Programmation orientée objet

Héritage

Classes abstraites

Interfaces

HÉRITAGE

Déclarer une sous-classe

- En Java : mot-clef extends
  En C# : symbole :

- Exemple :

```
class WebSite
{

}
class Blog : WebSite
{

}
```

Constructeur de la classe-fille

- En Java : mot-clef super et appel dans la méthode
  En C# : symbole **:** et appel dans l'en-tête de la méthode

- Exemple :

```
class WebSite
{
   public WebSite (string name) {
        this.name = name;
   }
}
class Blog : WebSite
{
   public WebSite (string name, string engine) : base(name) {
        this.engine = engine;
   }
}
```

Redéfinir une méthode

- En Java : aucun mot-clef nécessaire
  En C# : mots-clés virtual (mère) et override (redéfinition)

- Exemple :

```
class Personne
{
      public virtual void Conduire() {
        …;  // code en respectant le code de la route
      }
}
class Pilote : Personne
{
      public override void Conduire() {
        …; // code pour rouler sur circuit
      }
}
```

« Masquer » une méthode

- Pas d'équivalent en Java

- Exemple :

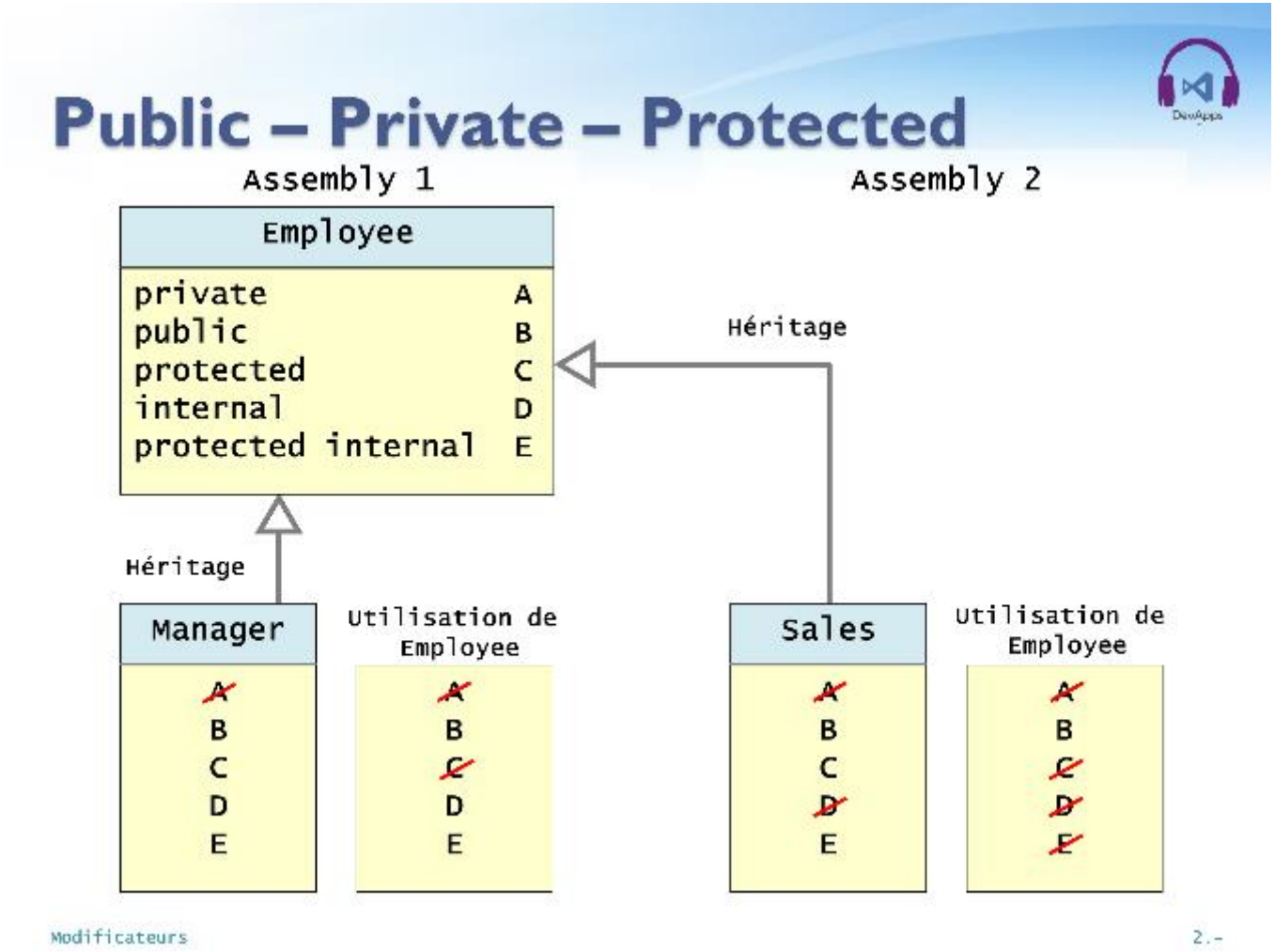
```
class Personne
{
      public void Conduire() {
        …;  // code en respectant le code de la route
      }
}
class Pilote : Personne
{
      public new void Conduire() {
        …; // code pour rouler sur circuit
      }
}
```

Propriétés = méthodes

- Même fonctionnement (par rapport à virtuel/override) :

```
class WebSite {
      public virtual string Url {
              get {
                      return $"http://www.{name}.com";
              }
      }
 }
class Blog : WebSite {
      public override string Url {
              get {
                      return base.Url + "/myblog";
              }
      }
}
```

Visibilité


Public – Private – Protected
Assembly 1
Assembly 2
Employee
private A
public B
protected C
internal D
protected internal E
Héritage
Héritage
Manager
A
B
C
D
E
Utilisation de Employee
A
B
C
D
E
Sales
A
B
C
D
E
Utilisation de Employee
A
B
C
D
E
Modificateurs
2.-

Remarques

- Comme en Java, pas de multihéritage

- Classe non dérivable (pas de sous-classe possible) ou méthode qui ne peut pas être redéfinie :
  - En Java, mot-clef : `final`
  - En C#, mot-clef : `sealed`

- Constante
  - En Java, mot-clef : `final`
  - En C#, deux mots-clefs : `const` et `readonly`

Mots-clefs const et readonly

- Mot-clef const : devant un attribut initialisé à la déclaration
  (donc même valeur pour toutes les instances)

```
public const int NB_PLACES_MAX = 15;
```

- Mot-clef readonly : attribut initialisé au plus tard dans le constructeur
  (permet donc des valeurs initiales différentes)

```
class Enfant {
   private readonly annéeInscription;
   public Enfant (…, int année)
   {
         ... ;
         annéeInscription = année;
   }
   public Enfant (…) : this (…, 2027)
   { }
}
```

CLASSES ABSTRAITES

Classe abstraite

- Non instanciable
- Peut contenir une à plusieurs méthodes abstraites

- Exemple :

```
abstract class MyAbstractClass
{
        ... abstract ... MyMethod();
}

class MyClass : MyAbstractClass
{
        ... override ... MyMethod()
        {
                ...//coding
        }
}
```

INTERFACES

Interface

- Ne contient que des méthodes ou propriétés abstraites
- Une classe peut implémenter plusieurs interfaces.
- Convention : nom commençant par `I`

- Exemple :

```
public interface ILecteurAudio {
        // propriété
        bool LectureEnCours { get; }
        // méthode
        void Lire(string cheminFichier);
}
… class LecteurMp4 : ILecteurAudio
{
        bool LectureEnCours { … }
        void Lire (string cheminFichier) { … }
}
```