DESIGN PATTERNS

CLASSE ABSTRAITE

INTERFACES

METHODE ABSTRAITE

- **Méthode non définie (sans corps dans la classe)**

```
 {
…
public abstract void dessine (int taille);
```

CLASSE ABSTRAITE

- **Contenant au moins une méthode abstraite**

```
abstract class Figure
{
…
public abstract void dessine (int taille);
```

- **Marquage *abstract* obligatoire**

CLASSE ABSTRAITE

- **Non instanciable**
- **Dérivable**
  - toutes les méthodes abstraites définies
    → classe complète et instanciable
  - il reste des méthodes non définies
    → classe abstraite
- **Classe marquée abstraite sans méthodes abstraites**
  - empêche l'instanciation

UTILISATION

- **Jeu Monopoly**
  - classe Propriété
  sous classes Terrain, Gare, E&E

  - rendre la classe Propriété non instanciable

  - rendre la méthode CalculerLoyer abstraite

UTILISATION

- **Classe Animal**

```
public abstract class Animal {
    abstract int mange();
    abstract void respire();
}
```

- non instanciable (animal ??)
- tout animal (non abstrait) devra fournir les deux méthodes mange() et respire()

UTILISATION

- **Classe Figure non abstraite**
  sans méthode dessine()

- **Sous classes Cercle, Rectangle**
  avec méthode dessine

```
Figure f;
…
f.dessine();          // erreur.
```

- **Solution** :
  méthode dessine abstraite dans Figure

INTERFACE

- **Déclare des méthodes sans fournir de corps**
  contrat à respecter

- **Une classe implémente une interface**
  définit le corps des méthodes du contrat
  S'il reste des méthodes non définies
  → classe abstraite *(abstract à spécifier)*

CARACTERISTIQUES

- **Toutes les méthodes sont publiques et abstraites**
- **Non instanciable**
- **Pas de champs**
  uniquement constantes

CARACTERISTIQUES

- **Toutes les méthodes sont public et abstraites**
- **Non instanciable**
- **Pas de champs**
  uniquement constantes
- **Dérivable**
  ajout de méthodes

UTILISATION

- **Utilisation d'Interfaces de la bibliothèque**
  IEnumerable, IComparable (C#)

- **Programmation d'interfaces**
  IDeplacable, Icoloriable

UTILISATION

- **Méthodes fonctionnant pour plusieurs classes** (hors héritage)

  → interface = services communs de ces classes

- **Héritage multiple**

  une classe hérite d'une seule classe, mais peut implémenter plusieurs interfaces