

Correction TP3 Dot Net

Exercise 1 :

```
/***** Classe Person *****/
class Person
{
    protected int age;
    public void sayHello()
    {
        Console.WriteLine("Hello");
    }
    public void SetAge(int n)
    {
        age = n;
    }
}

/***** Classe Student *****/
class Student : Person
{
    public void GoToClasses()
    {
        Console.WriteLine("I'm going to class.");
    }
    public void DisplayAge()
    {
        Console.WriteLine("My age is: {0} years old", age);
    }
}

/***** Classe Teacher *****/
class Teacher : Person
{
    private string subject;
    public void Explain()
    {
        Console.WriteLine("Explanation begins");
    }
}

/***** Programme Principal *****/
static void Main(string[] args)
{
    Person p = new Person();
    p.sayHello();
    Student s = new Student();
    s.GoToClasses();
    s.SetAge(15);
    s.sayHello();
    s.DisplayAge();
    Teacher t = new Teacher();
    t.SetAge(40);
    t.sayHello();
    t.Explain();
}
```

Exercice 2 :

/***** Classe Personne *****/

class Personne

```
{
    private string nom;
    private string prenom;
    private DateTime datenaissance;

    public string Nom
    {
        get { return nom; }
        set { nom = value; }
    }
    public string Prenom
    {
        get { return prenom; }
        set { prenom = value; }
    }
    public DateTime Datenaissance
    {
        get { return datenaissance; }
        set { datenaissance = value; }
    }

    public Personne(string n, string p, DateTime dn)
    {
        nom = n;
        prenom = p;
        datenaissance = dn;
    }

    public virtual void Afficher()
    {
        Console.Out.Write("Nom: " + nom + " Prénom: " + prenom + " Date de
naissance: " + datenaissance);
    }
}
```

/***** Classe Employe *****/

class Employe : Personne

```
{
    private double salaire;

    public double Salaire
    {
        get { return salaire; }
        set { salaire = value; }
    }

    public Employe(string n, string p, DateTime dn, double s): base(n, p, dn)
    {
        salaire = s;
    }

    public override void Afficher()
    {
        base.Afficher();
        Console.Out.WriteLine(" Salaire: " + salaire);
    }
}
```

```
/***** Classe Chef *****/
```

```
class Chef : Employe
{
    private string service;

    public string Service
    {
        get { return service; }
        set { service = value; }
    }

    public Chef(string n, string p, DateTime dn, double s, string ser)
        : base(n, p, dn, s)
    {
        service = ser;
    }

    public override void Afficher()
    {
        base.Afficher();
        Console.Out.Write(" Service: " + service);
    }
}
```

```
/***** Classe Chef *****/
```

```
class Directeur : Chef
{
    private string societe;

    public string Societe
    {
        get { return societe; }
        set { societe = value; }
    }

    public Directeur(string n, string p, DateTime dn, double s, string ser, string
soc)
        : base(n, p, dn, s, ser)
    {
        societe = soc;
    }

    public override void Afficher()
    {
        base.Afficher();
        Console.Out.Write(" Société: " + societe);
    }
}
```

```
/***** Programme Principal *****/
```

```
static void Main(string[] args)
{
    Personne[] P = new Personne[8];
    //Des Affectations utilisant le concept du polymorphisme
    P[0] = new Employe("NOM1", "PRENOM1", new DateTime(1985, 3, 4), 9000);
    P[1] = new Employe("NOM2", "PRENOM2", new DateTime(1984, 3, 4), 10000);
    P[2] = new Employe("NOM3", "PRENOM3", new DateTime(1983, 3, 4), 8000);
    P[3] = new Employe("NOM4", "PRENOM4", new DateTime(1982, 3, 4), 11000);
    P[4] = new Employe("NOM5", "PRENOM5", new DateTime(1981, 3, 4), 7000);
}
```

```

        P[5] = new Chef("NOM6", "PRENOM6", new DateTime(1988, 7, 6), 9000,
"SERVICE1");
        P[6] = new Chef("NOM7", "PRENOM7", new DateTime(1984, 7, 6), 9000,
"SERVICE2");
        P[7] = new Directeur("NOM8", "PRENOM8", new DateTime(1988, 7, 6), 9000,
"SERVICE3", "SOCIETE1");

        for (int i = 0; i < P.Length; i++)
        {
            P[i].Afficher();           //Appel d'une méthode polymorphique (qui prend
une nouvelle forme dans les classes dérivées)
            Console.Out.WriteLine("");
        }

        foreach (Personne p in P)
        {
            p.Afficher();
            Console.Out.WriteLine("");
        }

    }

