Introdução à Programação
Orientada a Objetos com Java

Conceitos Básicos de Programação Orientada a Objetos

Paulo Borba
Centro de Informática
Universidade Federal de Pernambuco

Depois de aprender e exercitar como programar...

Precisamos aprender a programar **em Java**

O objetivo desta aula é **começar** a discutir os conceitos básicos desta linguagem de programação

Programar em Java é...

Responder as perguntas abaixo **em Java**:

- Quem vai usar o programa?
- Que serviços são necessários?
- Que entidades realizam estes serviços?
- Como as entidades interagem?
- Como cada entidade realiza os seus serviços?

Não em português, UML, etc.

Respostas em Java

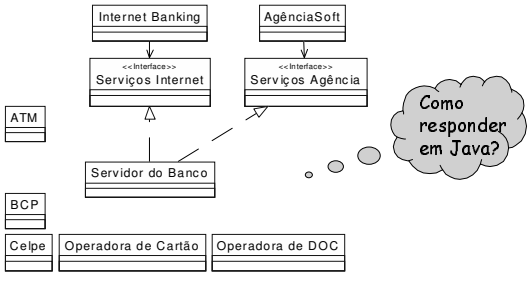
Na verdade, responderemos **em Java** apenas as últimas três perguntas:

- Que entidades realizam os serviços?
- Como as entidades interagem?
- Como cada entidade realiza os seus serviços?

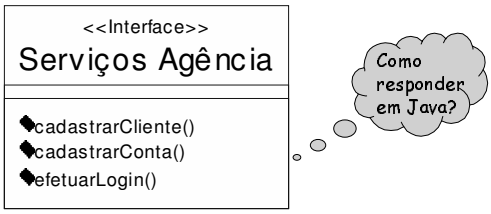
Respostas em Java

As respostas das outras duas perguntas fazem parte da **especificação de requisitos** do sistema, não do **programa**

Respostas em UML



Respostas em UML



Interfaces em Java, quase...

```
interface ServicosAgencia {
  cadastrarCliente();
  cadastrarConta();
  efetuarLogin();
}
```

Há regras que definem quais são os **nomes** válidos de interfaces, operações, etc.

Uma interface indica os serviços, **operações**, a serem oferecidos por uma dada entidade

Para escrever um serviço de uma interface...

Temos que indicar

- o nome do serviço
- as informações que o serviço necessita para ser realizado (parâmetros)
- o resultado gerado pelo serviço
- os erros que podem ocorrer durante a execução do serviço (exceções)

Para escrever um serviço de uma interface...

Temos que indicar a

assinatura

do serviço

Interfaces em Java

```
interface ServicosAgencia {
  void cadastrarCliente(String nome,
       String endereco,...);
  ...
}
```

Palavra-chave (palavra reservada)

Indica que a operação não retorna nenhum resultado

O nome do segundo parâmetro

O tipo do primeiro parâmetro

Um **tipo** é um...

Conjunto de valores relacionados e que são capazes de realizar as mesmas operações

- Todo parâmetro tem um **tipo**, que indica o tipo, as características das informações a serem fornecidas para a realização da operação

O tipo `String`

- É pré-definido:
  - Faz parte de Java,
  - Não precisamos programá-lo
  - Podemos usá-lo já
- Tem como valores qualquer seqüência de caracteres:

```
"Isto é uma String"    "Isto..."
"Um elemento do tipo String!"
""        " "     "    # Teste 1  "
```

As aspas não fazem parte da string, indicam apenas o começo e o fim da mesma

O tipo `String`

Tem várias operações pré-definidas:

- `String toUppercase()`
- `String toLowerCase()`
- `String substring(int indiceInicial)`
- `int length()`

O tipo do resultado da operação

O tipo, pré-definido, que tem como elementos os números inteiros

O tipo `String`

- Tem muito mais operações pré-definidas!
- Veja a documentação da **API** de Java...

Application Programming Interface

Melhorando a qualidade das interfaces...

