Modélisation et Programmation par Objets
HLIN406

Marianne Huchard, Clémentine Nebut

LIRMM / Université de Montpellier

2017

Sommaire

Langages et paradigmes de programmation

- paradigme de programmation : une approche pour définir, organiser, utiliser la connaissance.
- Principaux paradigmes
  - Programmation fonctionnelle, centrée sur une décomposition en fonctions (scheme, camel)
  - Programmation impérative, centrée sur une structuration “ séquentielle “ des traitements (pascal, C)
  - Programmation objet, centrée sur les données associées à leurs comportements (C++, Java)

Le raisonnement classificatoire

- Identification d'objets de base (Estelle, la voiture d'Estelle)
- Utilisation de ces objets comme des prototypes (la voiture d'Estelle vue comme une voiture caractéristique, à laquelle ressemblent les autres voitures, moyennant quelques modifications)
  ▷ modèles et langages à prototypes
- Regroupement des objets partageant des propriétés structurelles et comportementales en classes
  ▷ modèles et langages à classes

Classes

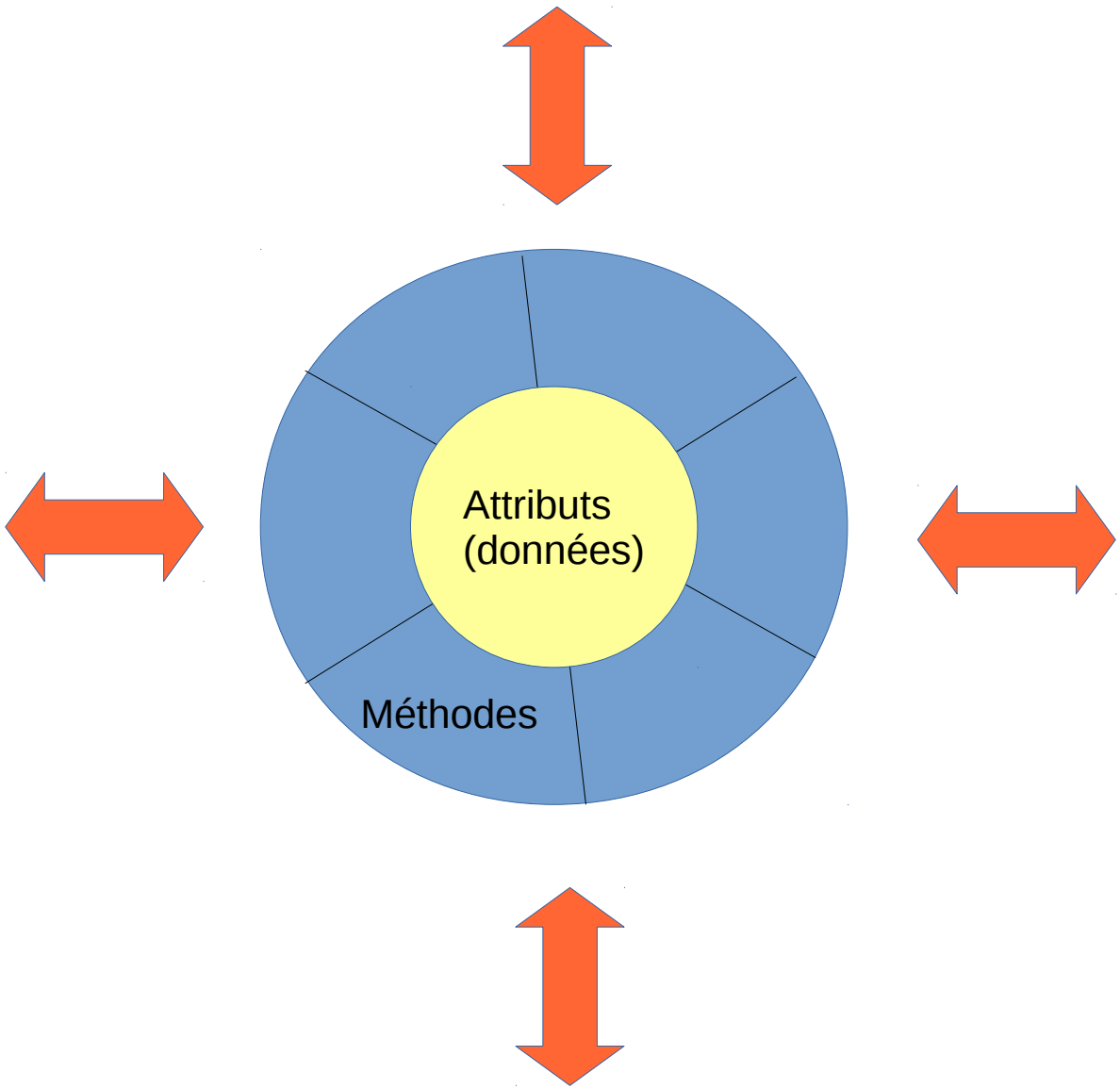
Concept du domaine sur lequel porte :

