Programmation orienté objets en C#

Maher REBAI: Enseignant en informatique à
Ecole supérieure d'ingénieurs Léonard de Vinci
Paris la défense(ESILV)
ECE de Paris

Année Universitaire : 2018-2019

Chapitres

- Révision
- Concept de classe
- Collections
- Héritage/polymorphisme
- Classe abstraite et interface
- La délégation
- Suivi de projet

Classe abstraite et interface

Exercice

- On souhaite réaliser un programme qui gère différentes Formes géométriques. Chaque objet Forme est caractérisé par une couleur. Il peut également se décrire à travers des propriétés (tel que la spécification du rayon pour un cercle ou la spécification du coté pour un carré etc.). En particulier, on peut enregistrer dans ce programme des cercles, des carrés, des rectangles et des losanges.

  Tracer le diagramme de classe pour ce programme

  Implémenter le en C#.

Comment faire ?

- Déclarer une classe Forme qui contient l'attribut couleur.
- Créer la méthode SeDécrire.
- Créer les classes filles Carre, Cercle, triangle, losange qui hérite de Forme.

Remarques:

- Dans le programme, on n'a pas besoin d'objet Forme. Donc inutile de créer une classe Forme instanciable.
- Les corps de la méthode SeDécrire est à définir dans les classes filles et non dans la classe mère.
- Solution: la classe Animal devrait être définie **abstraite**.

Les classes abstraites

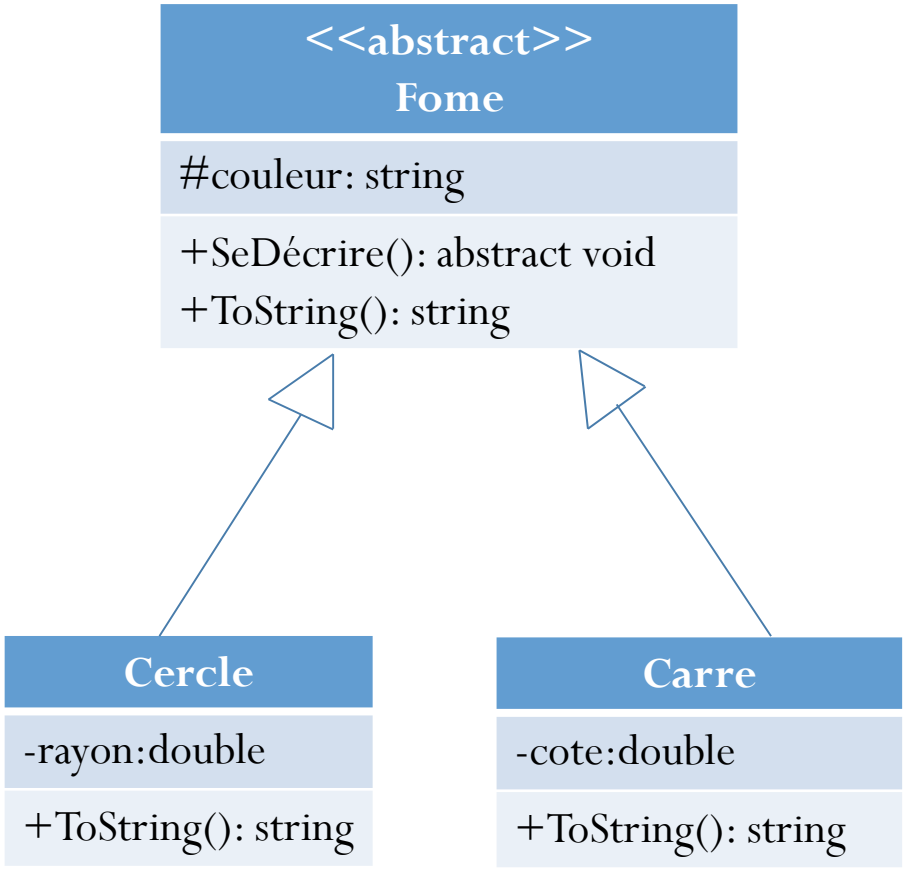
- Une classe abstraite est une classe qui ne permet pas d'instancier des objets.
- Elle ne peut servir que de classe de base pour une dérivation.
- Elle peut contenir des méthodes et des attributs dont hériteront les classes filles.
- Elle peut contenir des méthodes abstraites.

Les classes abstraites

- Une méthode abstraite est définie dans la classe mère uniquement par sa signature.
- Le corps de la méthode abstraite s'écrit différemment d'une classe fille à une autre.
- Les méthodes abstraites sont toujours publiques
- L'implémentation d'un méthode abstraite dans chaque classe fille est obligatoire.

