Programação Orientada por Objetos

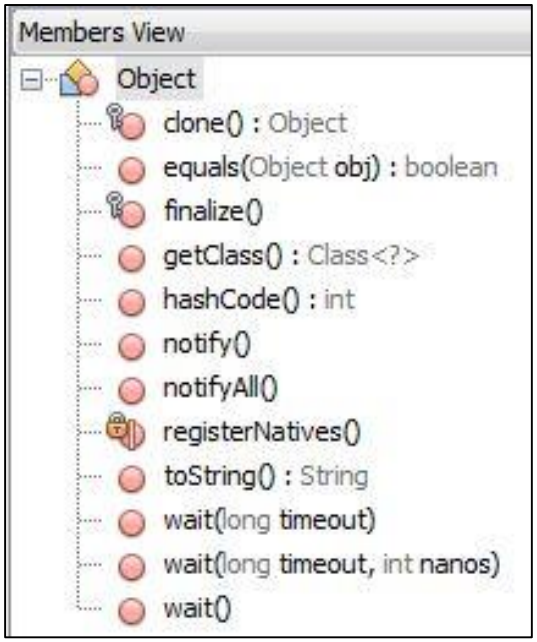
Método Equals

(Livro *Big Java, Late Objects* – Capítulo 9)

Sumário

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:

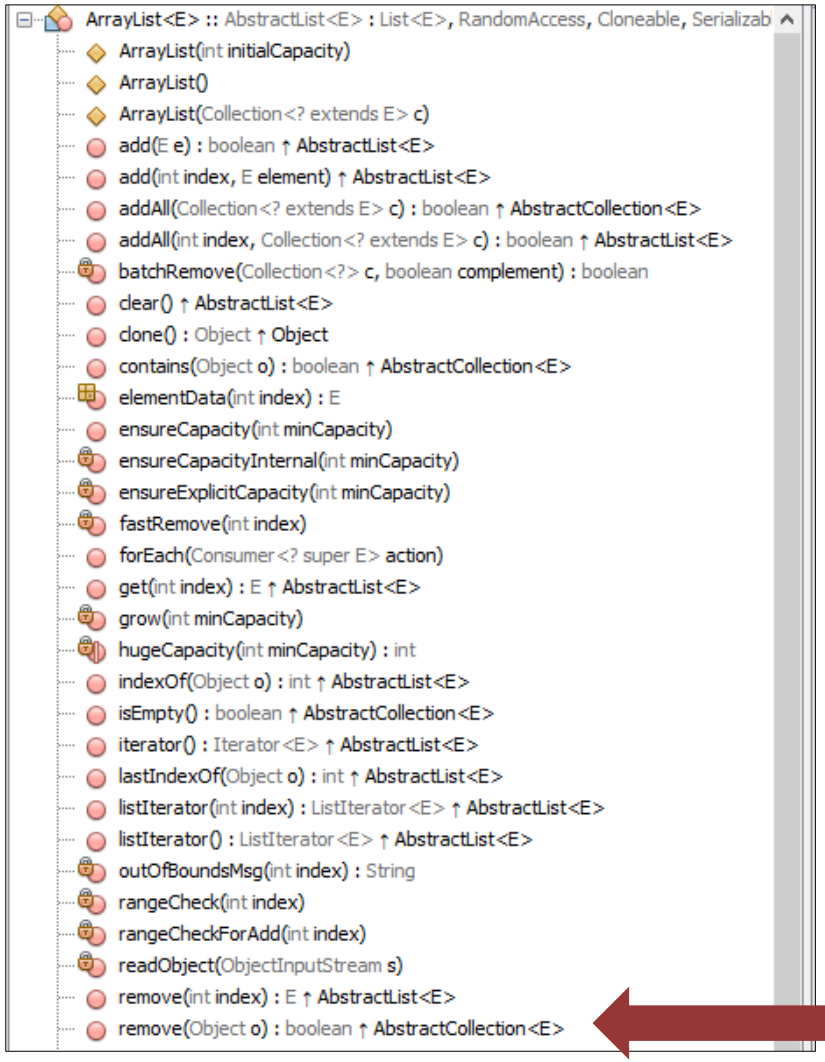
- Igualdade de referências de dois objetos:
  - this
  - obj // outro objeto qualquer

Retorna:

- true // referência this = referência obj
- false

Interesse

- Utilizar Código Java
  - Exemplo
    - Classe ArrayList:



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++)
                if (elementData[index] == null) {
                    fastRemove(index);
                    return true;
                }
        } else {
            for (int index = 0; index < size; index++)
                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 $\Rightarrow$ x.equals(x) =true
        - Simétrica: x=y $\Rightarrow$ 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 $\wedge$ y=z $\Rightarrow$ x=z
          - Se x.equals(y)=true e y.equals(z)=true $\Rightarrow$ 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

Reescrita numa Classe

- **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;
    }
}
```

Reescrita numa Subclasse

- 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;
    }
}
```