Ao invés de receber nome, endereço, etc., é melhor agrupar estas informações **relacionadas** e receber um Cliente

```
interface ServicosAgencia {
  void cadastrarCliente(Cliente cliente);
  ...
}
```

Não é pré-definido; temos que definir!

Interfaces em Java

```
interface ServicosAgencia {
  void cadastrarCliente(Cliente cliente);
  String cadastrarConta(Conta conta);
  boolean efetuarLogin(String senha,
            String agencia,String conta);
}
```

O número da conta cadastrada

O tipo que tem apenas dois elementos: `true` e `false`

Interfaces e exceções

```
interface ServicosAgencia {
  void cadastrarCliente(Cliente cliente)
    throws ClienteExistenteException;
  String cadastrarConta(Conta conta);
  boolean efetuarLogin(String senha,
            String agencia,String conta);
}
```

Palavra reservada de Java, seguida dos nomes (separados por ",") dos tipos das exceções que podem ser levantadas pela operação

As exceções...

- Também são agrupadas, classificadas, em tipos
  - Alguns destes tipos, como `Exception`, são pré-definidos
  - Outros, como `ClienteExistenteException`, precisam ser definidos pelo programador

Definindo novos tipos

Ao invés de tipos pré-definidos como `String` e `int`, os tipos específicos da aplicação, como `Conta` e `Cliente`, precisam ser definidos pelo programador

Os elementos dos tipos definidos pelo programador são **objetos**!

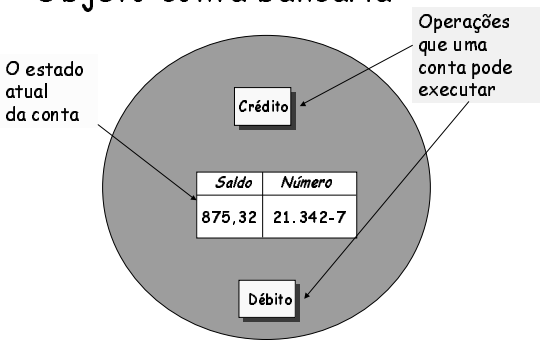
Objeto DVD

As operações que o DVD pode executar

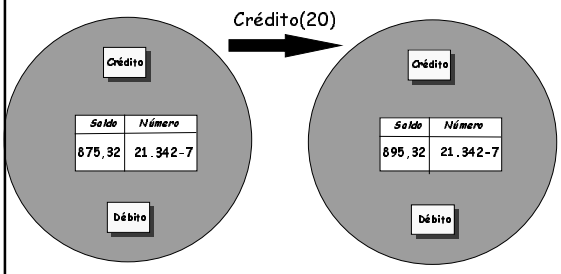
Fonte: http://www.amazon.com

O estado atual do DVD; o que ele está fazendo...

Objeto conta bancária



Estados do objeto conta

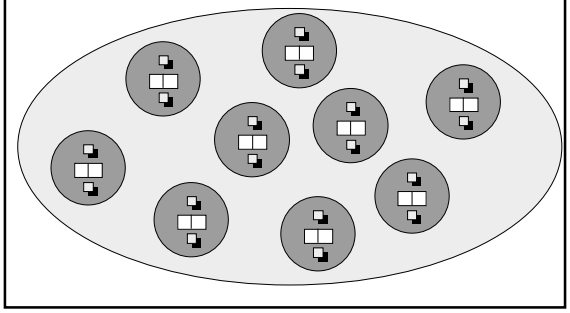


Um objeto consiste de...

- Métodos: as operações que o objeto pode executar
- Atributos: informações sobre o objeto, suas propriedades, seu estado atual

O estado é **encapsulado**, só pode ser modificado pelos métodos do próprio objeto

Classe de contas bancárias



Uma classe é um...

Conjunto de objetos do mesmo tipo, com as mesmas características (métodos e atributos)

Uma classe define um tipo: o tipo cujos elementos são objetos com os mesmos métodos e atributos

Definindo classes em Java

```
class NomeDaClasse {
    CorpoDaClasse
}
```

O corpo de um classe pode conter
- atributos
- métodos
- construtores
- outras classes...

