Destrutores e herança

Prof. Hugo de Paula

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Departamento de Ciência da Computação

Sumário

Java Garbage Collector

- Java possui coletor de lixo.
- 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 Thowable {
        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("Shulambs", 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());
   }
}
```

Herança

- Mecanismo para definição de uma classe em termos de outra classe existente.
- Relação: *é um tipo de* / *é um*.
- Herança permite o reuso do comportamento de uma classe na definição de outra.
- A classe derivada herda todas as características de sua classe base, podendo adicionar novas características.
- Baseada em dois princípios fundamentais do projeto de software:
  - especificação *(top-down)*
  - generalização / abstração *(bottom-up)*

Herança Simples

- Novas classes, chamadas de classes derivadas (ou subclasses), são definidas a partir de apenas uma classe base (ou superclasse).
- Exemplos:
  - Um professor é uma pessoa.
  - Um ônibus é um veículo.
  - Um automóvel é um veículo.
- Membros da classe base podem ser redefinidos na classe derivada.
- Em Java, qualquer classe herda da classe Object.
- Em Java, usa-se a palavra chave **extends**, para indicar herança.

Exemplo: Construtores em classes estendidas

```
class X {
    protected int mascaraX = 0x00ff;
    protected int mascaraTotal;
    public X() {
        mascaraTotal = mascaraX;
    }
    public int mascara(int orig) {
        return (orig & mascaraTotal);
    }
}

class Y extends X {
    protected int mascaraY = 0xff00;
    public Y() {
        mascaraTotal |= mascaraY;
    }
}
```

Exemplo: Construtores em classes estendidas

- Fases de cada construtor:
  - Chamar construtor da superclasse.
  - Inicializar os campos utilizando suas instruções de inicialização.
  - Executar corpo do construtor.
- Digite as classes X e Y e acompanhe os passos de criação com comandos de impressão (use uma classe Aplicacao para isso).

Construtores em classes estendidas

- Nova classe deve escolher qual construtor da superclasse a chamar.
- Em um construtor da subclasse pode-se chamar diretamente o construtor da superclasse: **super**().
- A assinatura do método **super**() deve ser a mesma assinatura do construtor que se deseja chamar.
- Se não especificar construtor executa-se chamada ao construtor padrão da classe base (se houver).

Exemplo: Bem de Consumo

```
public class BemDeConsumo extends Produto {
    private LocalDate dataValidade;

    public LocalDate getDataValidade() {  return dataValidade;  }
    public void setDataValidade(LocalDate dataValidade) {
        // a data de fabricação deve ser anterior à data de validade.
        if (getDataFabricacao().isBefore(dataValidade.atStartOfDay()))
            this.dataValidade = dataValidade;
    }
    public BemDeConsumo() {
        super();
        // o default é uma validade de 6 meses.
        dataValidade = LocalDate.now().plusMonths(6);
    }
    public BemDeConsumo(String d, float p, int q,
                        LocalDateTime f, LocalDate v) {
        super(d, p, q, f);
        setDataValidade(v);
    }
}
```

Exemplo: Bem Durável

```
public class BemDuravel extends Produto {
    private int mesesGarantia;

    public int getMesesGarantia() { return mesesGarantia; }
    public void setMesesGarantia(int mesesGarantia) {
        if (mesesGarantia > 0)
            this.mesesGarantia = mesesGarantia;
    }
    public BemDuravel() {
        super();
        // o valor default é garantia de 6 meses.
        mesesGarantia = 6;
    }
    public BemDuravel(String d, float p, int q,
                      LocalDateTime f, int g) {
        super(d, p, q, f);
        setMesesGarantia(g);
    }
}
```