- le logiciel ▷ voiture ou compte bancaire
- le domaine du logiciel ▷ un type de données abstrait tel que la pile

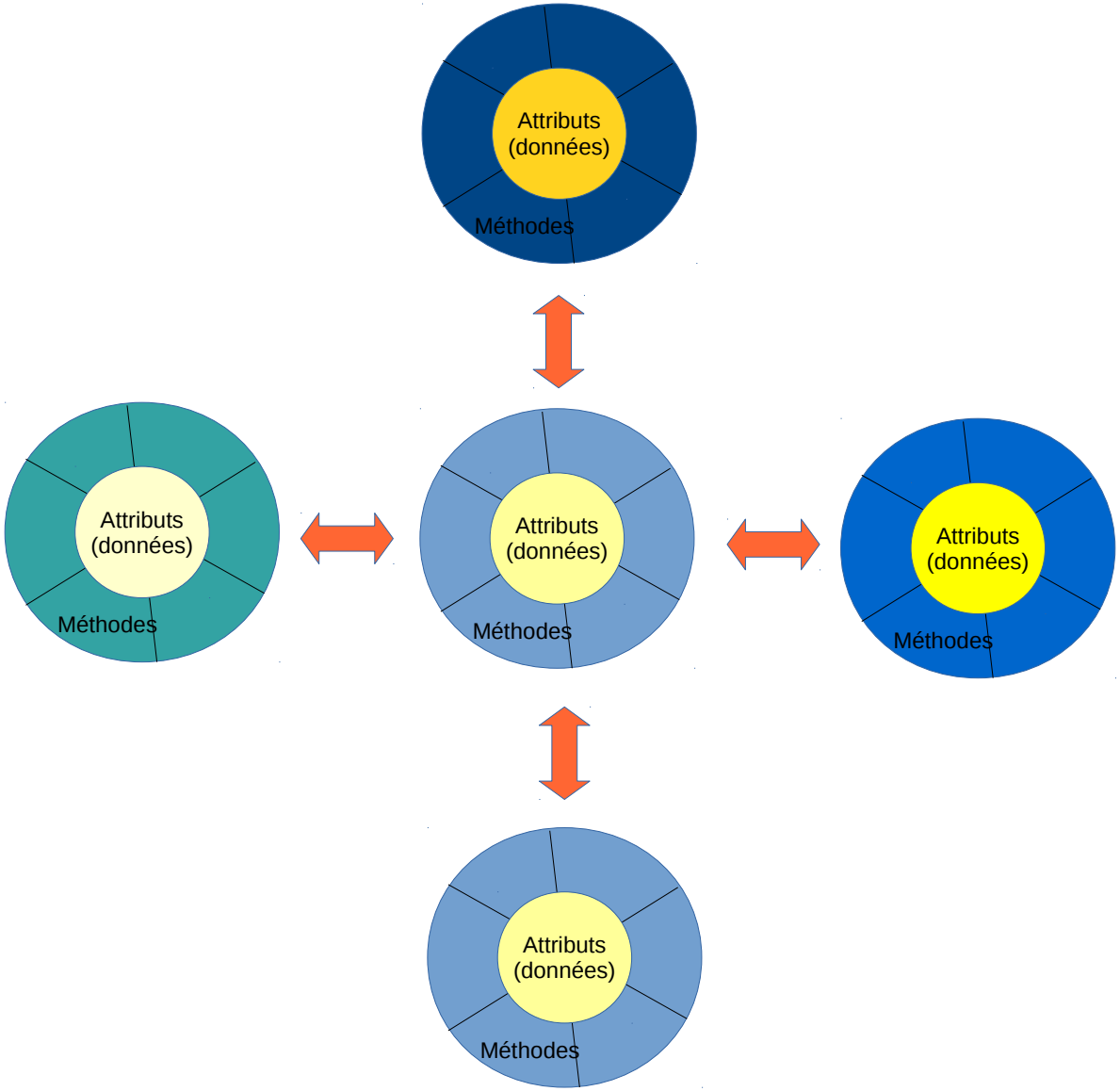
Une classe peut se voir selon trois points de vue :

