Pieronne
Kevin
M1 - IL

29/47

E
C

PIERONNE Kévin
M1 - 2014

14,5

Durée 2h
Documents Interdits

- **Répondre succinctement dans les espaces entre les questions et non ailleurs.**

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)

Soit la classe B existante

```
public class D : B        <- D hérite de B
{
    public D() : base()   <- base() appelle le constructeur de la classe mère.
    {
    }
}
```

3

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)

L'utilisation des mot-clés virtual / override font appel aux lien dynamique

Exemple : cas du polymorphisme :

```
public class B
{
    ....
    public virtual void method();
}

public class D : B
{
    ....
    public override void method(){ }
}

B obj = new D();
obj.method();
```

5 Bravo !

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

+2 C#

Java possède le même fonctionnement avec l'utilisation nécessaire des mots clés virtual / override NON

En C++, seulement virtual est nécessaire

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

Pour annuler les liens dynamiques ponctuellement, il faut caster l'objet.

4

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

Reference → accès direct à l'objet

Pointeur → accès par le contenu stocké à l'adresse

4

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]

```
public class instrument
{
  public instrument()
  {
  }
  public virtual void jouer();
}
```

```
public class vent : instrument
{
  public vent() : base()
  {
  }
  public override void jouer()
  {
    Console.WriteLine("ça souffle");
  }
}
```

```
public class corde : instrument
{
  public corde() : base()
  {
  }
  public override void jouer()
  {
    Console.WriteLine("ça pince");
  }
}
```

```
public class saxophone : vent
{
  public saxophone() : base()
  {
  }
  public override void jouer()
  {
    Console.WriteLine("ça souffle dans le saxo");
  }
}
```

```
public class clarinette : vent
{
  public clarinette() : base()
  {
  }
  public override void jouer()
  {
    Console.WriteLine("ça souffle la clarinette");
  }
}
```

6

Pieronne
Kevin
M1 - IL

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]

```
public class Magasin
{
    public int Nstock { get; } <- Uniquement le get.

    public List<instrument> stock { get; set; } <- get et set

    public Magasin()
    {
        Nstock = 0;
        stock = new List<instrument>();
    }

    public void jouerAll()
    {
        foreach (instrument inst in stock)
        {
            inst.jouer();
        }
    }

    /*
    Q.b : Un indexeur est la surcharge de l'opérateur []
    pour une classe
    */

}
```

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.

```
public class monThread : Thread
{
  public string lettre {get; set;}

  public monThread (string lettreA)
  {
    lettre = lettreA;
  }

  public override void Run()
  {
    lock (jeton)
    {
      afficherLettre();
    }
  }

  public void afficherLettre()
  {
    Console.WriteLine("Lettre du thread : " + lettre);
  }
}
```

private static object jeton = new object();

Dans le main →

```
monThread[2] tab = new monThread[2];
tab[0] = new monThread("B");
tab[1] = new monThread("A");
for (int i = 0; i < 2; i++)
{
  tab[i].start(tab[i].Run());
}
```

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);

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);}
}

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);
        }
}
```

// definition de la signature des fonctions qui sera acceptée.

Création de fonction. La signature est la même que celle requise pour le delegate

Il faut += pour rajouter la fonction

Ajout de method3 et method2 dans le delegate

Appel de la 1ère fonction ajouté au delegate

⇒ En admettant que l'erreur précédente est corrigée ⇒ appellera A.method1(-12) car method1 est la première à avoir été ajoutée.

et les autres ?