et la clarinette sont des instruments à vent mais « soufflés » différemment.

1. Ecrire les classes correspondant à ce système simplifié d'instruments à musique. [8 pts]

```
abstract class Instrument
{
    public Instrument();
    public abstract void jouer();
}
class InstruVent : Instrument
{
  public InstruVent() : base()
  {
  }
  public override void jouer()
  {
    Console.WriteLine("Je souffle");
  }
class InstruCorde : Instrument
{
  public InstruCorde() : base()
  {
  }
  public override void jouer()
  {
    Console.WriteLine("Je pince");
```

6

Souffle

```
class Saxophone : InstruVent
{
    public Saxophone() : base()
    {
    }
    public new void jouer()
    {
        Console.WriteLine("Je souffle dans le saxophone");
    }
}
class Clarinette : InstruVent
{
    public Clarinette(); base()
    {
    }
   public new void jouer()
   {
      Console.WriteLine("Je souffle dans la clarinette");
   }
```

2. Créer une classe Magasin qui contiendrait une collection d'instruments à corde et à vent (ceci peut s'élargir dans le futur !). Garder dans cette classe le nombre total d'instruments stockés.
    a. Ecrire la « propriété » qui permet d'accéder seulement en lecture au nombre d'instruments dans le magasin. [2 pts]
    b. Ecrire l'indexeur qui permet d'accéder à un instrument de la collection [2 pts]
    c. Ecrire la méthode qui permettrait de jouer de la musique avec tous les instruments de la collection [2 pts]

```
using System.Collection.
             Generic;
class Magasin<T> where T: Instrument, new()
{
  private int instruStock;
  private List<T> instruList;
  public int NombreInstru
  {
     get { return instruStock; }
  }
  public T this[int i]
  {
     get { return instruList.ElementAt(i);}
  }
  public void jouerTousInstru()
  {
     foreach(T instru in instruList)
     {
        instru.jouer();
     }
  }
}
```

1

2

2

Note : La solution proposée n'est pas totalement correcte dans la mesure ou la valeur B ou A n'est pas respective à chaque Thread . . .

Exercice 3 [10 pts]

Ecrire un programme qui lance 2 threads qui exécutent la même méthode qui affiche une lettre à l'écran, mais chaque thread doit afficher une lettre différente (A et B respectivement). De plus, il faudrait synchroniser les exécutions de façon à ce que B soit affichée avant A.

```
class ToBeRun
{
    static bool s1 = true;

    public void afficherLettre()
    {   if (s1)
        {
            Console.WriteLine("B");
        } s1 = false
        else
        {
            Console.WriteLine("A");
            s1 = true;
        }
    }
}
```

4

```
class MainClass
{
    public static void Main(String[] args)
    { Thread[] myTh = new Thread[2]
      for (int i=0; i< myTh.Leght; i++)
      [   myTh[i]=new Thread(new ThreadStart(new ToBeRun().affi
          myTh[i].Start();
    }   }
```

Exercice 4 [8 pts]

Annoter le code suivant (les lignes significatives) en expliquant ce qu'il est censé faire.

```
public delegate void Deleg1(int x);
```

← création d'un delegate: on lit la signature de la méthode

```
class A
{
        public void method1(int a) { Console.Writeline(a+1); }
        public void method2(int a) { Console.Writeline(a+1); }
        public void method3(int a) { Console.Writeline(a+3);}
}
```

} on créé des méthodes ayant la même signature que celle de Deleg1

```
class test
{
        static void Main(string[] args)
        {
                meths m = new meths();
                Deleg1 del = new Deleg1( A.method1() ); // Découvrez l'erreur
                del += new Deleg1(A.method3() );
                del += new Deleg1(A.method2() );

                del(12);
        }
}
```

← on instancie le delegate pour lui attribuer les méthodes ayant la même signature

// les méthodes ne sont pas static. Il faut instancier un objet de type A pour appeler les méthodes.

↑ on appel les méthodes par le biais du delegate

7

cher Lettre));