Les interfaces

- **Diagramme des classes**



Les classes abstraites

- **Implémentation en C#**

```
abstract class Forme
{
    //Attributs
    protected string couleur;
    //Constructeur
    public Forme(string couleur)
    { this.couleur = couleur;}
    //Méthodes
    abstract public void SeDecrire();
    public override string ToString()
    {
        return "La Forme est de couleur :"+couleur;
    }

}
```

Les classes abstraites

- **Implémentation en C#**

```
class Cercle:Forme
{   //Attributs
    private double rayon;
    //Constructeur
    public Cercle(string couleur, double rayon):base(couleur)
    { this.rayon = rayon; }
    //Méthodes
    public override void SeDecrire()
    {
        Console.WriteLine("Je suis un cercle: \nPérimètre = " +
            "2*PI*rayon\nAire = PI *rayon^2\n");
    }
    public override string ToString()
    {
        return base.ToString()+". Il s'agit d'un cercle de " +
            "rayon :" + rayon;
    }
}
```

Les classes abstraites

- **Implémentation en C#**

```
class Carre:Forme
{ //Attributs
    private double cote;
    //Constructeur
    public Carre(string couleur, double cote) : base(couleur)
    { this.cote = cote; }
    //Méthodes
    public override void SeDecrire()
    {
        Console.WriteLine("Je suis un carre: \nPérimètre = " +
            "4*cote\nAire = cote*cote\n");
    }
    public override string ToString()
    {
        return base.ToString() + ". Il s'agit d'un carré de " +
            "coté :" + cote;
    }
}
```

Les classes abstraites

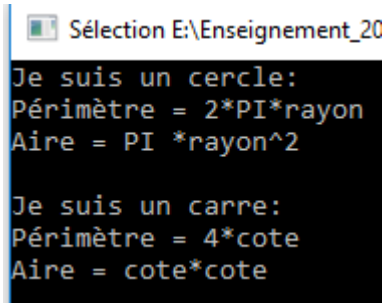
- **Implémentation en C#**

```
static void Main(string[] args)
{
    Forme f1 = new Forme("Bleu");
    Console.ReadKey();
}
```

Forme.Forme(string couleur)

Impossible de créer une instance de la classe abstraite ou de l'interface 'Forme'

```
static void Main(string[] args)
{
    Forme f1 = new Cercle("Bleu",15);
    Carre f2 = new Carre("Rouge", 12);
    f1.SeDecrire();
    f2.SeDecrire();
    Console.ReadKey();
}
```



Exercice

- On souhaite réaliser un programme qui gère différents types d'animaux. Chaque objet animal est caractérisé par une couleur, un poids et un nom. Il peut également décrire sa façon de se déplacer de manger et de crier. En particulier, on peut enregistrer dans ce programme des chiens, des canards et des chats.

