



ENCART CAMÉRA

CC BY NC SA

**Comparaison d'objets :
Unicité, Équivalence & Hachage**

Sébastien Mosser - INF5153
Chapitre 3 - Capsule 2
Automne 2020

UQÀM | Département d'informatique

ace

© 2020 UQÀM

Dans le chapitre précédent

ENCART CAMÉRA

CC BY NC SA

- On s'est intéressé à la comparaison des objets
 - Avec l'interface **Comparable**, ou encore les objets **Comparators**
 - Les constructions de Java reposent sur les codes des développeurs
 - *Prétexte pour étudier la réalisation et la ségrégation d'interface*
- Mais **comparer**, c'est **compliqué** !
 - Il est important de se poser ces questions quand on conçoit le logiciel

UQÀM | Département d'informatique

Unicité des objets : De l'équivalence à l'égalité

ENCART CAMÉRA

CC BY NC SA

- **Comparable** (et **Comparator**) permette de représenter :
 - un ordre (résultat $\neq 0$),
 - mais aussi une relation d'équivalence (résultat = 0).
 - Les cartes de même valeur mais de couleurs différentes sont considérées comme équivalentes dans Schotten Totten
- Implémenter **equals** permet de définir une relation d'égalité.

UQÀM | Département d'informatique

Unicité des objets : De l'équivalence à l'égalité

ENCART CAMÉRA



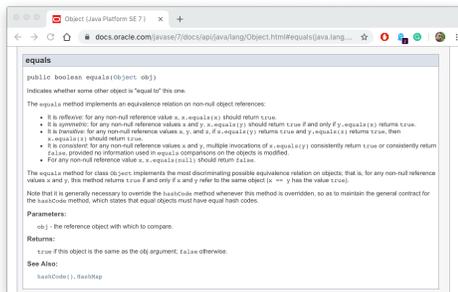
- Implémenter *equals* permet de définir une relation d'égalité.
 - qui est *symétrique*, *réflexive* et *transitive* elle aussi !
- L'égalité est une relation d'équivalence parmi d'autres
 - C'est la seule dont le **quotient** (l'ensemble des classes d'équivalence) **contient uniquement des singletons**.
 - Autrement dit, **pour tout élément e, le seul élément équivalent à e par la relation d'égalité est e lui-même**.

Encore un truc de prof qui sert à rien

ENCART CAMÉRA



C'est dans la documentation du langage !



Implémentation de l'égalité Schotten Totten

ENCART CAMÉRA



```
public class Card implements Comparable<Card> {  
  
    @Override  
    public int compareTo(Card that) {  
        return this.value.compareTo(that.value);  
    }  
  
    @Override  
    public boolean equals(Object o) {  
        if (this == o) return true;  
        if (!(o instanceof Card)) return false;  
        Card card = (Card) o;  
        if (color != card.color) return false;  
        return value == card.value;  
    }  
}
```

Attention à la signature

Référence Identique

Null, Classes ≠

Égalité Structurelle

Relation d'égalité & fonction de Hachage

ENCART CAMÉRA



- Le "hashCode" d'un objet correspond à une "empreinte"
 - Deux objets égaux doivent avoir le même hashCode
 - Deux objets \neq peuvent avoir le même hashCode
 - C'est une **collision**, il y a un impact de performances.
- Les structures de données à base de hachage reposent dessus
 - P.-ex. HashMap, HashSet, ...
- En java, si on redéfinit equals(), alors on redéfinit hashCode().
 - Si non les Collections risquent de faire n'importe quoi.

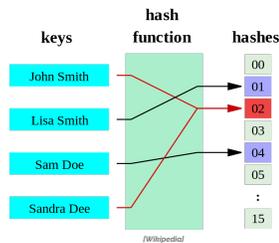
Principe du Hachage d'un objet

ENCART CAMÉRA



```
public class Player {
    private String name;

    public int hashCode() {
        return ...;
    }
}
```



Pour chercher dans une HashMap, on regarde le hash, et en cas de collision on vérifie l'égalité un par un.

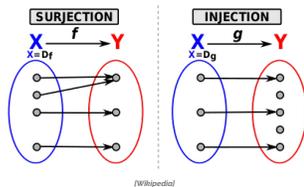
Injektivité & Surjectivité du Hachage

ENCART CAMÉRA



- Un hachage **surjectif** produit des **collisions**
 - Il faut vérifier avec equals si les objets sont les mêmes
 - $\mathcal{O}(|X|)$ dans le pire des cas


```
(int hashCode() { return 42; })
```
- Un hachage **injectif** est dit "parfait"
 - C'est très rare en réalité, p.-ex. $|CardPile| \gg |Int|$



[Wikipedia]

Le génie logiciel, une science de paresseux

ENCART CAMÉRA



```
public class Card {  
    @Override  
    public int hashCode() {  
        int result = color != null ? color.hashCode() : 0;  
        result = 31 * result + (value != null ? value.hashCode() : 0);  
        return result;  
    }  
}
```

Template IntelliJ

Approche par Annotation
et méta-programmation
(ici avec Lombok)

```
@EqualsAndHashCode  
public class Card {  
    private Color color;  
    private Value value;  
    @EqualsAndHashCode.Exclude  
    private String uselessField  
}
```

Il faut comprendre le principe
pour utiliser correctement l'outil

Il faut comprendre les
principes avant d'utiliser
ce genre de mécanismes

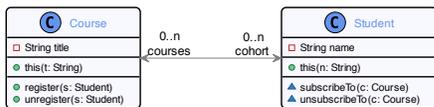
ENCART CAMÉRA



Un exemple d'erreur classique



ENCART CAMÉRA



```
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass())  
        return false;  
    Course course = (Course) o;  
    return Objects.equals(title, course.title) &&  
        Objects.equals(cohort, course.cohort);  
}
```

```
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass())  
        return false;  
    Student student = (Student) o;  
    return Objects.equals(name, student.name) &&  
        Objects.equals(courses, student.courses);  
}
```

