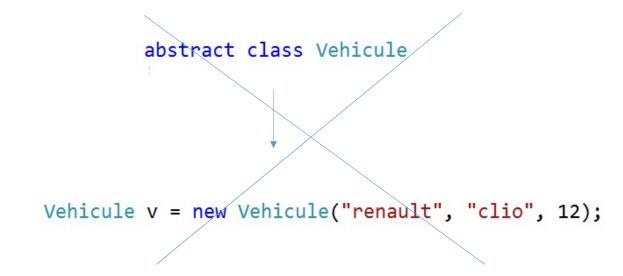
## Interface

Aline Ellul

#### Classe Abstraite

- Une classe abstraite ne peut pas être instanciée
- Une classe abstraite ne peut pas être sealed



#### Classe Abstraite

- Une classe abstraite ne peut pas être instanciée
- Une classe abstraite ne peut pas être sealed
- Une classe abstraite peut contenir des méthodes et des accesseurs classiques (vus auparavant)
- Une classe abstraite peut contenir des
  - Méthodes abstraites
  - Accesseurs abstraits
  - Les membres abstraits ne peuvent pas être privés

```
abstract class Vehicule
     protected string marque;
     protected string modele;
     protected int puissance;
      public int Puissance
     { get { return puissance; } }
      public Vehicule(string marque, string modele, int puissance)
         this.marque = marque;
         this.modele = modele;
         this.puissance = puissance;
      public override string ToString()
         return "\n " + " Marque " + marque + " Modele " + modele + " Puissance " + puissance;
                                                                                                                                                    Méthode
                                                                                                                                                     abstraite
      abstract public double taxe(double facteur);
   class VoitureLocation : Vehicule
       double kilometre;
       public VoitureLocation(string marque, string modele, int puissance, double kilometrage)
          : base(marque, modele, puissance)
          this.marque = marque;
          this.modele = modele;
          this.puissance = puissance;
          this.kilometre = kilometrage;
                                                                                                                                              Implémentation
       public double prixLocation(double facteur)
       { return facteur * kilometre; }
                                                                                                                                              obligatoire dans
       public override double taxe(double facteur)
       { return facteur * kilometre; }
                                                                                                                                              les classes filles
       public override string ToString()
          string retour = base.ToString() + " kilometrage " + kilometre;
          return retour;
```

#### Méthode abstraite

- Définition de la signature exclusivement à définir comme abstraite
- Une méthode abstraite est de facto virtual
- Possible que dans une classe abstraite
- Une méthode abstraite n'est que méthode d'instance
- A « overrider » dans la classe fille et à implémenter comme dans l'exemple précédent

## Propriété abstraite

```
public abstract int Puissance
{
    get;
}
public abstract int Puissance
{
    get { return puissance; }
}
```

## Propriété abstraite

- Définition de la signature exclusivement à définir comme abstraite
- Une propriété abstraite est de facto virtual
- Possible que dans une classe abstraite
- A « overrider » dans la classe fille et à implémenter comme dans l'exemple précédent

#### Interface

- Objectif :
  - Pouvoir créer un ensemble de comportements communs à différents types de classes.
- Comble les lacunes de l'héritage simple
- Permet de proposer aux classes qui n'ont aucun socle commun de partager des méthodes et propriétés communes ceci afin d'avoir une homogénéisation des comportements

#### Définition d'une interface

```
interface IEgalite
{
  bool Egalite(double T);
}
```

- Mot clé : interface qui définit un groupe de méthodes
- Le nom de l'interface commence par un I (règle de nommage communément adoptée)
- Définition de signatures de méthodes et/ou de définition de propriétés à implémenter dans les classes filles

### Implémentation dans les classes filles

```
interface IEgalite
{
  bool Egalite(double T);
}

Implémentation
  dans la classe fille
```

```
class Salarie : IEgalite
    string nom;
    string prenom;
    double salaire;
    public Salarie (string nom, string prenom)
        this.nom = nom;
        this.prenom = prenom;
    public bool Egalite(double val)
        return salaire == val;
```

# Implémentation d'une interface dans une classe Abstraite

abstract class Vehicule : IEgalite abstract public bool Egalite(double a);

#### Exécution

```
VoitureLocation vl = new VoitureLocation("peugeot", "206", 12, 10000);
Salarie s = new Salarie("toto", "titi", 10000);
Console.WriteLine(s.Egalite(10000));
Console.WriteLine(vl.Egalite(10000));
```

## Interface avec variable de type

```
interface IComparaison<T>
{
    bool Egalite1(T item);
}
```

```
class Salarie : IEgalite,IComparaison<Salarie>
    public bool Egalite1(Salarie s)
    { return this.Egalite(s.salaire); }
```