- un aspect *extensionnel* : l'ensemble des objets (ou instances) représentés par la classe,
- un aspect *intensionnel* : la description commune à tous les objets de la classe, incluant les données (partie statique ou attributs) et les opérations (partie dynamique),
- un aspect *génération* : la classe sert à engendrer les objets.

Encapsulation des données



Encapsulation des données et communication entre objets



Sommaire

Les classes et instances en UML

Modèle statique (ou structurel)

- Diagrammes d'objets ou diagrammes d'instances
  - objets du domaine modélisé ou de la solution informatique (par exemple des personnes, des comptes bancaires)
  - liens entre ces objets (par exemple le fait qu'une personne possède un compte bancaire) ;
- Diagrammes de classes
  - abstraction des diagrammes d'objets
  - classes, associations, attributs, opérations

Représentation des classes

Pour "ranger" les classes : les paquetages

- regroupement logique d'éléments UML, par exemple de classes
- structuration d'une application
- utilisés dans certains langages, notamment Java, ce qui assure une bonne traçabilité de l'analyse à l'implémentation
- liés par des relations de dépendance

Les paquetages en UML

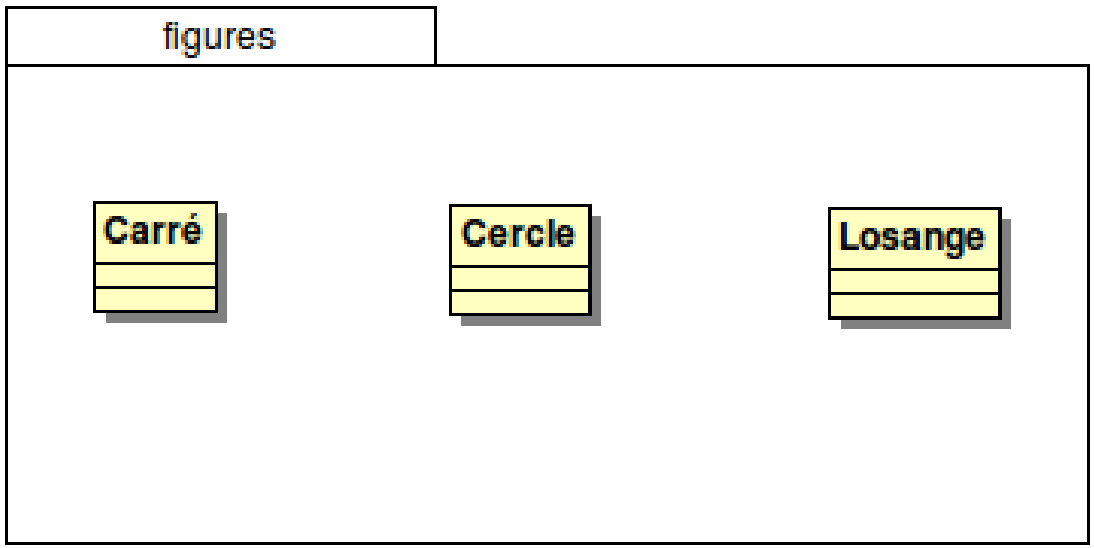


Figure – Paquetages en UML

Instances et classes en notation UML

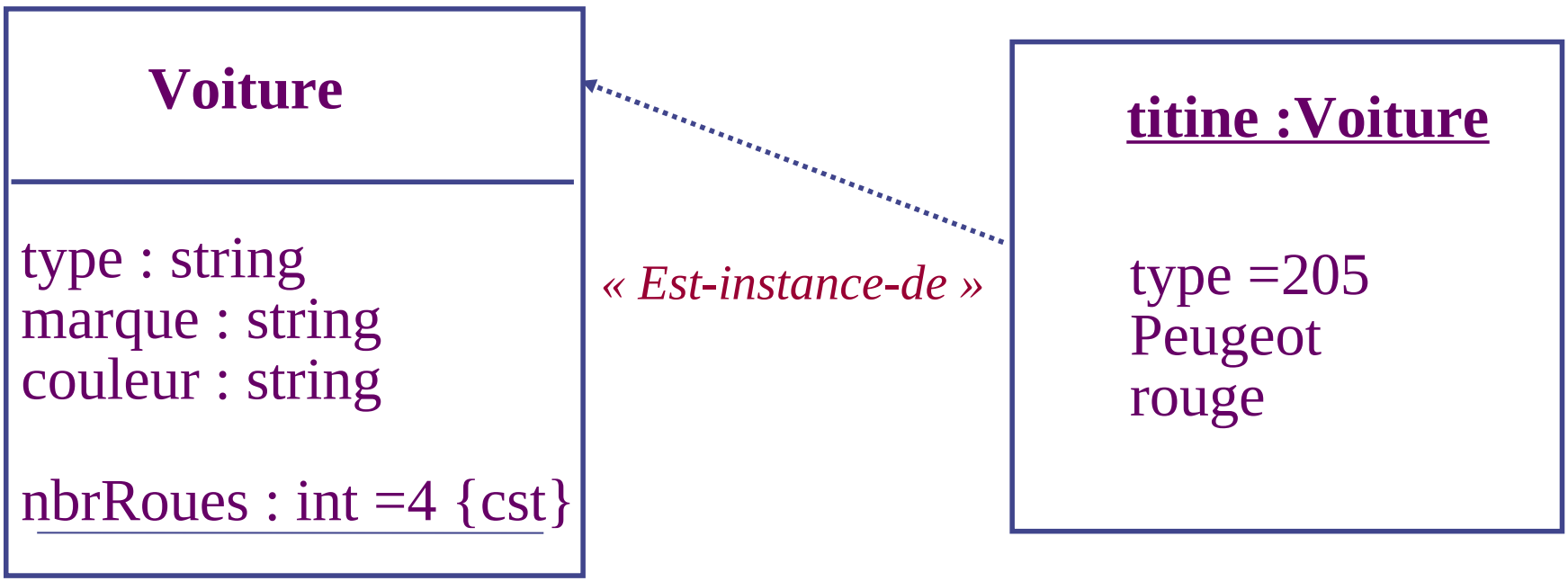


