# Programação Orientada por Objetos

Método Equals

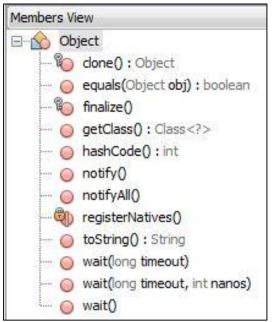
(Livro Big Java, Late Objects – Capítulo 9)

# Sumário

- <u>Introdução</u>
- Interesse
- Tipos de Igualdade
- Definição
- Reescrita
  - Numa Classe
  - Numa Subclasse

# Introdução

- Método de Instância
  - Comum a todas as classes do Java
    - Membro da classe Object



### **Implementação**



```
public boolean equals(Object obj) {
   return (this == obj);
}
```

#### Verifica:

- Igualdade de referências de dois objetos:
  - this
  - obj // outro objeto qualquer

#### Retorna:

- true // referência this = referência obj
- false

### Interesse

- Utilizar Código Java
  - Exemplo
    - Classe ArrayList:

```
ArrayList<E>:: AbstractList<E>: List<E>, RandomAccess, Cloneable, Serializabl ▲

    ArrayList(int initialCapacity)

ArrayList()

    ArrayList(Collection <? extends E > c)

add(E e): boolean ↑ AbstractList<E>
add(int index, E element) ↑ AbstractList<E>
addAll(Collection <? extends E > c): boolean ↑ AbstractCollection <E>
addAll(int index, Collection <? extends E> c): boolean ↑ AbstractList <E>
batchRemove(Collection <?> c, boolean complement): boolean

    dear() ↑ AbstractList<E>

a done(): Object ↑ Object
○ contains(Object o): boolean ↑ AbstractCollection <E>
elementData(int index) : E
ensureCapacity(int minCapacity)

    ensureCapacityInternal(int minCapacity)

    ensureExplicitCapacity(int minCapacity)

fastRemove(int index)
forEach(Consumer <? super E> action)

    get(int index) : E ↑ AbstractList<E>

grow(int minCapacity)
hugeCapacity(int minCapacity): int
indexOf(Object o): int ↑ AbstractList<E>
    isEmpty(): boolean ↑ AbstractCollection <E>
iterator(): Iterator <E> ↑ AbstractList <E>

    lastIndexOf(Object o): int ↑ AbstractList<E>

☐ listIterator(int index) : ListIterator <E> ↑ AbstractList <E>
| listIterator(): ListIterator<E> ↑ AbstractList<E>
outOfBoundsMsg(int index) : String
rangeCheck(int index)
rangeCheckForAdd(int index)
readObject(ObjectInputStream s)
o remove(int index) : E ↑ AbstractList<E>

    remove(Object o): boolean ↑ AbstractCollection <E>
```

#### Método remove(Object o)

```
@param o element to be removed from this list, if
   @return <tt>true</tt> if this list contained the
public boolean remove(Object o) {
    if (o == null) {
        for (int index = 0; index < size; index++)</pre>
            if (elementData[index] == null) {
                 fastRemove(index);
                 return true:
    } else +
        for (int index = 0; index < size; index++)</pre>
            if (o.equals(elementData[index])) {
                 fastRemove(index);
                 return true;
    return false:
```

#### Método equals

Usado na procura do objeto o

# **Igualdade de Objetos**

- Tipos
  - Igualdade de referências
  - Igualdade de atributos (estado) // em value objects
- Igualdade de Referências
  - Implementado:
    - No equals da classe Object
  - Não requer:
    - Reescrita do equals
- Igualdade de Atributos
  - Requisito:
    - Reescrita do equals
  - Exemplo:

```
public class Ponto2D {
    private int x;
    private int y;
}
```

■ Ponto2D(x1,y1)=Ponto2D(x2,y2)  $\Rightarrow$  x1=x2 e y1=y2

# Definição

- public boolean equals(Object obj)
  - Indica:
    - Se outro objeto obj é igual ao objeto this. // this.equals(obj)
  - Implementa:
    - Relação de equivalência em objetos:
      - Propriedades matemáticas (quaisquer objetos x, y e z referências não-NULL):
        - Reflexiva: x=x
          - Um objeto é igual a si próprio ⇒ x.equals(x) =true
        - Simétrica: x=y ⇒ y=x
          - Se um objeto é igual a outro então o inverso também é verdade.
          - x.equals(y)=true sse y.equals(x)=true
        - Transitiva:  $x=y \land y=z \Rightarrow x=z$ 
          - Se x.equals(y)=true e y.equals(z)=true ⇒ x.equals(z)=true
    - Consistência: {x.equals(y)=T, ..., x.equals(y)=T} ou {x.equals(y)=F, ..., x.equals(y)=F}
      - Múltiplas invocações de x.equals(y):
        - Retornam consistentemente true (T) ou false (F), se o estado dos objetos comparados (x e y) não for modificado.
    - Nenhum objeto é igual a NULL: x.equals(null)=false
    - Nenhum objeto x é igual a outro objeto y de classe diferente: x.equals(y)=false

#### Exemplo

```
public class Ponto2D {
   private int x;
   private int y;
   public Ponto2D(int x, int y) {...}
    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        if (obj == null || this.getClass() != obj.getClass()) {
            return false;
        Ponto2D p = (Ponto2D) obj;
        return x==p.x && y==p.y;
```

#### Exemplo

```
public class Ponto3D extends Ponto2D {
    private int z;
    public Ponto3D(int x, int y, int z) {...}
    @Override
    public boolean equals(Object obj) {
        if (!super.equals(obj)) {
            return false;
        Ponto3D p = (Ponto3D) obj;
        return z == p.z;
```