

# Destrutores em Java

Prof. Hugo de Paula



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS  
Departamento de Ciência da Computação

# Sumário

- 1 Alias e semântica de referência
- 2 Lixo de memória e referência Dangling
- 3 Coletores de lixo
- 4 Finalizando um objeto
  - Exemplo: Processa Arquivo
  - Exemplo: Classe Produto



# Alias e semântica de referência

- **Semântica de referência:** variáveis representam referências para objetos.
  - Objetos podem ser alocados e desalocados a qualquer tempo, enquanto as referências possuem um escopo e tempo de vida definidos.
- **Alias** ocorre quando o mesmo objeto é associado a dois nomes diferentes ao mesmo tempo (duas referências).



## Lixo de memória e referência *Dangling*

- **Lixo** é a memória que foi alocada no ambiente mas se torna inacessível ao programa.
  - Pode surgir quando um programador se esquece de desalocar uma variável dinâmica antes de alterar o estado do ponteiro que referencia esta região de memória.
- **Referência *dangling*:** ponteiro que aponta para uma área de memória que foi liberada.



# Coletores de lixo

- **Coletor de lixo** é um processo que automaticamente elimina o lixo, liberando a memória que não é mais utilizada.
  - Eliminam a necessidade de se desalocar memória explicitamente
  - Coletores de lixo eliminam o vazamento de memória.
  - Coletores de lixo eliminam referências *dangling*.
- **Java**
  - Não possui ponteiros explícitos (apenas semântica de referência).
  - Não possui operadores de desalocação de memória (*free* ou *delete*).
  - Possui coletor de lixo que faz a gestão da desalocação de memória automaticamente.



# Java Garbage Collector

- Várias variáveis podem apontar para um mesmo objeto.
- Um objeto é elegível para coleta de lixo quando:
  - não pode mais ser acessado por nenhuma referência;
  - referencia um outro objeto que também o referencia formando um ciclo único e isolado.



# Finalizando um objeto

- Pode ser necessário resolver pendências antes de um objeto ser removido.
- Quando um objeto vai ser removido pelo coletor de lixo, um método de finalização é executado.

## Método `finalize` da classe `Object`

```
protected void finalize() throws Throwable {  
    ...  
}
```



## Exemplo: Processa Arquivo

```
public class ProcessaArquivo {  
    private Stream arq;  
  
    public processaArquivo(String caminho) {  
        arq = new Stream(caminho);  
    }  
    ...  
    public void close() {  
        if (arq != null) { arq.close();  
                        arq = null; }  
    }  
    protected void finalize() throws Throwable {  
        super.finalize();  
        close();  
    }  
}
```





# Exemplo: Destrutor da classe Produto

```
class Produto {  
    private static int instancias = 0;  
  
    public static int getInstancias() {  
        return instancias;  
    }  
  
    public Produto(String d, float p, int q) {  
        instancias++;  
    }  
  
    public Produto() {  
        instancias++;  
    }  
  
    /**  
     * É executado quando um objeto está sendo removido da memória.  
     */  
    @Override  
    protected void finalize() throws Throwable {  
        super.finalize();  
        System.out.println("Finalizando um produto ....");  
        instancias--;  
    }  
}
```



# Exemplo: Destrutor da classe Produto

```
class Aplicacao {  
    public static void main(String args[]) {  
        System.out.println("\nInstancias prods: " + Produto.getInstancias());  
  
        Produto p1 = new Produto();  
        System.out.println("\nInstancias prods: " + Produto.getInstancias());  
  
        Produto p2 = new Produto("Shulams", 1.99F, 600);  
        System.out.println("\nInstancias prods: " + Produto.getInstancias());  
  
        System.out.println("Produto: " + p1.getDescricao());  
        System.out.println("Produto: " + p2.getDescricao());  
  
        // Referência p1 aponta para produto da referência p2.  
        // produto anteriormente apontado por p1 se torna inacessível.  
        p1 = p2;  
  
        System.out.println("Produto: " + p1.getDescricao());  
        System.out.println("Produto: " + p2.getDescricao());  
        // Coletor de lixo ainda não foi executado.  
        System.out.println("\nInstancias prods: " + Produto.getInstancias());  
  
        int i = System.in.read(); // artifício para parar o programa.  
        System.gc();  
        // Coletor de lixo já foi executado.  
        System.out.println("\nInstancias prods: " + Produto.getInstancias());  
    }  
}
```