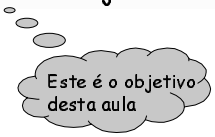
# Introdução à Programação
Orientada a Objetos com Java

Manipulação de Objetos

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco

## Depois de aprender a criar objetos...

Precisamos aprender a
**manipular**
estes objetos




## Para manipular um objeto...

Temos três alternativas:

- Armazenar o objeto para depois ter acesso ao mesmo
- Executar (invocar ou chamar) os métodos do objeto
- Passar o objeto como argumento de um método ou construtor

Tudo isso é feito com as **referências**, não diretamente com os objetos...

## Na verdade, para manipular um objeto...

Temos três alternativas:

- Armazenar a referência para o objeto, para depois ter acesso a mesma
- Executar (invocar ou chamar) os métodos do objeto através da sua referência
- Passar a referência para o objeto como argumento de um método ou construtor

## Armazenando referências em atributos

```
class Cliente {
  private String nome;
  private Endereco endereco;
  Cliente(String n) {
    nome = n;
    endereco = new Endereco();
    ...
  }
  ...
}
```

A referência para o objeto criado é armazenada no atributo **endereco**

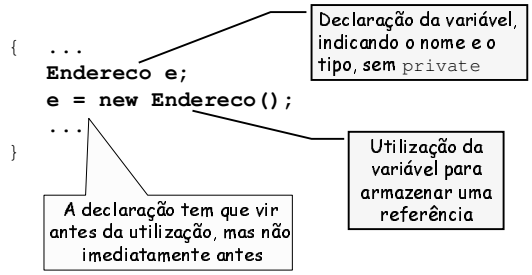
Para ter acesso à referência basta ter acesso a **endereco**

## As variáveis locais...

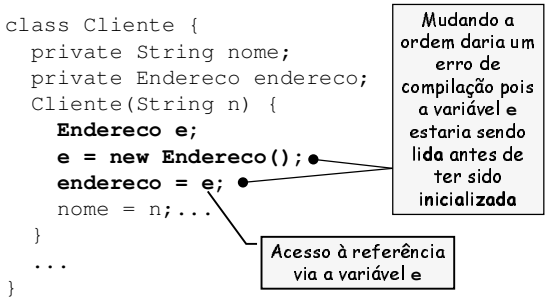
Têm a mesma capacidade de armazenamento que os atributos mas

- São declaradas dentro de um método ou construtor
- Só existem durante a execução do método ou construtor
- Não são inicializadas automaticamente

## Armazenando referências em variáveis locais



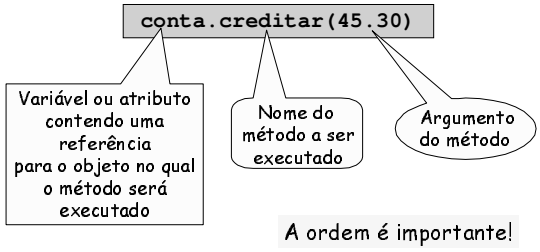

## Inicialização e leitura de variáveis locais



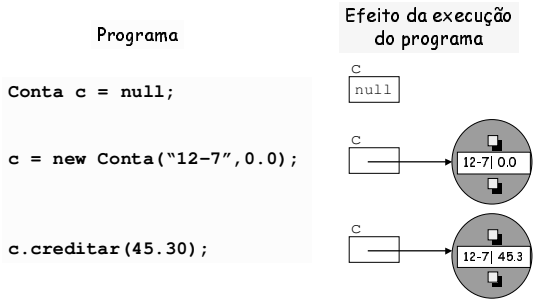

## As variáveis locais servem para...

- Armazenar resultados temporários que serão utilizados depois
- Armazenar resultados que serão utilizados mais de uma vez
- Melhorar a legibilidade do método ou construtor, quebrando uma expressão grande em expressões menores

## Executando métodos




## Efeito da execução de um método




## Um objeto da classe `Console`...

Representa os dispositivos padrões de entrada (teclado) e saída (janela onde o programa está sendo executado) de dados de um computador

## Métodos da classe `Console`...

- `double readDouble()`
  - Espera que o usuário digite um número real e aperte "Enter", devolvendo o número digitado como resultado
- `String readString()`
  - Similar a `readDouble` mas o usuário pode digitar qualquer coisa seguida de "Enter", e retorna uma string, não um número real

## Mais métodos da classe `Console`...

- `void println(double d)`
  - Imprime um número real na janela onde o programa está executando
- `void println(String string)`
  - Imprime uma string na janela onde o programa está executando

## Usando a classe `Console`

```
Console c = new Console();

String numero = c.readString();
double saldoInicial = c.readDouble();

Conta conta;
conta = new Conta(numero,saldoInicial);

conta.creditar(12.90);

console.println(conta.getSaldo());
```

## Comunicação entre objetos

- Os objetos se comunicam para realizar serviços
- A comunicação é feita através da troca de mensagens ou chamada de métodos
- Cada mensagem é uma requisição para que um objeto execute um método específico

## Aliasing

Mais de uma variável armazenando a referência para um dado objeto

```
Conta a = new Conta("12-7",34.00);
Conta b;
b = a;
b.creditar(100);
console.println(a.getSaldo());
```

a e b passam a referenciar o mesmo objeto

qualquer efeito via b é refletido via a

## Remoção de objetos

- Não existe mecanismo de remoção explícita de objetos da memória em Java
- O Garbage Collector de Java elimina um objeto da memória quando este não é mais referenciado

Não existe mais nenhuma variável ou atributo que armazene a referência para este objeto

## O método `finalize`

É possível liberar recursos quando o objeto está na iminência de ser destruído

```
class Conta {...
  void finalize() {
    ...
  }
}
```

O método finalize é chamado antes do objeto ser removido

O último pedido do objeto...

## Passagem de parâmetro

Em Java, a passagem de parâmetro é por valor: o valor da expressão é avaliado primeiro e depois passado para o método chamado

## Passagem de parâmetro por valor

```
class PassagemPorValor {...
  void incrementa(int x)  {
    x = x + 1;
    console.println("x = " + x);
  }
}
```

Concatenação de strings, transformando elementos de outros tipos em `String`

```
PassagemPorValor p;
p = new PassagemPorValor();
int y = 1;
console.println("y = " + y);
p.incrementa(y);
console.println("y = " + y)
```

não altera o valor de `y`

## Referências são valores!

```
class Referencia {
   void redefine(Conta a)  {
     Conta b = new Conta("567-8",55);
     a.creditar(100);
     a = b;
     a.creditar(100);
   }
}
```

não altera o valor de `c`

```
Referencia r = new Referencia();
Conta c = new Conta("123-4",12);
r.redefine(c);
console.println(c.getSaldo());
```

Altera o estado do objeto referenciado por `c`!

## Resumindo...

- Armazenamento de referências em atributos e variáveis
- Execução de métodos
- A classe `Console`
- Aliasing
- Remoção de objetos
- Passagem de parâmetros