  Tracer le diagramme de classe pour ce programme

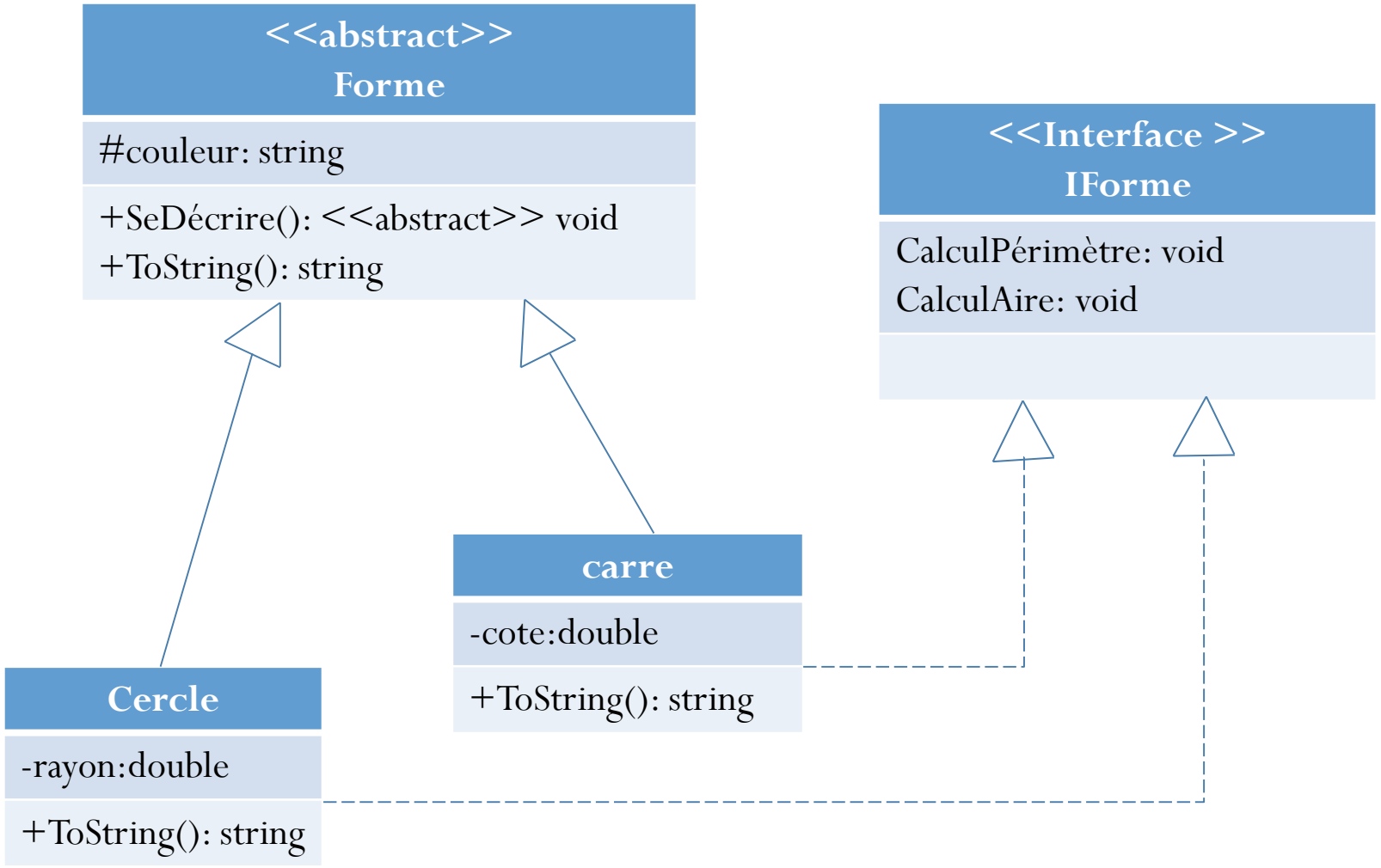
  Implémenter le en C#.

Les interfaces

- Une interface est une classe dans laquelle on ne retrouve que :
  - des méthodes abstraites
- Les méthodes d'une interface sont nécessairement publiques et abstraites.
- Une classe ne peut hériter qu'une seule fois (d'une classe abstraite ou non).
- Une classe peut implémenter d'autant d'interfaces que souhaité

