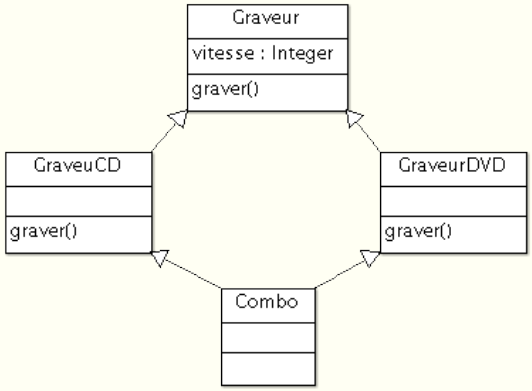
Les interfaces

Les Interfaces

- C'est un modèle d'implémentation.
  - *Tous les objets qui implémentent une interface possèdent les méthodes déclarées dans celle-ci.*
  - *On utilise souvent les interfaces dans les types de paramètres.*
    - En effet, dans certains cas, peu importe la classe d'un objet donné, car cela ne nous intéresse pas.
    - Nous avons seulement besoin de savoir que la classe se comporte de telle ou telle manière.
- Une interface est une classe abstraite définie par le mot clé interface. (abstract est implicite)
  - *Toutes les méthodes d'une interface sont abstraites.*
  - *Les seules attributs autorisés sont des constantes de classe. (static et final implicites)*

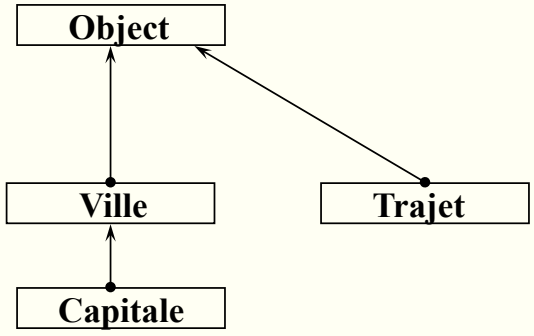
Les Interfaces

- Héritage Simple ou Multiple ?
  - *Une classe Fille ne peut hériter que d'une classe mère*
    - => Simplicité, pas «d' héritage en diamant»
  - *Une classe peut **implémenter** plusieurs interfaces (héritage de comportement)*
- Une interface permet d'éviter l'héritage multiple

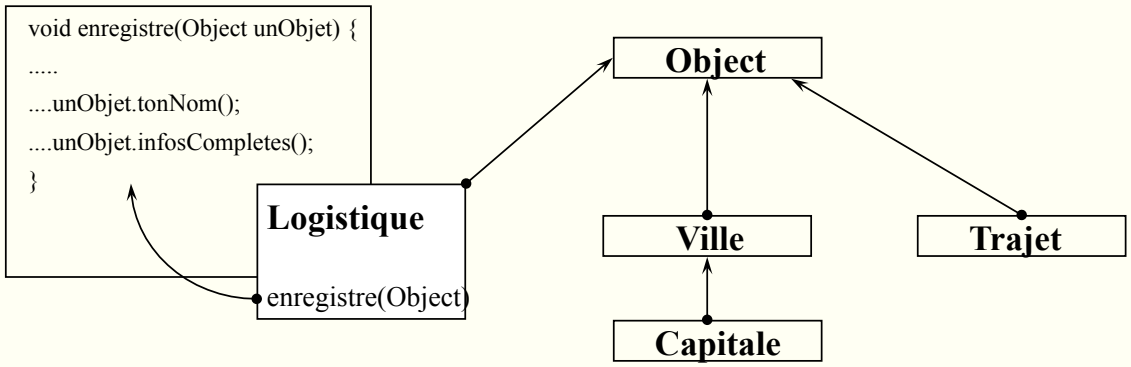


Les Interfaces : Un exemple

- On dispose d'une classe Ville, d'une classe Capitale héritant de Ville, et d'une classe Trajet.
- On dispose d'une classe Logistique avec une méthode enregistre (Object unObjet)
- On souhaite utiliser cette méthode pour stocker les informations relatives aux Villes, Capitales et Trajets

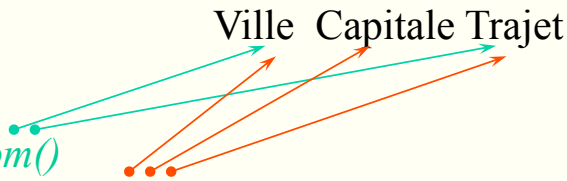


Les Interfaces : Un exemple



On souhaite utiliser la méthode enregistre pour stocker les informations des villes, capitales et trajets.