Definindo atributos em Java

```
class Cliente {
  private String nome;
  private String cpfCGC;
  private String endereco;
  ...
}
```

Palavras reservadas

Tipo do atributo

Nome do atributo

Cada atributo tem um tipo específico, que caracteriza as propriedades dos objetos da classe

Que atributos definir?

Os atributos a serem definidos em uma classe devem ser necessários para implementar pelo menos um dos serviços do sistema

Por exemplo, lembre que informações são necessárias para cadastrar um cliente

Information hiding

```
class Cliente {
  private String nome;
  private String cpfCGC;
  ...
}
```

A palavra reservada **private** indica que os atributos só podem ser acessados (isto é, lidos ou modificados) pelas operações da classe correspondente

Information hiding e Java

- Java não obriga o uso de **`private`**, mas vários autores consideram isto uma pré-condição para programação orientada a objetos
- O bug do ano 2000 e **`private`**...
- Grande impacto durante a evolução, modificação do sistema
- Usem **`private`**!

Qualidade!

Definindo atributos em Java

```
class Cliente {
  private String nome, cpfCGC;
  private String rua, CEP, cidade;
  private String pais = "Brasil";
  ...
}
```

Indica mudança de valor

Valor para inicialização

- vários atributos de um mesmo tipo podem ser declarados conjuntamente
- podemos especificar que um atributo deve ser inicializado com um valor específico

Definindo métodos em Java

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

  void creditar(double valor) {
    saldo = saldo + valor;
  }
  ...
}
```

Tipo dos números reais

Um método é uma operação que executa **instruções** e modifica os valores dos atributos do objeto responsável pela sua execução

Comando de atribuição

Um método pode executar várias instruções, mas a mais simples é a instrução (comando) de **atribuição**:

```
atributo = expressão
```

- Para executar esta instrução o computador
  - avalia `expressão`, obtendo um valor
  - faz com que o novo valor de `atributo` seja o valor gerado pela expressão

Definindo métodos em Java

```
class Conta {
  ...
  void debitar(double valor) {
    saldo = saldo - valor;
  }...
```

parâmetros do método

tipo de retorno

corpo do método

Por que o método `debitar` não tem como parâmetro o número da conta?

Tipos e `void`

Usa-se **`void`** para indicar que um método não retorna nenhum valor, apenas altera os valores dos atributos de um objeto

`void` é o tipo que não tem nenhum elemento nenhum!

Definindo métodos em Java

- O tipo do valor a ser retornado pelo método
- Nome do método
- Lista, possivelmente vazia, indicando o tipo e o nome dos argumentos a serem recebidos pelo método
- Exceções que podem ser levantadas pelo método

Métodos que retornam resultados

```
class Conta {
  private String   numero;
  private double   saldo;
  String getNumero() {
    return numero;
  }
  double getSaldo() {
    return saldo;
  }
  ...
```

Os métodos que retornam valores como resultado usam o comando **`return`**

Comando `return`

```
return expressão
```

- Para executar este comando o computador:
  - avalia `expressão`, obtendo um valor
  - devolve este valor como resultado, terminando a execução do método no qual ele se encontra

Que métodos definir?

Os métodos a serem definidos em uma classe devem ser necessários para implementar pelo menos um dos serviços do sistema

Por exemplo, lembre que operações são necessárias para realizar uma transferência

O Corpo do método...

- Contém comandos que determinam as ações a serem realizadas pelo método
- Estes comandos podem
  - realizar simples atualizações dos atributos de um objeto
  - retornar valores
  - executar ações mais complexas como se comunicar com outros objetos, repetir comandos, etc.

Veremos em breve!

Resumindo...

- Assinatura (nome, resultado, parâmetros, exceções) de métodos
- Interfaces
- Tipos (`int`, `String`, `double`, `void`, `boolean`, etc.)
- Objetos, classes, métodos e atributos
- Encapsulamento e *information hiding*
- Comandos de atribuição e `return`