Figure – Classe (à gauche) et objet/instance (à droite)

Attributs

Quelles informations pour décrire un attribut ? Principalement :

- son nom
- son type
- sa visibilité (qui peut le voir/l'utiliser ?)
- la multiplicité (combien de valeurs sont portées simultanément par l'attribut ?)

Description des attributs

`[visibilité][/]nom[:type][[multiplicité]][= valeurParDéfaut]`

- `visibilité` $\in \{+, -, \#, \sim\}$, et `multiplicité` définit une valeur (1, 2, n, ...) ou une plage de valeurs (1..*, 1..6, ...).
- visibilité : d'où est visible l'attribut ?
  - Public. + est la marque d'un attribut accessible partout (public)
  - Privé. - est la marque d'un attribut accessible uniquement par sa propre classe (privé)
  - Package. $\sim$ est la marque d'un attribut accessible par tout le paquetage
  - Protected. # est la marque d'un attribut accessible par les sous-classes de la classe
- le nom est la seule partie obligatoire de la description

Description des attributs

`[visibilité][/]nom[:type][[multiplicité]][= valeurParDéfaut]`

- la multiplicité décrit le nombre de valeurs que peut prendre l'attribut (à un même moment)
- le type décrit le domaine de valeurs
- la valeur initiale décrit la valeur que possède l'attribut à l'origine
- des propriétés peuvent préciser si l'attribut est constant (`{constant}`), si on peut seulement ajouter des valeurs dans le cas où il est multi-valué (`{addOnly}`), etc.

