


Java Academy
Webinar #2







Construtores: os
Responsáveis Pela
Multiplicação dos Objetos

Construtor


- Chamado quando o objeto é criado
- Não é um método, mas tem características semelhantes
- É definido com o mesmo nome da classe




Construtor Padrão




- Quando o construtor não é fornecido, o Java define um construtor padrão na classe
 - Isto é feito no processo de compilação
- Se um construtor for explicitamente definido, o construtor padrão não é gerado




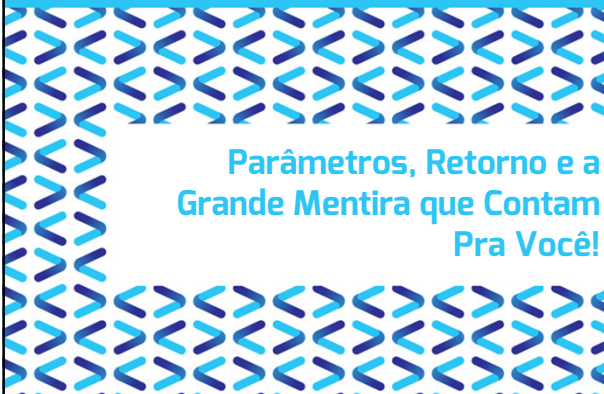
Construtor com Parâmetros



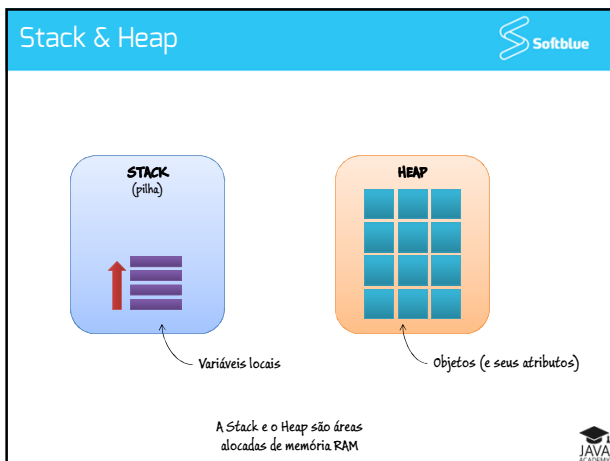
- Construtores podem receber parâmetros da mesma forma que métodos
- Os parâmetros são passados na construção do objeto, no operador **new()**
- A classe pode definir mais de um construtor
 - Tem que haver variação de parâmetros
 - Chamado de **sobrecarga (overloading)**

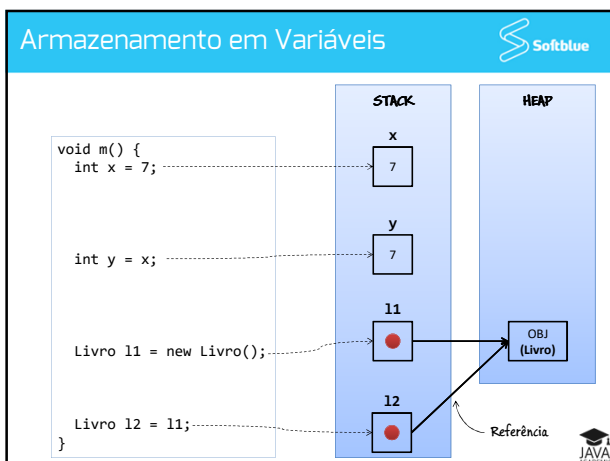






Parâmetros, Retorno e a Grande Mentira que Contam Pra Você!





Passagem de Parâmetro

➤ No Java, a passagem de parâmetros é sempre feita **por cópia!**

- Nunca por referência
- **O conteúdo é sempre copiado**

Muita atenção nisso!

Garbage Collector (Coletor de Lixo)
Softblue

- Serviço da JVM
- A JVM decide quando ele vai executar
- Libera a área de memória do Heap que não é mais acessível

O GC pode fazer a coleta.

Softblue

String: a Classe Fora da Casinha do Java

Strings
Softblue

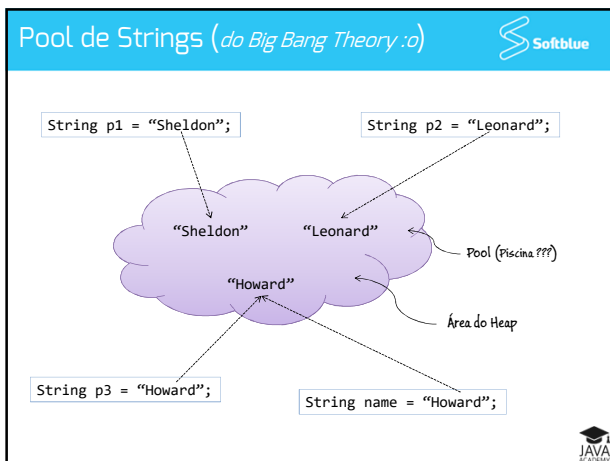
- Armazenam conjuntos de caracteres
- String é uma classe

```
String s = new String("java");
```

Praticamente não se usa

```
String s = "java";
```

Esta sintaxe só existe em Strings!



A Imutabilidade das Strings

- Strings no pool são compartilhadas
- E se a String mudasse?
 - Confusão total!
- Solução: Strings são imutáveis!
- Para mudar uma String, só criando outra
- Para evitar a criação desnecessária de Strings (e sobrecarregar o GC), uma alternativa é usar a classe **StringBuilder**

Softblue


JAVA

Encapsulamento:
Escondendo o Que Não Deve
Ser Visto


Softblue

Encapsulamento

Um dos princípios da orientação a objetos!


