

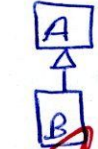
Ex DE LA ROCHE SAINT ANDRE Matthieu
M1 - 2014
Chr
Durée 2h
Documents Interdits

37/47

- **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)



```
public class A {
    private int -x;
    A(int arg) {
        -x = arg;
    }
}
```

```
public class B : A {
    B() : base(10) {
        // do stuff
    }
}
```

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)

```
A a = new A();
A ab = new B();
```

```
a.doStuff(); // calls A.doStuff
```

```
ab.doStuff(); // calls B.doStuff => fonctionne uniquement avec un lien dynamique
// sinon, c'est A.doStuff qui est appelée
```

C'est très utile pour gérer une liste hétérogène contenant des objets ^{instances de classes} dérivants d'une même ^{classe mère} interface.

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

Ce n'est pas le comportement de base. En l'absence de virtual et d'override,
ab.doStuff() appelle A.doStuff(). C'est donc similaire au C++ (pas besoin d'override). En Java, on utilise l'annotation @Override pour clarifier.

*et le compilateur lance un warning.

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

Ils sont résolus à l'exécution, ce qui ralentit le programme.

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

Intérêt

Références

Pointeurs

+ moins risqués
+ un attribut d'une classe ne peut être une référence
+ plus simple à utiliser (syntaxe, ...)

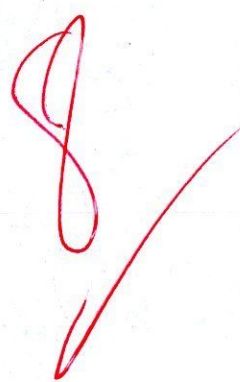
+ plus de libertés pour le programmeur
- doit spécifier les zones à risque dans le code
- dangereux. Sujet aux erreurs du programmeur

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 abstract class Instrument {  
    public virtual void jouer();  
}  
  
public abstract class InstrVent : Instrument {  
    public override void jouer() {  
        souffler();  
    }  
    public virtual void souffler();  
}  
  
public class Saxo : InstrVent {  
    public override void souffler() {  
        Console.WriteLine("saxo");  
    }  
}  
  
public class Clarinette : InstrVent {  
    public override void souffler() {  
        Console.WriteLine("Clarinette");  
    }  
}  
  
public abstract class InstrKorde : Instrument {  
    public override void jouer() {  
        pincer();  
    }  
    public virtual void pincer();  
}
```



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.

- Ecrire la « propriété » qui permet d'accéder seulement en lecture au nombre d'instruments dans le magasin. [2 pts]
- Ecrire l'indexeur qui permet d'accéder à un instrument de la collection [2 pts]
- Ecrire la méthode qui permettrait de jouer de la musique avec tous les instruments de la collection [2 pts]

```
public class Magasin {  
    private ArrayList ArrayList <Instrument> l = new ArrayList ();  
    public int Count {  
        1 get {  
            return l.Count;  
        }  
    }  
}
```

```
    public Instrument getInstrument (int index) {  
        2 return l.get(index);  
    }
```

```
    public void playAll () {  
        foreach (Instrument i in l) {  
            i.jouer();  
        }  
    }
```

```
}
```

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.

```
void async(string X) {  
    Console.WriteLine(X);
```

```
public class A {  
    private static object flag;  
    public static async (string X) {  
        lock sync (flag) { // pas sûr du mot clé. Mais celui-ci permet  
            Console.WriteLine(X); // de bloquer un bout  
        } // de code à l'aide  
        // d'un objet commun  
    }  
}
```

```
public static void Main (string[] args) {  
    Thread a = new StartThread (async);  
    Thread b = new StartThread (async);
```

```
a.Start("A");
```

```
b.Start("B");  
a.Start("A");
```

```
}
```

```
}
```


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 // contient juste une liste de méthodes dont la signature correspond
{ // au delegate Deleg1
    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(); // meths indéfini. m inutilisé.
```

```
        Deleg1 del = new Deleg1(A.method1); // Découvrez l'erreur // "abonne" A.method1
        del += new Deleg1(A.method3); // ajoute A.method3 en l'ajoutant à la
        del += new Deleg1(A.method2); // ajoute A.method2 liste de "callbacks"
                                                    del
```

```
        del(12); // affiche 13 dans un ordre imprévisible
```

```
        13
        15
```

```
        // car ça appelle toutes les méthodes ajoutées
        // au delegate del
```

