


Fundamentos de Java


Classes e Objetos




Tópicos Abordados




- Programação procedural e orientada a Objetos
- Classes
 - Atributos e métodos
- Objetos
- Notação UML
- Sobrecarga de métodos
- Objetos e referências
 - Heap e Stack
- Garbage collector
- Operador *this*




Programação Procedural




- Problemas
 - Mudança de requisitos na aplicação
 - Mudança de desenvolvedor
 - Muitas pessoas responsáveis por colocar o mesmo código em vários lugares




Orientação a Objetos




- Benefícios
 - Escrever menos código
 - Concentrar responsabilidades nos locais certos
 - Flexibilizar a aplicação
 - Encapsular lógica de negócio
 - Polimorfismo (variação do comportamento)




Classes: Estruturas de Dados



- Uma classe representa um **tipo de dados**
- É uma **estrutura**





Classes e Seus Métodos







Atributos X Métodos


- Atributos
 - Características da classe
 - Representados por **substantivos**
- Métodos
 - Operações que a classe é capaz de realizar
 - Representados por **verbos**

Classes X Objetos

- A estrutura do **Livro** a qual nós nos referimos não representa um livro propriamente dito
- Ela é apenas uma estrutura (**classe**) usada como modelo para construir os livros propriamente ditos (**objetos**)
- Classe e Objeto são conceitos diferentes!**
- Classes são usadas para **instanciar** objetos

Classes X Objetos


Declarando Classes no Java




- No Java, classes são declaradas utilizando a palavra *class*

```
public class Livro {  
    ...  
}
```

- Um arquivo *.java* pode ter apenas uma classe declarada como pública dentro dele



Declarando Métodos e Atributos




Atributos


Métodos

```
public class Livro {  
    String isbn;  
    int numPaginas;  
    ...  
  
    void emprestar(Cliente c) {  
        ...  
    }  
  
    void devolver() {  
        ...  
    }  
}
```


O Java inicializa os atributos automaticamente



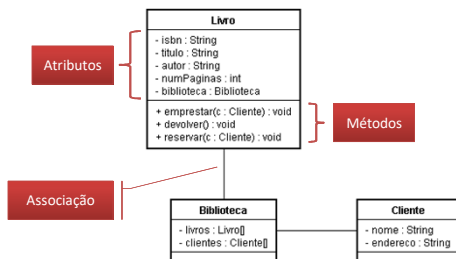
A Notação UML



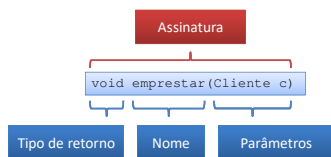
- U**nified **M**odeling **L**anguage
- Utilizada para documentar sistemas orientados a objetos
- Composta por diversos diagramas
 - Um deles é o **Diagrama de Classes**, que mostra as classes do sistema, juntamente com seus respectivos métodos e atributos



A Notação UML: Diagrama de Classes



Assinatura de um Método



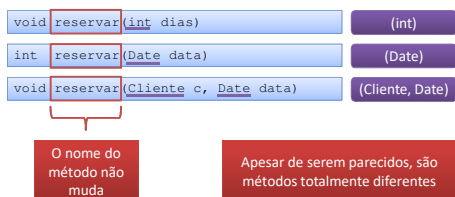
- Se o método não retornar valores, é utilizado o *void*
- Um método pode ter zero ou mais parâmetros, e todo parâmetro deve ter um tipo definido



Sobrecarga de Métodos



- Sobrecarregar um método significa criar outros métodos com o mesmo nome, mas com assinatura diferente



O nome do método não muda

Apesar de serem parecidos, são métodos totalmente diferentes



Sobrecarga de Métodos

```

reservar(10);
void reservar(int dias)

reservar(new Date());
void reservar(Date data)

reservar(new Cliente(), new Date());
void reservar(Cliente c, Date data)

```

Criando e Manipulando Objetos

- Um objeto é sempre instância de uma classe
- Para instanciar objetos, é utilizado o *new*

```

Livro livro1 = new Livro();
Cliente cliente1 = new Cliente();

```

- O objeto possui acesso ao que foi definido na sua estrutura (classe) através do "."

```

livro1.titulo = "Aprendendo Java";
livro1.emprestar(cliente1);

```

Criando e Manipulando Objetos

- Cada objeto criado com o *new* é único
- Os atributos de objetos diferentes pertencem apenas ao objeto

```

Livro livro1 = new Livro();
livro1.isbn = "1234";

Livro livro2 = new Livro();
livro2.isbn = "4321";

Livro livro3 = new Livro();
livro3.isbn = "1212";

```

Cada livro possui o seu próprio ISBN

Objetos e Referências



- Uma variável cujo tipo é uma classe não guarda o objeto diretamente
- A variável guarda uma referência ao objeto
- O *new* aloca uma área de memória e retorna a referência da área de memória alocada
- As variáveis declaradas em métodos são criadas numa área de memória chamada **stack**
- Os objetos são criados numa área de memória chamada **heap**

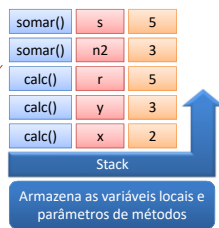


Como Funciona a Stack



```
void calc() {  
    int x = 2;  
    int y = 3;  
    int r = somar(x, y);  
}  
  
int somar(int n1, int n2) {  
    int s = n1 + n2;  
    return s;  
}
```

Em Java, os valores dos parâmetros são sempre copiados



Como Funciona o Heap

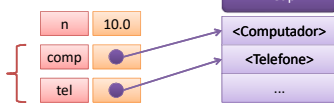


```
double n = 10.0;  
Computador comp = new Computador();  
Telefone tel = new Telefone();
```


O operador *new()* cria um objeto no *heap*

O operador "=" faz a ligação da variável com a referência do objeto

A conteúdo da variável é uma referência (ponteiro) para o objeto no heap



Como Funciona o Heap



```
Computador c1 = new Computador();
Computador c2 = new Computador();
Computador c3 = c2;
c1 = null;
```

Este objeto não pode mais ser utilizado

Heap

<Computador>

<Computador>

...

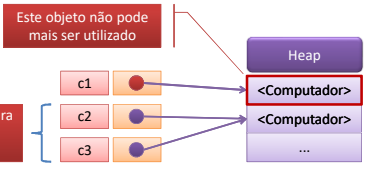
As variáveis apontam para o mesmo objeto armazenado no heap

c1

c2


c3

Garbage Collector



- Serviço da JVM que executa em segundo plano
- Procura objetos no heap que não são mais utilizados pela aplicação e os remove
- Não pode ser controlado pelo desenvolvedor

Garbage Collector



Objeto liberado da memória

Heap

<Computador>

<Computador>

...

Garbage Collector

O que está alocado na stack é liberado assim que o método termina

c1

c2

c3

8

O Operador *this*



- Normalmente não é obrigatório
- Usado em basicamente duas situações
 - Diferenciar um atributo do objeto de um argumento do método
 - Fornecer a referência do próprio objeto para outro método



O Operador *this*



```
public class Circulo {  
    private double raio;  
    public void setRaio(double raio) {  
        this.raio = raio;  
    }  
}
```

Atributo da classe

Parâmetro do método



Palavra de Otimismo!



- Esse é o ponto do curso onde costumam surgir as maiores dificuldades
 - O paradigma de orientação a objetos muda a forma de pensar
- A dificuldade é normal
- É preciso dedicação e prática para conseguir compreender
- Não tente entender tudo de uma só vez