Design Patterns – Value Object & Entity

CHEBAH Megdouda / COUDERT Alexandre / BARRAUD Maxime

Sommaire

- Rappel sur les Patterns
- Entité, Objet Valeur ? De quoi s'agit-il?
  - Entité – Définition
  - Valeur Objet – Définition
  - Relations
- Comment cela fonctionne ?
  - Objet Valeur
  - Entité
- Conclusion
  - Quand utiliser ce design pattern ?
  - Une autre approche au code ?

Entité – Définition

- Un ensemble de bonnes pratiques d'implémentation de code afin de rendre le code propre, maintenable, solide et évolutif.

- Il existe 3 principaux types de design pattern :les patterns de création, les patterns de structuration et les patterns comportementaux.

- Tout DP à : Un nom, problème, solution et conséquence.

Entité – Définition

- Objet ayant un concept d'identité (donc il y aura un système d'identifiant)

  → On peut donc penser à une approche comme les tables en base de données.

- Mutable, donc modifiable
- Durée de vie plus longue qu'un Objet Valeur

Objet Valeur – Définition

- Objet contenant simplement, comme le nom l'indique, une ou plusieurs valeurs.
- Durée de vie courte de par leur fonctionnement
- Ex : Une Adresse contenant un numéro, un nom de rue, une ville.
- Les objets valeurs sont immutables (pas de changement!)

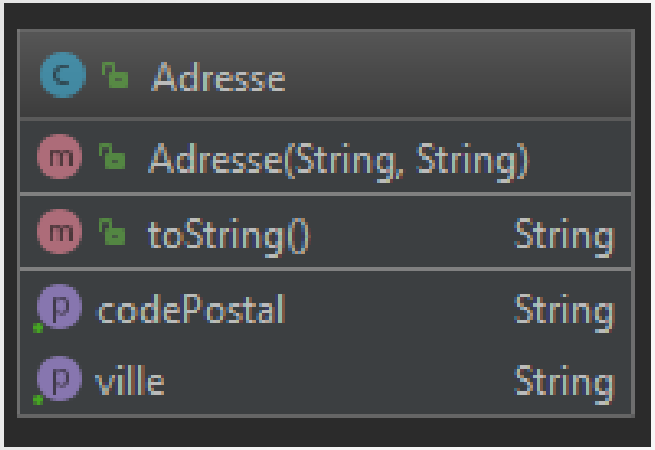
Relations

- Les objets valeurs sont souvent utilisés par les entités

  → Les entités sont donc composés d'objets valeurs, mais également de leurs identifiants !

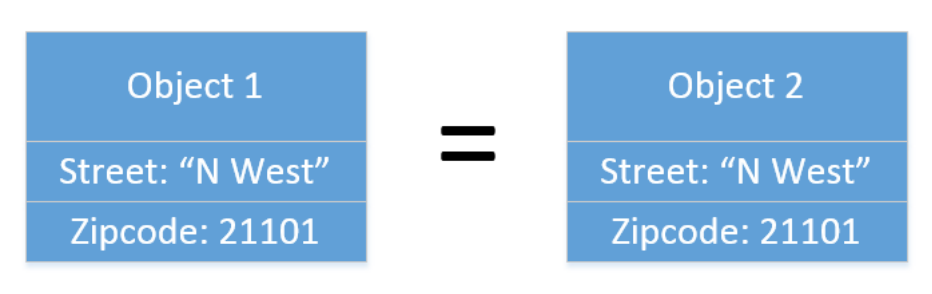
- Si on doit remplacer un objet valeur dans une Entité, on va générer une nouvel objet valeur, on ne le modifie pas !