Description des attributs

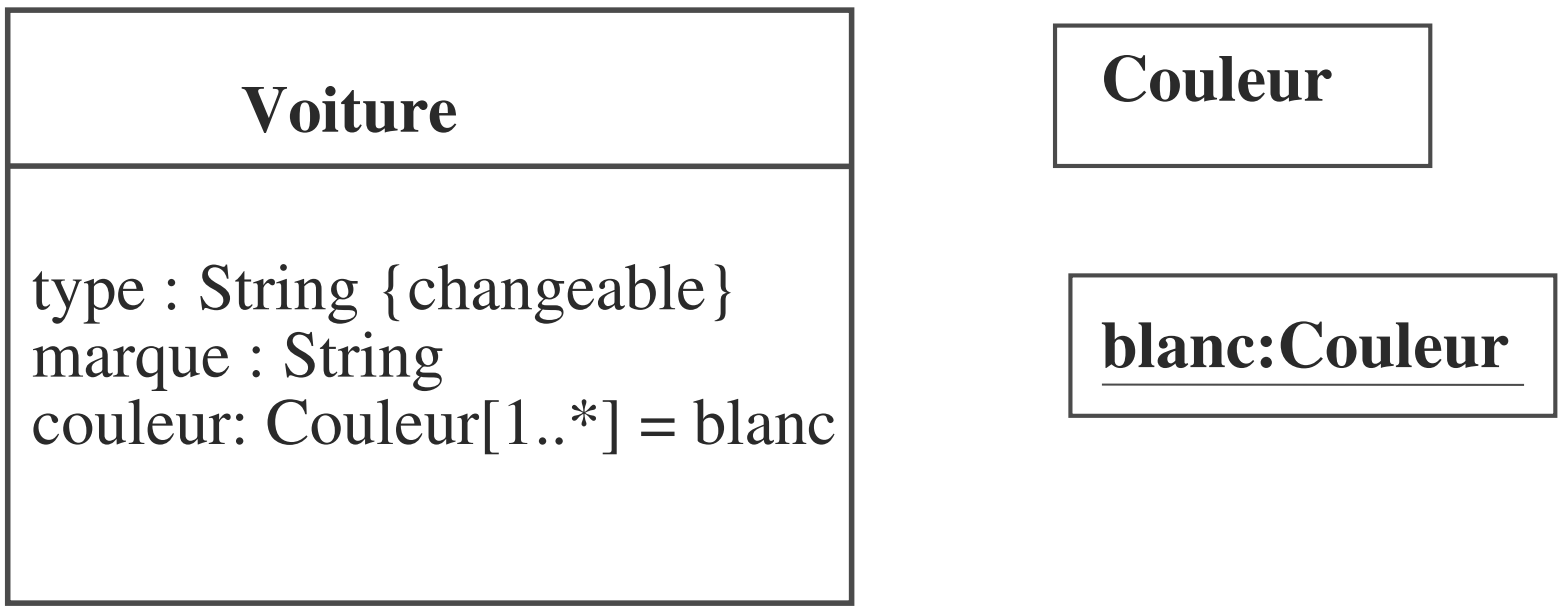


Figure – Détails sur la syntaxe de description des attributs

Sommaire

Types de base en Java

- `boolean`, constitué des deux valeurs `true` et `false`. Les opérateurs se notent :

Java	non	égal	différent	et alors / et	ou sinon / ou
	!	==	!=	&&	||
				&	|

- `int`, entiers entre -2^{31} et $2^{31}-1$
- `float`, `double`, ces derniers sont des réels entre $-1.79E+308$ et $+1.79E+308$ avec 15 chiffres significatifs.
- `char`, caractères représentés dans le système Unicode. Les constantes se notent entre apostrophes simples, par exemple `'A'`.
- `String`, qui est ... une classe. Les chaînes de caractères constantes se notent entre guillemets, par exemple `"hello world"`.

Écriture des classes

```
package exemplesCours1;
public class Voiture
{
private String type; // null
private String marque; // null
private String couleur; // null
private static int nbrVoitures; // 0
private static final int NBR_ROUES = 4;
}
```

Conventions :

- le nom de la classe commence par une majuscule,
- le nom du package commence par une minuscule,
- le nom des attributs commence par une minuscule,
- les constantes sont en majuscules.