```

Exercise 3 :

```

/***** Classe Employe *****/
abstract class Employe
{
    private int matircule;
    private string nom;
    private string prenom;
    private DateTime datenaissance;

    public int Matircule
    {
        get { return matircule; }
        set { matircule = value; }
    }

    public string Nom
    {
        get { return nom; }
        set { nom = value; }
    }

    public string Prenom
    {
        get { return prenom; }
        set { prenom = value; }
    }

    public DateTime Datenaissance
    {
        get { return datenaissance; }
        set { datenaissance = value; }
    }

    public Employe(int matricule, string nom, string prenom, DateTime dn)
    {
        this.matircule = matricule;
        this.nom = nom;
    }
}

```

```

        this.prenom = prenom;
        this.datenaissance = dn;
    }

    public override string ToString()
    {
        return "Matricule: " + matircule + " Nom: " + nom + " Prénom: " + prenom +
        "Date de naissance: " + datenaissance.ToShortDateString();
    }

    public abstract double GetSalaire();
}

/***** Classe Ouvrier *****/
class Ouvrier : Employe
{
    private DateTime dateentree;
    private static double sMIG = 2500;

    public static double SMIG
    {
        get { return sMIG; }
    }

    public DateTime Dateentree
    {
        get { return dateentree; }
        set { dateentree = value; }
    }

    public Ouvrier(int m, string n, string p, DateTime dn, DateTime de)
        : base(m, n, p, dn)
    {
        dateentree = de;
    }

    public override string ToString()
    {
        return "Employé:" + " " + base.ToString() + " Date d'entrée: " +
        dateentree.ToShortDateString();
    }

    public override double GetSalaire()
    {
        double salaire;
        int Anciennete = DateTime.Now.Year - dateentree.Year;

        if (sMIG + Anciennete * 100 <= 2 * sMIG)
            salaire = sMIG + Anciennete * 100;
        else
            salaire = sMIG * 2;

        return salaire;
    }
}

```

```

/***** Classe Cadre *****/

```

```

class Cadre : Employe
{
    private int indice;

    public int Indice
    {
        get { return indice; }
        set { indice = value; }
    }

    public Cadre(int matricule, string nom, string prenom, DateTime dn, int
indice)
        : base(matricule, nom, prenom, dn)
    {
        this.indice = indice;
    }

    public override string ToString()
    {
        return "Cadre: " + " " + base.ToString() + " Indice: " + indice;
    }

    public override double GetSalaire()
    {
        if (indice == 1)
            return 13000;
        else if (indice == 2)
            return 15000;
        else if (indice == 3)
            return 17000;
        else if (indice == 4)
            return 20000;
        else
            return -1; //Problème d'indice
    }
}
/***** Classe Patron *****/
class Patron : Employe
{
    private static double ca;
    private double pourcentage;

    public static double Ca
    {
        get { return ca; }
        set { ca = value; }
    }

    public double Pourcentage
    {
        get { return pourcentage; }
        set { pourcentage = value; }
    }

    public Patron(int matricule, string nom, string prenom, DateTime dn, double p)
        : base(matricule, nom, prenom, dn)
    {
        this.pourcentage = p;
    }

    public override string ToString()
    {

