EXAMEN " Introduction à la programmation orientée objet''

Exercice 01 (10pts)

1. Ecrire la classe Calcul qui contient la fonction statique x^n (n entier naturel positif) sous une forme **récursive** :

float puissance(float x; int n);

```
class Calcul
    {
        public static float puissance(float x, int n) (1)
        {
            if (n == 0) return 1;
            else return x*puissance(x,n-1);
        }

    }
```

2. Modélisation de polynôme sous forme d'un tableau

On cherche à modéliser les polynômes de degrés inférieurs à un maximum (100) sous la forme d'un tableau contenant ses coefficients

$p(x) = a_0 + a_1x + a_2x^2 + ... + a_{n-1}x^{n-1} + a_nx^n$ sera représenté dans le tableau

a_0	a_1	a_2	...	a_{n-1}	a_n

- Ecrire une classe **TPolynome** représentant un polynôme sous cette forme, en encapsulant l'accès au coefficients grâce à 2 méthodes **setcoefficient(float coefl ; int degre)** et float **getcoefficient(int degre)**

- Ecrire son **constructeur** qui initialisera tout polynôme à p(x)=0

-Ecrire la méthode float **CalculP(float x)** qui retourne la valeur du polynôme pour un paramètre x donné, en utilisant la formule traditionnelle

$p(x) = a_0 + a_1x + a_2x^2 + ... + a_{n-1}x^{n-1} + a_nx^n$ en utilisant la fonction puissance définie à la question1.

```
class TPolynome
    {
        float[] tab;(0,25)
        public void setcoefficient(float coefl, int degre)  (0,5)
        {
            tab[degre] = coefl;
        }
        float getcoefficient(int degre)   (0,5)
        {
```

```
            return tab[degre];
        }
        public TPolynome(int n) (0,5)
        {
            tab = new float[100];
            for (int i = 0; i < n; i++) tab[i] = 0;
        }
        public float CalculP(float x) (1)
        {
            float poly = 0;
            for (int i = 0; i < tab.Length; i++)
                poly += tab[i] * Calcul.puissance(x, i);
            return poly;
        }
    }
```

3. Monôme

- Ecrire une classe **TMonome** permettant de mémoriser un monôme $a.x^n$ défini par le couple d'attributs (degré, coefficient).

- Ecrire le constructeur **TMonome(int degre;float coef**) permettant d'initialiser un objet de cette classe.

- Ecrire la méthode float **Calcul(float x)** calculant la valeur du monôme pour le paramètre x en utilisant la fonction puissance définie à la question 1.

```
class TMonome
    {
        public int degre; float coef; (0,25)
        TMonome(int degre,float coef) (0,5)
        {
            this.degre = degre;
            this.coef = coef;
        }
        float calcul(float x) (1)
        {
            return coef * Calcul.puissance(x, degre);
        }
    }
```

4. Modélisation de polynôme sous forme d'une liste de monômes

- Ecrire une classe **TPolynomeListe** qui contiendra dans un objet de la classe **ArrayList** une liste d'objets **Tmonome.**

- Ecrire le constructeur de cette classe**.**

- Ecrire la méthode float **getcoefficient(int degre)** qui cherche si un monôme de degré **degre** existe dans la liste et ramène son coefficient (sinon la fonction ramène 0).

- Ecrire la méthode **setcoefficient(float coef ; int degre)** qui ajoute un monôme si le polynôme ne contient pas de monôme de ce degré dans la liste, ou qui remplace dans la liste le coefficient du monôme du degré considéré, s'il existe dans la liste.

- Ecrire la méthode float **CalculP(float x)** qui retourne la valeur du polynôme pour un paramètre x.

```
class TPolynomeListe
    {
        ArrayList tab; (0,25)
        public TPolynomeListe()(0,25)
        {
            tab = new ArrayList();
        }
        public float getcoefficient(int degre) (1,5)
        {
            int i=0;
            while ((i < tab.Count) && (((TMonome)tab[i]).degre != degre))
            {
                i++;
            }
            if (i == tab.Count) return 0;
            else return ((TMonome)tab[i]).coef;
        }
        public void setcoefficient(float coef,  int degre) (1,5)
        {
            if (getcoefficient(degre) == 0)
            {
                tab.Add(new TMonome(degre, coef));
            }
            else
            {
                int i = 0;
                while ((i < tab.Count)&&(((TMonome)tab[i]).degre != degre))
                {
                     i++;
                }
                ((TMonome)tab[i]).coef = coef;

            }
        }
        public float CalculP(float x) (1)
        {
            float poly=0;
            foreach (TMonome y in tab)
            {
                poly += y.calcul(x);
            }
            return poly;

        }


    }
```

ArrayList
La classe ArrayList contient des éléments de type Object
La propriété Count permet de connaître le nombre d'éléments dans la collection
La méthode Add permet d'ajouter un object à la fin de la ArrayList

La méthode insert permet d'ajouter un object à une position dans la ArrayList
La méthode remove permet de supprimer supprimer un object de la ArrayList
Utiliser [] pour accéder aux éléments de la ArrayList
La méthode Clear permet de supprimer tous les éléments

Exercice 2 (3 pts)