ENCAPSULAR  **Esconder** detalhes sobre **como** algo funciona


Importante para possibilitar mudanças futuras!



Modificadores de Visibilidade

Aplicados em classes e elementos de classes


public  Acesso Irrestrito Liberado!


private  Acesso Privado!

No Java, ainda existem os modificadores protected e "default"




Regra Padrão do Encapsulamento


Métodos  **public**


Atributos  **private**


Atributos representam o estado interno do objeto.
Apenas o próprio objeto deve ter acesso a esse estado.

? Como outros objetos vão ler ou modificar atributos, se eles são privados?

 Métodos Getters e Setters!









Elementos Estáticos: Uma Pausa no Uso de Objetos


Atributos e Métodos Estáticos




- Algumas vezes, atributos e/ou métodos podem não estar atrelados a um objeto específico, mas sim à classe
- Atributos ou métodos da classe são assim definidos através do modificador *static*.




Elementos Estáticos




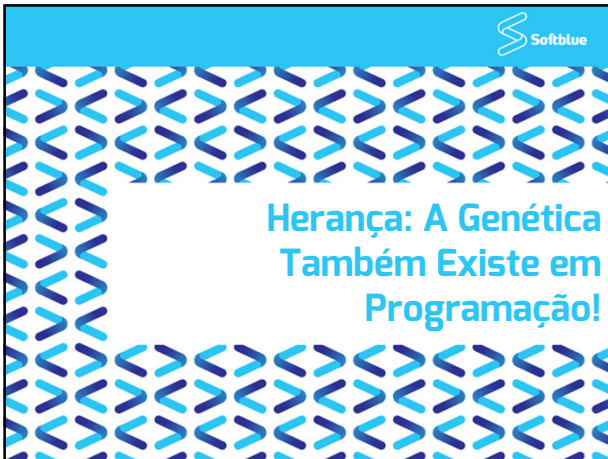
ESTÁTICO  Não está atrelado a um objeto

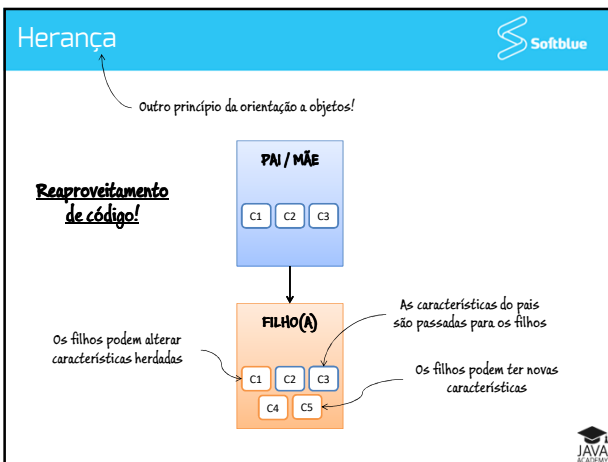
Podem ser estáticos: classes, métodos e atributos

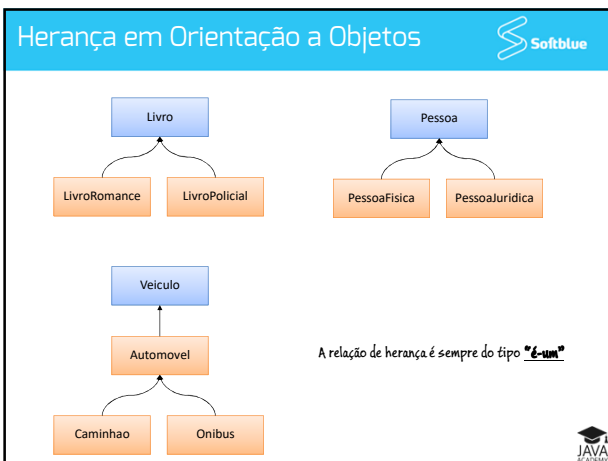
static

 Modificador usado para elementos estáticos









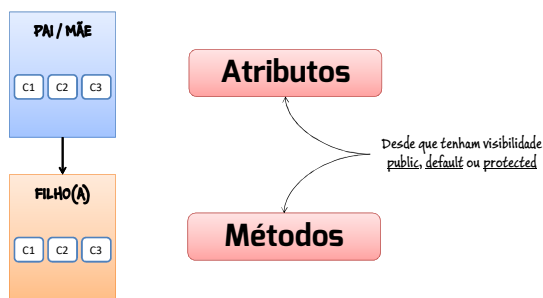
Regras da Herança no Java



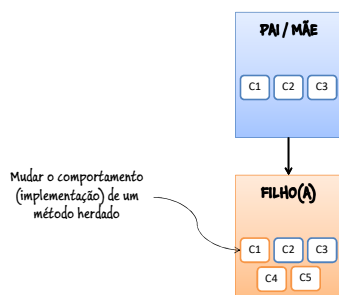
- Uma classe pode ter zero ou mais subclasses
- Toda classe pode ter no máximo uma superclasse direta
 - Herança simples
- Toda classe em Java herda de **Object**
 - Mesmo que você não especifique a herança de forma explícita



O Que é Herdado na Herança?



Sobrescrita (Overriding)



Não confunda **sobrecarga** (overloading) com **sobrescrita** (overriding)