```

```

        return "Patron: " + " " + base.ToString() + " Pourcentage: " + pourcentage
+ "%";
    }

    public override double GetSalaire()
    {
        return Math.Round((ca * pourcentage / 100) / 12, 2);
    }
}

/***** Classe Programme Principal *****/
static void Main(string[] args)
{
    Ouvrier o = new Ouvrier(1, "Nom1", "Prenom1", new DateTime(1980, 2, 3),
        new DateTime(2000, 4, 5));
    Console.Out.WriteLine(o);
    Console.Out.WriteLine("Salaire de l'employé: " + o.GetSalaire());
    Console.Out.WriteLine("*****");
    Cadre c = new Cadre(2, "Nom2", "Prenom2", new DateTime(1986, 4, 3), 3);
    Console.Out.WriteLine(c);
    Console.Out.WriteLine("Le salaire du cadre est: " + c.GetSalaire());
    Console.Out.WriteLine("*****");
    Patron.Ca = 17000000;
    Patron p = new Patron(3, "Nom3", "Prenom3", new DateTime(1970, 6, 6), 3);
    Console.Out.WriteLine(p);
    Console.Out.WriteLine("Le salaire du patron est: " + p.GetSalaire());
}

```

Exercice 4 :

```

/***** Classe Vecteur2D *****/
class Vecteur2D
{
    private double x;
    private double y;
    private static int nb_vecteurs = 0;

    public double X
    {
        get { return x; }
        set { x = value; }
    }

    public double Y
    {
        get { return y; }
        set { y = value; }
    }

    public static int Nb_vecteurs
    {
        get { return nb_vecteurs; }
    }

    public Vecteur2D() { nb_vecteurs++; }

    public Vecteur2D(double x, double y)
    {
        this.x = x;
        this.y = y;
        nb_vecteurs++;
    }
}

```

```

public Vecteur2D(Vecteur2D v)
{
    x = v.x;
    y = v.y;
    nb_vecteurs++;
}

public override string ToString()
{
    return "X=" + x + " Y=" + y;
}

public virtual double Norme()
{
    return Math.Sqrt(x * x + y * y);
}

public override bool Equals(object obj)
{
    if (obj == null) //Comparer à null
        return false;
    else if (obj == this) //Comparer à lui même
        return true;
    else if (obj.GetType() != this.GetType()) //Deux types incomparables
        return false;
    else
    {
        Vecteur2D v = (Vecteur2D)obj; //Convertir l'objet en
        if (this.x == v.x && this.y == v.y) //Comparer le contenu
            return true;
        else
            return false;
    }
}
}

vecteur

/***** Classe Vecteur3D *****/
class Vecteur3D : Vecteur2D
{
    private double z;

    public double Z
    {
        get { return z; }
        set { z = value; }
    }

    public Vecteur3D() : base() { }

    public Vecteur3D(double x, double y, double z)
        : base(x, y)
    {
        this.z = z;
    }

    public Vecteur3D(Vecteur3D v)
        : base(v.X, v.Y)
    {
        z = v.z;
    }
}

```



```

    public override string ToString()
    {
        return base.ToString() + " Z=" + z;
    }

    public override bool Equals(object obj)
    {
        if (obj == null) //Comparer à null
            return false;
        else if (obj == this) //Comparer à lui même
            return true;
        else if (obj.GetType() != this.GetType()) //Deux types incomparables
            return false;
        else
        {
            Vecteur3D v = (Vecteur3D)obj; //Convertir
l'objet en vecteur
            if (this.X == v.X && this.Y == v.Y && this.Z == v.Z) //Comparer
le contenu
                return true;
            else
                return false;
        }
    }

    public override double Norme()
    {
        return Math.Sqrt(X * X + Y * Y + z * z);
    }
}

/***** Classe Programme Principal *****/
static void Main(string[] args)
{
    double x, y, z;
    Console.Out.WriteLine("****Vecteur 2D****");
    Console.Out.WriteLine("Vecteur 1:");
    Console.Out.Write("Donner X: ");
    x = double.Parse(Console.In.ReadLine());
    Console.Out.Write("Donner Y: ");
    y = double.Parse(Console.In.ReadLine());
    Vecteur2D V1 = new Vecteur2D(x, y);
    Console.Out.WriteLine(V1);
    Console.Out.WriteLine("La norme est: " + V1.Norme());

    Console.Out.WriteLine("Vecteur 2:");
    Console.Out.Write("Donner X: ");
    x = double.Parse(Console.In.ReadLine());
    Console.Out.Write("Donner Y: ");
    y = double.Parse(Console.In.ReadLine());
    Vecteur2D V2 = new Vecteur2D(x, y);
    Console.Out.WriteLine(V2);
    Console.Out.WriteLine("La norme est: " + V2.Norme());

    if (V1.Equals(V2))
        Console.Out.WriteLine("Les deux vecteurs sont identiques");
    else
        Console.Out.WriteLine("Non identiques");

    Console.Out.WriteLine("****Vecteur 3D****");
}

