TP2 : TicTacToe

13 novembre 2007

1 Reprise

1.1 Fonctions Virtuelles

```
using System;
using System.Text;

    public abstract class A
    {
        public virtual void test(){}
    }

    public class B2 : A
    {
        public override void test()
        {
            Console.WriteLine("B2");
        }
    }
    public class B : A
    {
        public override void test()
        {
            Console.WriteLine("B");
        }
    }

    public class C : B
    {
        public override void test()
        {
            Console.WriteLine("C");
        }
    }
```

```
public class test
{
    public static void Main()
    {
        C test = new C();
        test.test();

        B test2 = (B)test;
        test2.test();
    }
}
```

Le programme va afficher

```
C
C
```

Ceci car le mot clé "override" indique que la fonction virtuelle recoit une nouvelle interprétation. Virtuelle implique que la fonction de la classe sous-jacente est appelée. Regardez

```
B test2 = (B)test;
test2.test();
```

test2 est une interpretation de la variable test du type C en type B. Comme la fonction est virtuelle, l'appel test2.test() appelle la fonction d'un type C ! ! ! (En Java toutes les fonctions se comporte ainsi)

Imaginez une liste d'objets géométriques. Chaque objet à une fonction *aire*. Mais selon le vrai type de l'object (cercle, carré ...) une fonction adapté est appelé sans nous obliger de faire un cast particulier.

En CSharp il y a aussi la possibilité de casser cette chaîne de fonctions virtuelles avec le mot clé "new" qui remplace "overridden". Maintenant la fonction qui est exécuté est celle du type courant. Le changement d'"overridden" en "new" donne l'affichage suivante :

```
C
B
```

1.2 Références

```
swap(ref int t1, ref int t2) {
    int temp;
    temp=t1;
    t1=t2;
    t2=temp;
}
```

Par défaut toutes les valeurs dans une fonction sont passées par valeur. Sans le mot "ref" on manipule donc seulement des copies et la fonction ne marche pas.

```
using System;
using System.Text;

public class A {
    public int a;
}

public class Prog {
    public static void setA(A v, int t)
    {
        v.a = t;
    }

    public static void Main()
     {
        A temp=new A();
        temp.a=3;

        setA(temp, 0);

        Console.WriteLine("{0}", temp.a);
     }
}
```

Pour un object, passer par valeur veut dire : une copie de la référence sur un objet. Du coup, le swap ne marcherait toujours pas sans "ref", mais changer les variables membres d'une copie d'une référence change l'objet qui est "référencié".

En Java, une implementation d'une fonction Swap comme celle ci est IMPOSSIBLE. En CSharp ca marche, du au mot clé "ref".