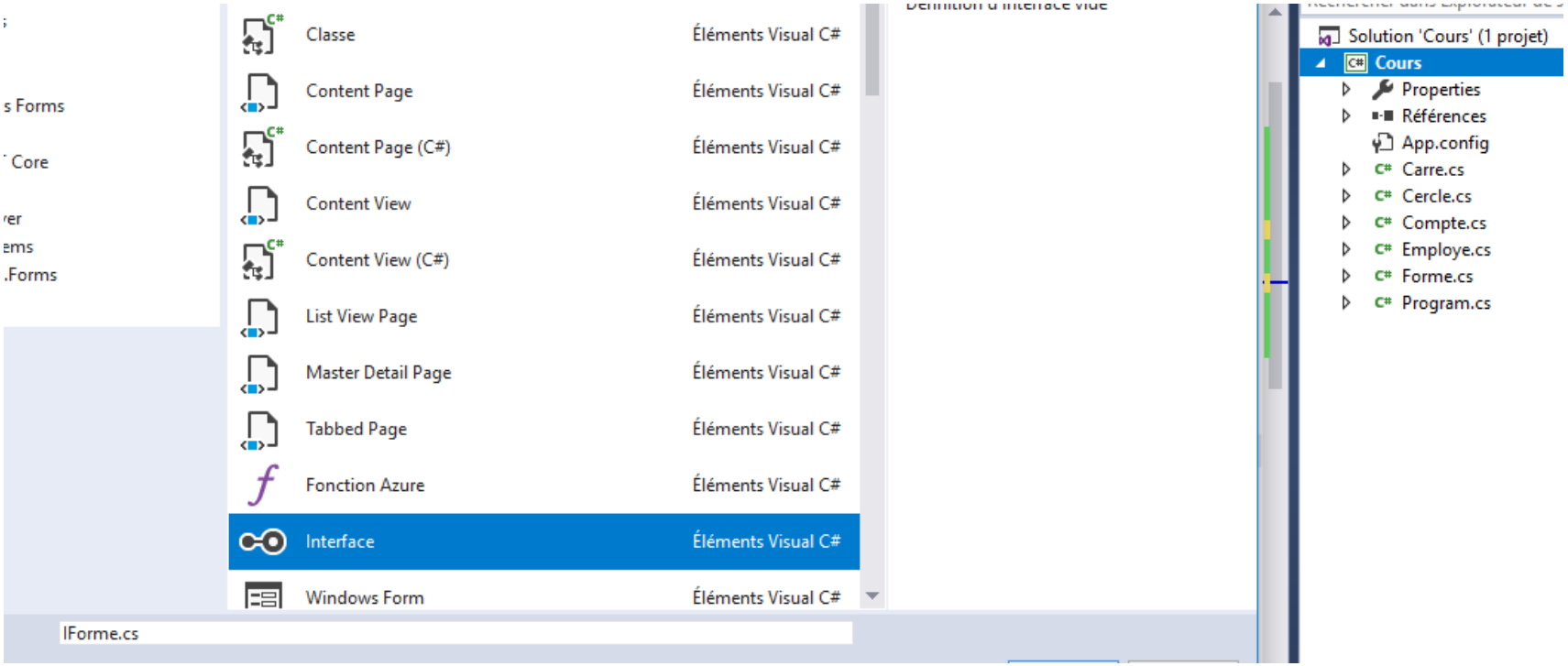
Les interfaces

- **Diagramme des classes**



Les interfaces

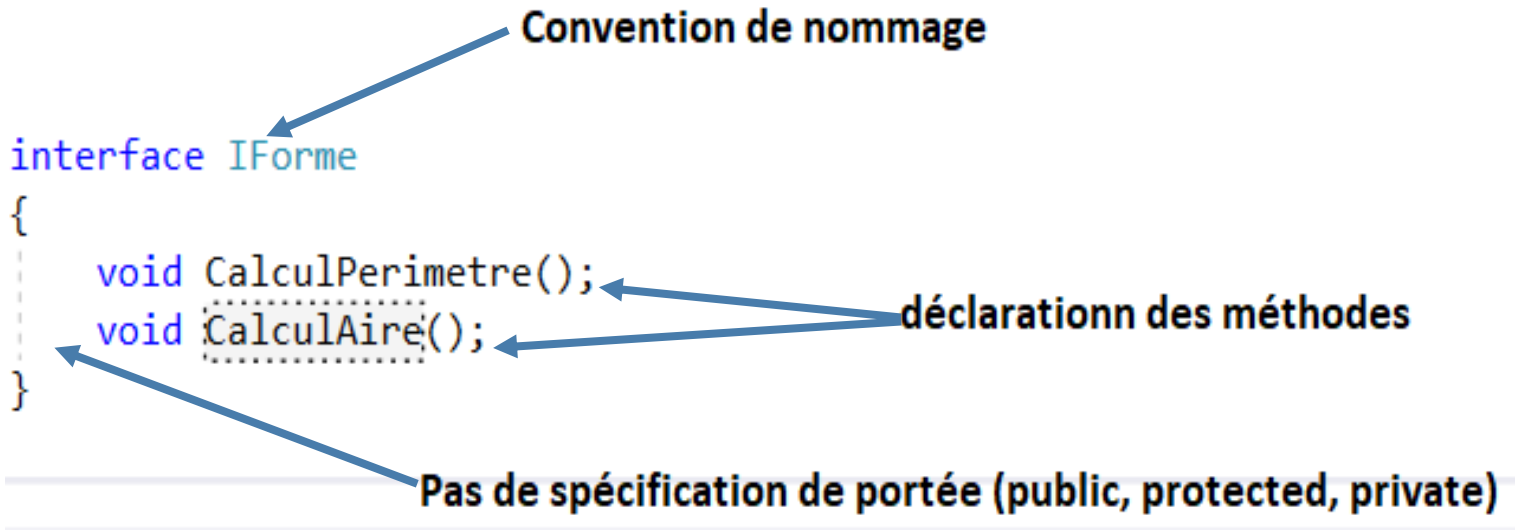
- **Implémentation en C# (définition)**




```
interface IForme
{
    void CalculPerimetre();
    void CalculAire();
}
```