```

```

        Console.Out.WriteLine("Vecteur 1:");
        Console.Out.Write("Donner X: ");
        x = double.Parse(Console.In.ReadLine());
        Console.Out.Write("Donner Y: ");
        y = double.Parse(Console.In.ReadLine());
        Console.Out.Write("Donner Z: ");
        z = double.Parse(Console.In.ReadLine());
        Vecteur2D V3 = new Vecteur3D(x, y, z);
        Console.Out.WriteLine(V3);
        Console.Out.WriteLine("La norme est: " + V3.Norme());

        Console.Out.WriteLine("Vecteur 2:");
        Console.Out.Write("Donner X: ");
        x = double.Parse(Console.In.ReadLine());
        Console.Out.Write("Donner Y: ");
        y = double.Parse(Console.In.ReadLine());
        Console.Out.Write("Donner Z: ");
        z = double.Parse(Console.In.ReadLine());
        Vecteur2D V4 = new Vecteur3D(x, y, z);
        Console.Out.WriteLine(V4);
        Console.Out.WriteLine("La norme est: " + V4.Norme());

        if (V3.Equals(V4))
            Console.Out.WriteLine("Les deux vecteurs sont identiques");
        else
            Console.Out.WriteLine("Non identiques");

        Console.Out.WriteLine("Le nombre de vecteurs créés est: " +
            Vecteur2D.Nb_vecteurs);
    }
}

```

Exercice 5 :

```

/***** Classe Compte *****/
class Compte
{
    private int code;
    private double solde;
    private static int nb_comptes = 0;

    public int Code
    {
        get { return code; }
    }

    public double Solde
    {
        get { return solde; }
    }

    public static int Nb_Comptes
    {
        get { return nb_comptes; }
    }

    public Compte()
    {
        nb_comptes++;
        code = nb_comptes;
    }

    public Compte(double solde)

```

```

    {
        nb_comptes++;
        code = nb_comptes;
        this.solde = solde;
    }

    public virtual void deposer(double somme)           //méthode virtuelle
qui peut être redéfinie dans une classe dérivée
    {
        solde += somme;
    }

    public virtual void retirer(double somme)           ///méthode virtuelle
qui peut être redéfinie dans une classe dérivée
    {
        solde -= somme;
    }

    public override string ToString()
    {
        return "Code: " + code + " Solde: " + solde;
    }
}

/***** Classe CompteEpargne *****/

class CompteEpargne : Compte
{
    private double tauxinteret = 6;

    public double Tauxinteret
    {
        get { return tauxinteret; }
    }

    public CompteEpargne() : base() { }           //constructeur par défaut

    public CompteEpargne(double solde) : base(solde) { } //constructeur
d'initialisation

    public void CalculerInteret() //une nouvelle méthode qui utilise une méthode
héritée
    {
        deposer((Solde * tauxinteret) / 100);
    }

    public override string ToString() //redéfinition de la méthode ToString()
    {
        return "Compte Epargne: " + base.ToString() + " Taux intérêt: " +
tauxinteret;
    }
}

/***** Classe ComptePayant *****/

class ComptePayant : Compte
{
    public ComptePayant() : base() { }           //Constructeur par défaut
    public ComptePayant(double solde) : base(solde) { } //Constructeur d'initon
    public override string ToString()           //ToString redéfinie
    {
        return "Compte Payant: " + base.ToString();
    }
}