Création des instances

Déclaration d'une variable

```
Voiture titine;
```

Création d'une instance

```
titine = new Voiture();
```

Accès aux attributs, affectations

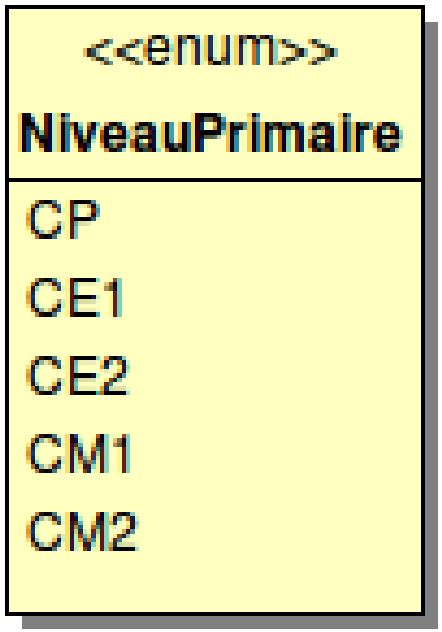
Notation '.' Exemple `titine.type`
Ecriture de valeurs dans des attributs d'instance

```
titine.type = "205";
titine.marque = "Peugeot";
titine.couleur = "rouge";
```

Sommaire

Les énumérations

- Un type de données dont on peut énumérer toutes les valeurs possibles
- Exemples :
  - Civilité → Mme, M, Mlle
  - Stations de ski du grand domaine → Valmorel, Combelouvière, Saint-François-Longchamp
  - Niveaux à l'école primaire → CP, CE1, CE2, CM1, CM2



Petit aperçu des énumérations en Java

```
public enum Niveau{
CP, CE1, CE2, CM1, CM2;
}

...

Niveau n=Niveau.CP;
...
```

Attributs dérivés

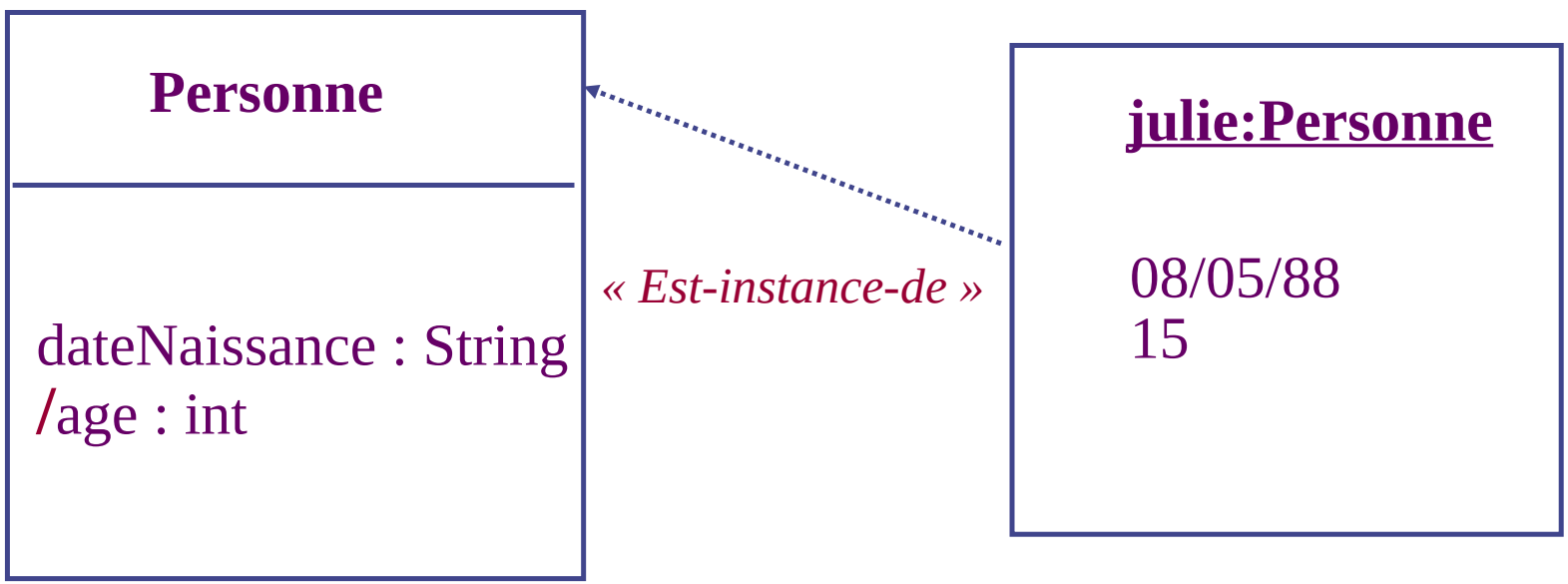
Valeur déduite de la valeur d'autres attributs ou d'autres éléments décrivant l'objet

Classe

Attributs dérivés

Instance

Valeurs d'attributs (État)