Les Interfaces : Un exemple



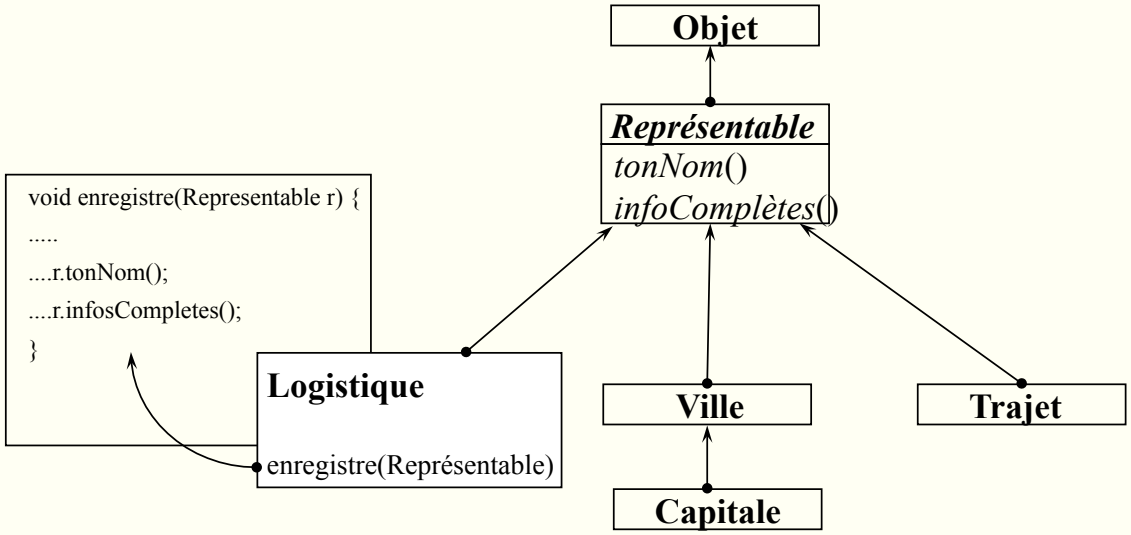
- Solution :
  - *2 implémentations de tonNom()*
  - *3 implémentations de infoComplètes()*
- Problème : Comment le compilateur peut comprendre : ***unObjet.tonNom()*** ?????
  - ***Solution*** *ajouter les 2 méthodes à la classe* Object

Remarque :

Ca va pas la tête, c'est interdit

Les Interfaces : Un exemple

- Nouvelle Solution :
  - *Ajouter une **classe abstraite** à la hiérarchie entre Object et ces classes avec les méthodes tonNom() et infoCompletes()*



Les Interfaces : Un exemple

- Notre application utilise également une classe *Chauffeur* héritant de *Employé*.
- Nous voulons également enregistrer les informations des *Chauffeurs* en utilisant la méthode *enregistre*.
  - *Il faudrait que* Chauffeur *hérite de* Representable *(**ce qui est interdit**, pas d'héritage multiple)*

Les Interfaces : Un exemple

- Solution : Transformer notre classe abstraite Representable en Interface.

```
public interface Representable {
   String tonNom();
   String infoCompletes();
}
```

- L'implémenter dans les classes Ville, Trajet, Chauffeur

*Class Chauffeur **extends** Employe* **implements** Representable {....}

Les Interfaces : Un exemple

- L'implémentation d'une interface est une forme d'héritage,
- On peut dire que Ville, Capitale, Trajet et Chauffeur heritent de l'interface Representable.
  - *On peut dire Ville est une Representable…*
- Comme pour l'héritage de classe,
  - *L'héritage d'interface définit un cast implicite de la classe fille vers l'interface mère.*
- Les interfaces permettent de définir des types.
  - *Représentable est un type.*
  - *On peut manipuler les objets du type Représentable via leurs comportements (tonNom et infosComplètes) sans ce soucier de leur classe réelle : Vive le polymorphisme !!!*

Autre exemple

```
interface Conduisible
{
  void demarrerMoteur();
  void couperMoteur();
  void tourner(float angle);
}

class Voiture implements Conduisible
{
  // ...
  void demarrerMoteur(); {...}
  void couperMoteur() {...}
  void tourner(float angle){...}
 }

class TondeuseGazon implements Conduisible
{
  // ...
  void demarrerMoteur(); {...}
  void couperMoteur() {...}
  void tourner(float angle){...}
}
```


Autre exemple

```
// ...

Voiture maVoiture = new Voiture();
TondeuseGazon maTondeuse = new TondeuseGazon();
Conduisible vehicule;
Boolean weekEnd;

// ...

if(weekEnd == true)
    {
       vehicule = maTondeuse;
    }
else
    {
       vehicule = maVoiture;
    }
vehicule.demarrerMoteur();
vehicule.tourner(90.0F);
vehicule.couperMoteur();

// ...
```

Pour conclure

- Une interface correspond à une classe abstraite où toutes les méthodes sont abstraites
  - *on ne peut pas instancier une interface (pas de new)*
- Une classe peut implémenter (implements) une ou plusieurs interfaces tout en héritant (extends) d'une seule classe
  - *elle s'engage alors à fournir une implémentation pour toutes les méthodes définies dans l'(les)interface(s)*

```
public maClasse extends MaClasseMere implements Interface1, Interface2{
…
```


Pour conclure

- Une interface peut hériter (extends) d'une autre interface
  - *les interfaces forment une hiérarchie séparée de celle des classes : hiérarchie de comportements*
- Une interface ne peut contenir que des attributs constants (static final) et des méthodes publiques non implémentées.

```
interface OlympicMedal {
  static final String GOLD = "Gold";
  static final String SILVER = "Silver";
  static final String BRONZE = "Bronze";
}
```