```

```

    }

    public override void deposter(double somme)    //La méthode "deposer" redéfinie
    {
        base.deposer(somme);
        base.retirer(5);
    }

    public override void retirer(double somme)    //La méthode "retier" reféfinie
    {
        base.retirer(somme);
        base.retirer(5);
    }
}

/***** Classe Programme Principal *****/
static void Main(string[] args)
{
    Compte C1 = new Compte();
    CompteEpargne C2 = new CompteEpargne();
    ComptePayant C3 = new ComptePayant();
    C1.deposer(10000);
    C2.deposer(2000);
    C3.deposer(3000);
    C1.retirer(2000);
    C2.retirer(500);
    C3.retirer(400);
    C2.CalculerInteret();
    Console.Out.WriteLine(C1);
    Console.Out.WriteLine(C2);
    Console.Out.WriteLine(C3);
}

```

Exercice 6 :

```

/***** Classe Batiment *****/

class Batiment
{
    private string adresse;

    public string Adresse
    {
        get { return adresse; }
        set { adresse = value; }
    }

    public Batiment()
    {
    }

    public Batiment(string adresse)
    {
        this.adresse = adresse;
    }

    public override string ToString()
    {
        return "Adresse: " + adresse;
    }
}

```

```

/***** Classe Maison *****/

class Maison : Batiement
{
    private int nbPiece;

    public int NbPiece
    {
        get { return nbPiece; }
        set { nbPiece = value; }
    }

    public Maison() : base()
    {
    }

    public Maison(string adresse, int nb)
        : base(adresse)
    {
        nbPiece = nb;
    }

    public override string ToString()
    {
        return base.ToString() + " Nombre de pièces:" + nbPiece;
    }
}

/***** Classe Programme Principal *****/
static void Main(string[] args)
{
    Batiement B = new Batiement("Marrakech");
    Console.Out.WriteLine(B);
    Maison M1 = new Maison("Marrakech", 4);
    Console.Out.WriteLine(M1);
    Maison M2 = new Maison();
    M2.Adresse = "Rabat";
    M2.NbPiece = 3;
    Console.Out.WriteLine(M2);
}

```

Exercice 7 :

```

using System;

class House
{
    protected int surface;

    protected Door door;
}

```

```
public House(int surface)

{

    this.surface = surface;

    door = new Door();

}


public int Surface

{

    get { return surface; }

    set { surface = value; }

}


public Door Door

{

    get { return door; }

    set { door = value; }

}


public virtual void Display()

{

    Console.WriteLine("Je suis une maison, ma surface est de {0} m2.",surface);

}

}
```

```
class Door
{
    protected string color;

    public Door()
    {
        color = "blue";
    }

    public Door(string color)
    {
        this.color = color;
    }

    public string Color
    {
        get { return color; }
        set { color = value; }
    }

    public void Display()
    {
        Console.WriteLine("Je suis une porte, ma couleur est {0}.", color);
    }
}
```

```
class Apartment : House
{
    public Apartment() : base (50)
    {
    }

    public override void Display()
    {
        Console.WriteLine("Je suis un appartement, ma surface est " +
            surface + " m2");
    }
}

class Person
{
    protected string name;
    protected House house;

    public Person()
    {
        name = "Thomas";
        house = new House(150);
    }

    public Person(string name, House house)
    {
        this.name = name;
        this.house = house;
    }
}
```



```
}

public string Name

{
    get { return name; }
    set { name = value; }
}

public House House

{
    get { return house; }
    set { house = value; }
}

public void Display()

{
    Console.WriteLine("Je m'appelle {0}.", name);

    house.Display();

    house.Door.Display();
}
}

class Test

{
    static void Main()

    {
```

```
Apartment MyApartament = new Apartment();

Person person = new Person();


person.Name = "Thomas";

person.House = MyApartament;

person.Display();

}

}
```