Les interfaces

- **Implémentation en C# (définition)**



Les interfaces

- **Implémentation en C# (définition)**

```
class Cercle:Forme,IForme
{   //Attributs
    private double rayon;
    //Constructeur
    public Cercle(string couleur, double rayon):base(couleur)
    { this.rayon = rayon; }
    //Méthodes
    public override void SeDecrire()
    {Console.WriteLine("Je suis un cercle: \nPérimètre = " +
            "2*PI*rayon\nAire = PI *rayon^2\n");
    }
    public override string ToString()
    {return base.ToString()+". Il s'agit d'un cercle de " +
            "rayon :" + rayon;
    }
    public void CalculPerimetre()
    { Console.WriteLine("Périmètre = " + 2*3.14*rayon); }
    public void CalculAire()
    { Console.WriteLine("Aire = " + 3.14*rayon*rayon); }
}
```

Les interfaces

- **Implémentation en C# (définition)**

```
class Carre:Forme,IForme
{ //Attributs
    private double cote;
    //Constructeur
    public Carre(string couleur, double cote) : base(couleur)
    { this.cote = cote; }
    //Méthodes
    public override void SeDecrire()
    {Console.WriteLine("Je suis un carre: \nPérimètre = " +
            "4*cote\nAire = cote*cote\n");
    }
    public override string ToString()
    { return base.ToString() + ". Il s'agit d'un carré de " +
            "coté :" + cote;
    }
    public void CalculPerimetre()
    {Console.WriteLine("Périmètre = " + 4*cote);}
    public void CalculAire()
    { Console.WriteLine("Aire = " + cote * cote); }
}
```

Les interfaces

- **Implémentation en C# (définition)**

```
static void Main(string[] args)
{
    Forme f1 = new Cercle("Bleu",15);
    Carre f2 = new Carre("Rouge", 12);
    f1.SeDecrire();
    f2.SeDecrire();
    f1.
    Con                                    ;
}
```

Equals
GetHashCode
GetType
SeDecrire — void Forme.SeDecrire()
ToString

```
static void Main(string[] args)
{
    Forme f1 = new Cercle("Bleu",15);
    Carre f2 = new Carre("Rouge", 12);
    f1.SeDecrire();
    f2.SeDecrire();
    f2.
    Con
}
```

CalculAire
CalculPerimetre
Equals
GetHashCode
GetType
SeDecrire — void Carre.SeDecrire()
ToString

```
static void Main(string[] args)
{
    Forme f1 = new Cercle("Bleu",15);
    Carre f2 = new Carre("Rouge", 12);
    f1.SeDecrire();
    f2.SeDecrire();
    f2.CalculAire();
    f2.CalculPerimetre();
    Console.ReadKey();
}
```

```
E:\Enseignement_2018_2019\E
Je suis un cercle:
Périmètre = 2*PI*rayon
Aire = PI *rayon^2

Je suis un carre:
Périmètre = 4*cote
Aire = cote*cote

Aire = 144
Périmètre = 48
```

Les interfaces

- **Notes**
  - Une interface ne s'instancie jamais
  - Une interface ne contient pas d'attributs ni de constructeur ni propriétés.
  - Par défaut, les méthodes dans une interfaces sont public. Cependant, on ne met jamais de portée devant une méthodes (ni public, ni private, ni protected).
  - On ne peut pas définir une classe dans une interface, ni énumération, etc.
  - Une interface ne peut pas hériter (ni d'une classse ni d'une interface)

Exercice

- On se propose de créer un POO pour la gestion des comptes d'une banque. Un compte est caractérisé par un numeroCompte, un nomClient, un prénomClient, une adresseClient et un Solde. La banque propose à ses clients quatre types de compte. Le premier type de compte, « CompteChèque », ne rapporte pas d'intérêt. Cependant, ce type de compte nécessite des frais de tenu de compte. Les frais sont de 10 euro pour les salariés et 5 euros pour les étudiant. Le deuxième type de compte, « Livret_A_Jeune », génère des intérêts chaque mois selon un taux de 0,7%. Le troisième type de compte, « Livret_A_Majeur », génère des intérêts chaque mois selon un taux de 0,9%.

Exercice

- Créer le diagramme de classe pour ce programme sachant que chaque compte devrait être capable d'afficher le nom et le prénom du client ainsi que le solde du compte. Pour les compteChèque, on veut avoir la possibilité de modifier les frais de tenu de compte. Enfin, il faut créer pour chaque type de compte une méthode ToString() qui permet de décrire le compte.
- Traduisez le diagramme de classe en langage C#.