{age = date du jour – date de naissance}

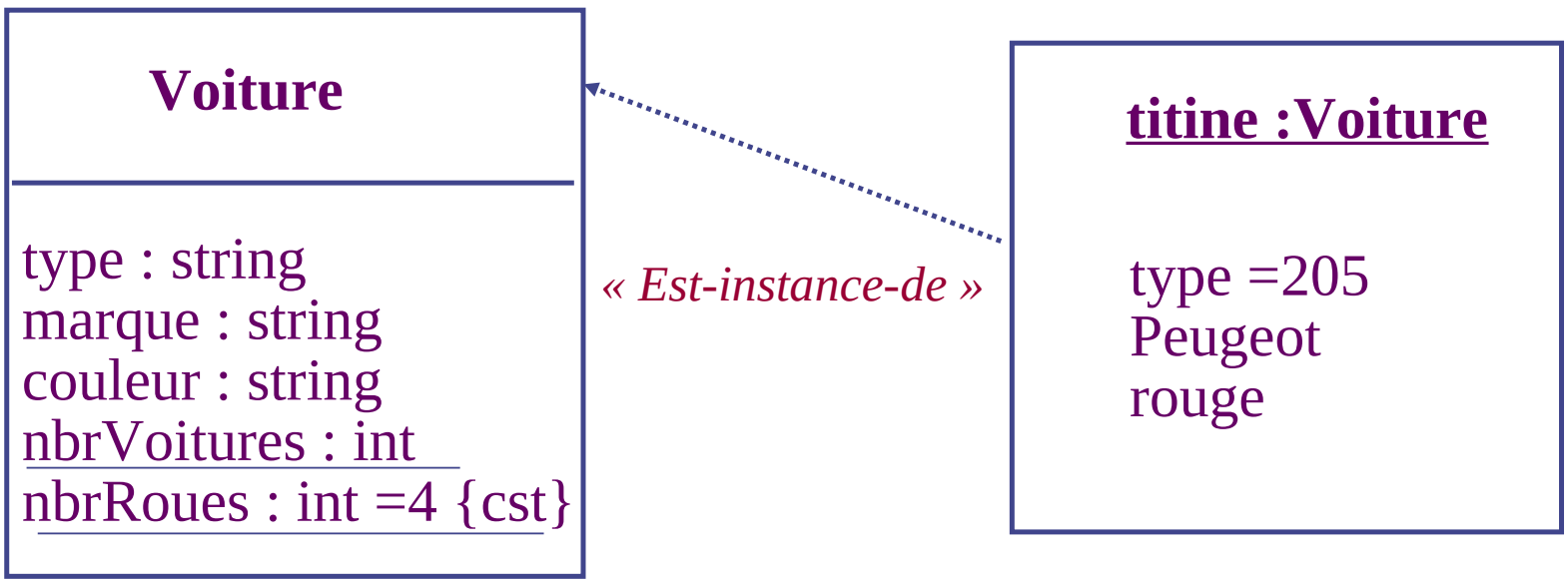
Peut-être une opération déguisée ? (stockage optionnel)

Attributs *de classe*

Valeur partagée par toutes les instances

Classe
Attributs de classe

Instance
Valeurs d'attributs (État)
Sauf des attributs de classe



Attribut de classe ~ caractéristique partagée
Révèle souvent une modélisation à approfondir

Figure – Attributs de classe

Accès aux attributs de classe

Notation : Nom de la classe ou de l'instance '.' nom de l'attribut

```
Voiture.nbrVoitures = 3;
Voiture titine=new Voiture();
titine.nbrVoitures=4;
```

Sommaire

Prochaine étape

- Animer les objets
- Opérations décrivant la manière dont les objets répondent aux messages

Mais avant ...

On s'intéresse aux puzzles 2D. Leur dimension se définit par une longueur et une largeur donnée généralement en centimètres. Ils ont un certain nombre de pièces (20, 50, 100, 1000, etc.) et en général un nom. Ils sont des reproductions d'images (photos, peintures, ou dessins). On trouve des puzzles en bois ou en carton.

- Réalisez un diagramme de classes pour les puzzles.
- Réalisez un diagramme d'objets pour un puzzle en carton, nommé "Femmes au marché", de 98x75cm, étant une reproduction de peinture, et ayant 2000 pièces.

Le diagramme de classe