Exercise 8 :

```
namespace transport {
    public class Marchandise {
        private int numero;
        public int Numero {
            get { return numero; } set { numero = value; }
        }
        private double poids;
        public double Poids {
            get { return poids; } set { poids = value; }
        }
        private double volume;
        public double Volume {
            get { return volume; } set { volume = value; }
        }
        public Marchandise(int num, double p, double v)
        {
            this.numero = num; this.poids = p; this.volume = v;
        }
        public override string ToString() {
            return "Num=" + numero + " Poids=" + poids + " vol=" + volume;
        }
    }
}
```

```
namespace transport{
    public abstract class Cargaison {
        protected int distance;
        protected List<Marchandise> marchandises = new List<Marchandise>();
        public List<Marchandise> getMarchandises() { return marchandises; }
        public Cargaison(int d) { this.distance = d; }
        public void Add(Marchandise m) { marchandises.Add(m); }
        public void afficher() {
            foreach (Marchandise m in marchandises) {
                Console.WriteLine(m.ToString());
            }
        }
        public Marchandise GetMarchandises(int num) {
            foreach (Marchandise m in marchandises) {
                if (m.Numero == num) return m;
            }
            return null;
        }
    }
}
```

```

public double GetPoidsTotal() {
    double P = 0;
    foreach (Marchandise m in marchandises) {
        P += m.Poids;
    }
    return P;
}

public double GetVolumeTotal() {
    double V = 0;
    foreach (Marchandise m in marchandises) {
        V += m.Poids;
    }
    return V;
}

public abstract double cout();
}
}

```

```

namespace transport {
    class CargaisonAerienne : Cargaison
    {
        public CargaisonAerienne(int d): base(d) { }
        public override double cout()
        {
            if (GetVolumeTotal() < 80000)
                return 10 * distance * GetVolumeTotal();
            else
                return 12 * distance * GetPoidsTotal();
        }
        public override string ToString(){
            return "Cargaison Aérienne Distance =" + distance;
        }
    }
}
}

```

```

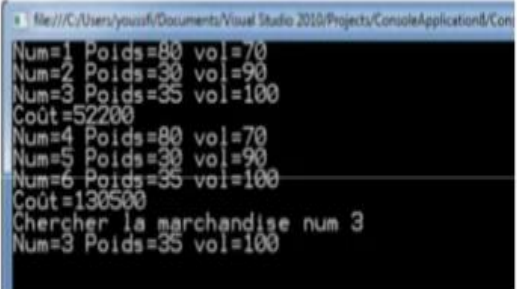
namespace transport {
    class CargaisonRoutiere :Cargaison
    {
        public CargaisonRoutiere(int d):base(d) { }
        public override double cout() {
            if (GetVolumeTotal() < 380000)
                return 4 * distance * GetVolumeTotal();
            else
                return 6 * distance * GetPoidsTotal();
        }
        public override string ToString(){
            return "Cargaison Routière Distance =" + distance;
        }
    }
}

```

```

namespace test {
    class Program {
        static void Main(string[] args) {
            Cargaison cr = new CargaisonRoutiere(90);
            cr.Add(new Marchandise(1, 80, 70));
            cr.Add(new Marchandise(2, 30, 90));
            cr.Add(new Marchandise(3, 35, 100));
            cr.afficher();
            Console.WriteLine("Coût="+cr.cout());
            Cargaison ca = new CargaisonAerienne(90);
            ca.Add(new Marchandise(4, 80, 70));
            ca.Add(new Marchandise(5, 30, 90));
            ca.Add(new Marchandise(6, 35, 100));
            ca.afficher();
            Console.WriteLine("Coût=" + ca.cout());
            Console.WriteLine("Chercher la marchandise num 3");
            Marchandise m = cr.GetMarchandises(3);
            Console.WriteLine(m);
            Console.ReadLine();
        }
    }
}

```



```

File:///C:/Users/yousif/Documents/Visual Studio 2010/Projects/ConsoleApplication8/Con
Num=1 Poids=80 vol=70
Num=2 Poids=30 vol=90
Num=3 Poids=35 vol=100
Coût=52200
Num=4 Poids=80 vol=70
Num=5 Poids=30 vol=90
Num=6 Poids=35 vol=100
Coût=130500
Chercher la marchandise num 3
Num=3 Poids=35 vol=100

```