1. On considère le code suivant : (1)

```
class program
    {
        public static void Main()
        {
            E e1 = new Ebiss();
            e1.f();
            ((E)e1).f();
            ((Ebiss)e1).f();
        }
    }
    class E
    {
        public virtual void f()
        {
            Console.WriteLine("E.f");
            g();
        }
       public virtual void g()
        {
            Console.WriteLine("E.g");
        }
    }
    class Ebiss:E
    {
       public override void f()
        {
            Console.WriteLine("Ebiss.f");
            g();
        }
        public override void g()
        {
            Console.WriteLine("Ebiss.g");
        }
    }
```

Trouver la ou les bonnes réponses
a- Ce programme ne peut pas être compilé.
b- Il génère une erreur à l'exécution.
c- La sortie sera :Ebiss.f Ebiss.g Ebiss.f Ebiss.g Ebiss.f Ebiss.g.
d- La sortie sera :Ebiss.f Ebiss.g E.f E.g Ebiss.f Ebiss.g.
e- La sortie sera :E.f E.g E.f E.g Ebiss.f Ebiss.g.

2. L'interface IClonable est définie comme suit : (0,5)

```
interface ICloneable
{
      public object Clone();
}
```

Soit la classe P telle que :

```
class P : ICloneable
    {
        public int i=10;
        P(int i){this.i=i;}
```

```
        public object Clone()
        {
            try
            { return base.Clone(); }
            catch (Exception e) { return null;}
            }
    }
```

Quelle est la bonne réponse et pourquoi ?
a- Cette implémentation est correcte.
b- Cette implémentation n'est pas correcte. La classe Object ne contient pas de méthode Clone

3. (1,5)

```
interface I {
    void f();
    int f(int i);
}
interface J : I{
    void f(double d);
}
class A : J{
    public void f(){ ;}
    public int f(int i){return 0 ;}
    public void f(double d){ ;}
    void f(char c){ ;}
}
```

Trouver la ou les bonnes réponses :
(a) ce code ne peut pas être compilé ;
(b) le code suivant est correct (pas d'erreur ni à la compilation ni à l'exécution) :
I i=new A() ; J j=(J)i ; j.f(3.2) ;
(c) le code suivant est correct (pas d'erreur ni à la compilation ni à l'exécution) :
I ii= new J() ; ii.f() ;

Exercice 03 (07 pts)

(1) Une méthode déclarée static: (Trouver la ou les bonnes réponses) (0,5)
A. est toujours une méthode de classe
B. est toujours une méthode d'instance
C. peut être redéfinie dans une classe dérivée
D. ne peut être appelée qu'à partir de méthodes de la classe où elle a été définie

(2) Une variable static déclarée dans une classe A: (Trouver la ou les bonnes réponses) (0,5)
A. ne peut être utilisée qu'à l'intérieur d'une méthode statique
B. ne peut être modifiée
C. ne peut être initialisée
D. est indépendante de l'instance des objets de A

(3) Parmi les déclarations de classe et d'interface suivantes, indiquer lesquelles ne sont pas valides (certaines déclarations seront réutilisées par les déclarations qui les suivent) : (1)
(a) _ class Foo { }
(b) _ class Bar : Foo { }
(c) _ interface Baz { }
(d) _ interface Fi { }
(e) _ interface Fee : Baz { }
(f) _ interface Zee : Foo { }
(h) _ interface Zoo : Foo, Fee { }
(i) _ interface Boo : Fi { }
(j) _ class Zoom : Fi, Baz { }

(k) _ class Toon : Foo, Zoom { }
(l) _ interface Vroom : Fi, Baz { }
(m) _ class Yow : Foo, Fi { }

(4) Considérer les classes suivantes et indiquer la/les réponses correcte(s) si l'instruction proposée est insérée en ligne 8 : (1,5)

```
1  class program
2      {
3          public static void Main()
4          {
5              X x1 = new X();
6              X x2 = new Y();
7              Y y1 = new Y();
8              /* à remplacer*/
9          }
10      }
11     class X
12     {
13          public void do1(){}
14     }
15     class Y : X
16     {
17         public void do2() { }
18     }
```

	Erreur (compilation)	Erreur (exécution)	Ok
`x2.do2();`	■	□	□
`(Y) x2.do2();`	■	□	□
`((Y) x2).do2();`	□	□	■

(5) Considérer les classes suivantes et indiquer la/les réponses correcte(s) si l'instruction proposée est insérée en ligne 7 : (3,5)

```
1 class program
2      {
3          public static void Main()
4          {
5              A a = new A();
6              B b = new B();
7              /* à remplacer*/
8          }
9      }
10     class A
11     {
12          public virtual  void doTheJob(){}
13     }
14     class B : A
15     {
16         public override void doTheJob() { }
17     }
18     class C
19     {
20         public void doTheJob() { }
21     }
```

	Erreur (compilation)	Erreur (exécution)	Ok
`A x = a;`	☐	☐	■
`A x = b;`	☐	☐	■
`B x = a;`	■	☐	☐
`B x = b;`	☐	☐	■
`A x = (A) b;`	■	☐	☐
`B x = (B) a;`	☐	■	☐
`C x = (C) b;`	■	☐	☐