Comment cela fonctionne ? - Objet Valeur

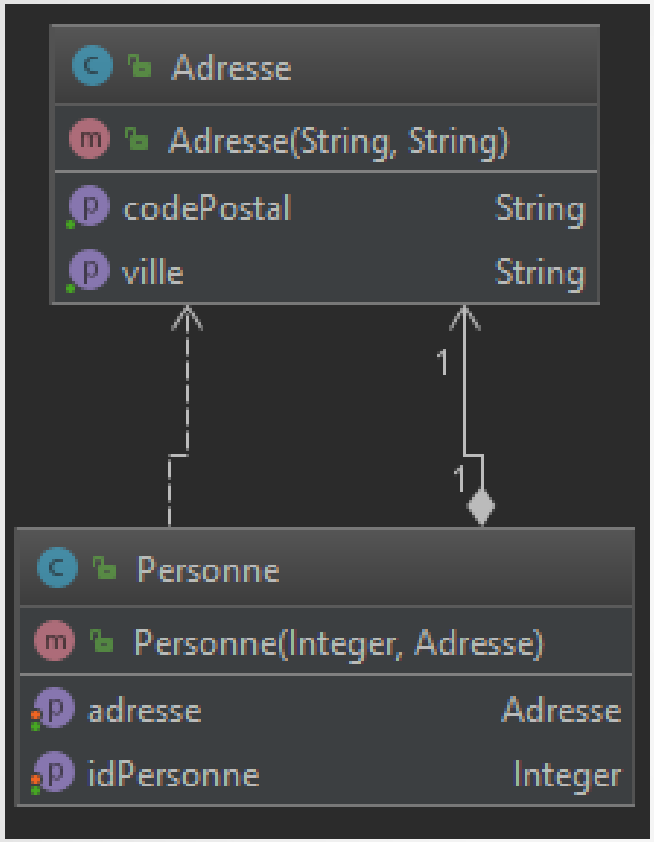



```java
public static void main(String[] args) {
    Adresse adresse = new Adresse( ville: "Limoges",  codePostal: "87100");
    System.out.println(adresse);
    //adresse.setCodePostal("87000"); -> KO: Objet Valeur, donc immutable !
    adresse = new Adresse( ville: "Nantes",  codePostal: "44000"); //OK
    System.out.println(adresse);
}
```

```
Adresse [ville=Limoges; codePostal=87100]
Adresse [ville=Nantes; codePostal=44000]
```



Comment cela fonctionne ? - Entité




```java
Adresse limoges = new Adresse( ville: "Limoges", codePostal: "87100");
Adresse nantes = new Adresse( ville: "Nantes", codePostal: "44000");
Adresse strasbourg = new Adresse( ville: "Strasbourg", codePostal: "67000");
Personne personneDeLimoges = new Personne( idPersonne: 0, limoges);
Personne personneDeNantes = new Personne( idPersonne: 1, nantes);
Personne personneDeStrasboug = new Personne( idPersonne: 1, strasbourg);
System.out.println(personneDeLimoges.equals(personneDeNantes));          // -> false
System.out.println(personneDeLimoges.equals(personneDeStrasboug));       // -> true
```

Object 1		Object 2
Id: 1	=	Id: 1
Name: “object”		Name: “obj”

Une autre approche au code ?

- Spécificité du design pattern : le DDD
- DDD : Domain Driven Design
- “Domain Driven Design, Tackling Complexity in the Heart of Software”, par Eric Evans
- Permet de mieux appréhender la complexité d’un projet en partageant un langage commun

Sources

- https://enterprisecraftsmanship.com/2016/01/11/entity-vs-value-object-the-ultimate-list-of-differences/
- http://www.informatix.fr/articles/conception/valueobject-qu-est-ce-que-c-est-192
- http://blog.xebia.fr/2013/03/15/le-value-object-et-lentity-du-ddd/
- https://deviq.com/value-object/
- https://stackoverflow.com/questions/75446/value-vs-entity-objects-domain-driven-design
- https://hackernoon.com/is-it-a-value-object-or-an-entity-2d3e29d5367f
- https://martinfowler.com/bliki/ValueObject.html
- http://seedstack.org/guides/ddd-for-beginners/entities-and-value-objects/