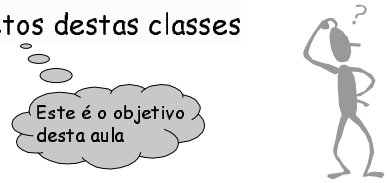
Introdução à Programação
Orientada a Objetos com Java

Criação de Objetos

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco

Depois de aprender a definir interfaces e classes...

Precisamos aprender a
criar
os objetos destas classes



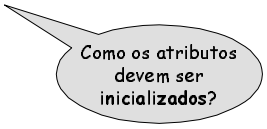
E precisamos criar objetos pois...

As entidades de um sistema, bem como os dados trocados entre elas, são implementadas como **objetos**

- Antes de criar um objeto precisamos especificar as suas características definindo a sua classe
- Os contratos para comunicação e interação entre as entidades são definidos como interfaces

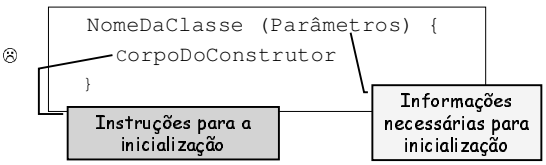
Antes de criar um objeto...

É necessário indicar quais serão os valores iniciais dos seus atributos



Além de métodos e atributos...

O corpo de uma classe pode conter
construtores
definindo como os atributos de um objeto devem ser inicializados



Um construtor sem parâmetros

```
class Conta {
  private String numero;
  private double saldo;
  ...

  Conta() {
    saldo = 0;
    numero = "";
  }
}
```

Mesmo nome da classe

O ";" separa uma instrução de outra, determina a seqüência de execução das instruções

Construtores e declarações com inicialização

```
class Conta {
  private String numero;
  private double saldo;
  Conta() {
     saldo = 0; numero = "";
   } ...
}
```

```
class Conta {
  private String numero = "";
  private double saldo = 0;...
}
```

Soluções equivalentes, quando a inicialização é feita com valores fixos

Nem sempre é o caso

Dois construtores

```
class Conta {
  ...
  Conta() {saldo = 0; numero = "";}
  Conta(String numeroConta,
        double saldoInicial) {
    numero = numeroConta;
    saldo = saldoInicial;
  }
}
```

Várias instruções podem aparecer em uma mesma linha

A ordem de inicialização é determinada pelo programador

Temos overloading quando...

O mesmo **nome** é utilizado para representar mais de uma **construção** (método, construtor, atributo)

O nome fica sobrecarregado

Overloading = Sobrecarga

Overloading e compilação

Distingue-se uma construção da outra pelo uso de "()" ou pela

quantidade, ordem e tipos

dos parâmetros da construção

Caso a distinção não possa ser feita, é gerado um erro de compilação

Por exemplo, não é possível ter dois construtores sem parâmetros na mesma classe

Construtor default de Java

Caso não seja definido um construtor na classe, um construtor **implícito default**, equivalente a

```
NomeDaClasse() {}
```

é fornecido, inicializando os atributos com seus valores **default**...

Valores default para os atributos de java

- `0` para `int`, `double`, e outros tipos numéricos
- `false` para `boolean`
- `null` para os tipos **referência** (`String`, `Conta`, `Cliente`, etc.)

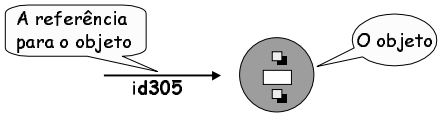
Os tipos em Java podem ser...

- Primitivos
  - `int`
  - `boolean`
  - `double`
  - ...
- Referência
  - Classes
  - ...

Os elementos de um tipo primitivo são **valores primitivos**, enquanto os elementos de um tipo referência são **referências** para objetos!

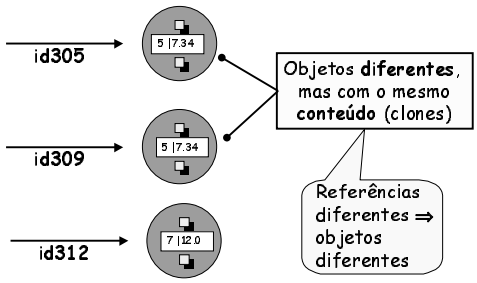
Referências para objetos

Todo o acesso e manipulação de objetos é feito **indiretamente**, através de uma referência para o objeto...



A referência funciona como a **identidade** do objeto

Referências e identidade de um objeto



Criação de objetos

Um objeto é criado através do operador **new**

```
new Conta()
```

Palavra reservada ordenando a criação de um objeto

Nome da classe do objeto a ser criado

Avaliação do operador `new`

Para avaliar uma expressão do tipo
`new NomeDaClasse()`
o computador...

- Cria um objeto da classe `NomeDaClasse` e armazena na sua memória
- Inicializa os atributos deste objeto usando o construtor desta classe que não tem parâmetros
- Devolve como resultado da avaliação uma referência para o objeto criado

Construtores e `new`

A escolha do construtor para inicializar os atributos é determinada pela lista de **argumentos**, entre parênteses

☹ **`new NomeDaClasse(argumentos)`**

O número, ordem e tipos dos argumentos determina o construtor

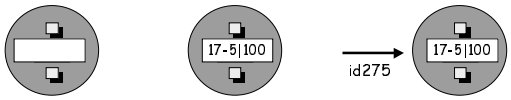
Parâmetros e argumentos

- Os **parâmetros** são nomes que aparecem na **declaração** do construtor
- Os **argumentos** são expressões que aparecem na expressão de criação, na **invocação** do construtor

A mesma definição vale para métodos...

Criando um objeto...

```
new Conta("17-5",100.00)
```



1. Cria o objeto na memória
2. Inicializa o objeto
3. Associa uma referência ao objeto
4. Devolve a referência como resultado da avaliação

A ordem de execução do operador `new`...

- É normalmente 1, 2, 3, 4 (conforme a transparência anterior)
- Pode ser 1, 3, 4, 2 ou 1, 3, 2, 4

Não é equivalente as outras ordens, mas só dá para notar a diferença em ambientes concorrentes...

Equivalente a 1, 2, 3, 4

Resumindo...

- Construtores
- Overloading
- Construtor default e valores default para atributos
- Referências para objetos
- Tipos primitivos e referência
- Parâmetros e argumentos
- O comando `new` e criação de objetos