

Introdução a Programação - IF669
<http://www.cin.ufpe.br/~if669>

Conceitos Básicos de Orientação a Objetos e Strings


AULA 06

Ricardo Massa F. Lima Sérgio C. B. Soares
rmfl@cin.ufpe.br scbs@cin.ufpe.br


Até aqui . . .

- Conceitos gerais de programação
- Tipos primitivos, entrada e saída de dados, operadores
- Comando condicional, seleção e repetição
- **HOJE:** Introdução a orientação a objetos



Objetivos

- Compreender
 - Benefícios da programação orientada a objetos
 - Conceitos básicos e terminologia da programação orientada a objetos
 - objeto, classe, atributo, método




Benefícios da tecnologia de objetos

- Acelerar o tempo de desenvolvimento
- Reduzir o tempo de manutenção
- Mais fácil de entender e adaptar
- Código de melhor qualidade




Programação orientada a objetos

- Foco nos dados (objetos) do sistema, não nas funções
- Estruturação do programa é baseada nos dados, não nas funções
- As funções mudam mais do que os dados
- Objetos e atividades do mundo real



O que é um objeto?

- É o agrupamento dos dados e operações que representam um **conceito**
 - Conta bancária
 - número e saldo
 - creditar e debitar
 - Aluno da UFPE (cadastrado no Sig@)
 - nome, cpf, endereço ...
 - corrigir nome, atualizar endereço
 - Produto (de supermercado)
 - código, descrição, valor ...
 - atualizar estoque, remarcar preço...



Objetos

- Blocos básicos para construção de um programa
- Contém dados que podem ser usados e modificados
- Possuem
 - Identidade (identificação única)
 - Estado (os valores armazenados)
 - Interface (como se comunicar com ele)
 - Comportamento (operações que pode executar)



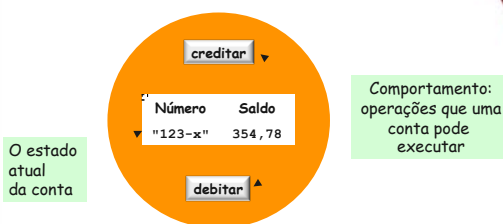
Objetos



- Um carro pode ser considerado um objeto
 - Identidade ("1")
 - Estado (sua cor, tipo de pneu, etc)
 - Interface (volante, pedal do freio, etc)
 - Comportamento (respostas ao giro do volante, ao pisar o pedal do freio)
- Muitos textos definem um objeto como possuindo duas características apenas: estado e comportamento
 - Nesses casos, a identidade é parte do estado e a interface é parte do comportamento

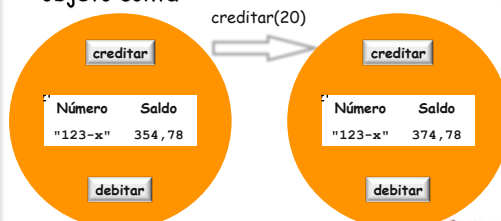


Objeto Conta Bancária



Estados do Objeto Conta

- Comportamento mudou o estado do objeto conta



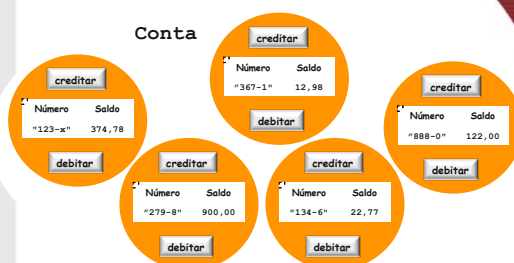
Classe

- Agrupamento de objetos que têm propriedades comuns e realizam as mesmas operações
- Descreve como os objetos daquela classe são estruturados internamente (propriedades e operações)
- Classe é um conceito, o objeto é uma instância deste conceito
- Portanto, podemos ter vários objetos pertencentes a mesma classe
 - Todos os objetos conta de um banco (um para cada conta)

Classe é uma forma, objetos são os bolos



Classe x Objeto



Múltiplos objetos são criados a partir da mesma classe



Mas como declarar uma classe em Java?

Exemplo:

- Temos uma aplicação bancária que deverá armazenar os dados de todas as contas correntes de um banco
- Contas têm saldo e número e podemos realizar créditos e débitos nas mesmas

Definindo Classes em Java

```
public class Conta {  
    CorpoDaClasse  
}
```

- O corpo de uma classe pode conter
 - atributos (dados)
 - métodos (operações)
 - construtores (inicializadores)
 - outras classes...

Definindo Atributos em Java

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

- cada atributo tem um tipo específico que caracteriza as propriedades dos objetos da classe
- `double` e `String` denotam os tipos cujos elementos são reais e strings (texto)

Tipos em Java

Primitivos

- `char`
- `int`
- `boolean`
- `double`
- ...

Referência

- classes (`String`, `Object`, `Livro`, `Conta`, etc.)
- interfaces
- arrays

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

Strings (String)

- Não é um tipo primitivo e sim uma classe
 - Literais: `" "` `"a"` `"CIn\nUFPE"`
 - Operadores: `+` (concatenação)
`"maio " + "de " + 99 = "maio de 99"`
- Note a conversão de inteiro para string
- Há uma conversão implícita de todos os tipos primitivos para `String`

Mais operadores sobre Strings

- Comparação (igualdade) de dois `String`

```
String a ...  
String b ...  
if ( a.equals(b) ) {  
    ...  
}
```

- Tamanho de um `String` a

```
String a ...  
int i = a.length();
```

Usando String

```
String s1 = "ricardo";
String s2 = "Ricardo";
if (s1.equals(s2)) {
    System.out.println("igual");
} else {
    System.out.println("diferente");
}
if (s1.equalsIgnoreCase(s2)) {
    System.out.println("igual");
} else {
    System.out.println("diferente");
}
```

Mais operadores sobre Strings

- Extrair um substring de um String maior

```
String saudacao = "Bem-vindo";
String s = saudacao.substring(0,3);
// s = "Bem"
// caracteres das posições 0, 1 e 2
```

A classe String em Java contém mais de 50 métodos
<http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html>

Information Hiding

Não use preposições

```
public class Livro {
    private int anoDePublicacao;
    private String titulo;
    ...
}
```

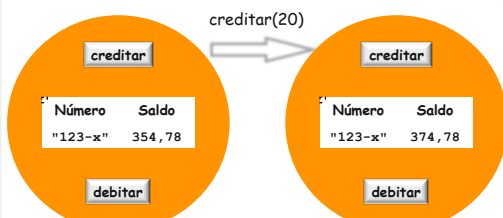
A palavra reservada `private` indica que os atributos só podem ser acessados (isto é, lidos ou modificados) pelas operações da classe onde foram definidos

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 em extensibilidade
- Usem `private`!

Definindo métodos em Java

- Como definir o método `creditar`?



Definindo métodos em Java

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

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

Um método é uma operação que realiza ações e modifica os valores dos atributos do objeto responsável pela sua execução

Definindo métodos em Java

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

Diagram labels:

- tipo de retorno (points to `void`)
- parâmetros do método (points to `double valor`)
- corpo do método (points to `saldo = saldo - valor;`)

Por quê o método `debitar` não tem como parâmetro o número da `Conta`?

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

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

Definindo métodos em Java

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

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

O corpo do método

- Comandos que determinam as ações do método
- Estes comandos podem
 - realizar simples atualizações dos atributos de um objeto
 - retornar valores
 - executar ações mais complexas como se comunicar com outros objetos

Comunicação entre objetos

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

```
Conta c = ...  
c.creditar(45.30);
```

variável contendo referência para objeto

nome do método a ser executado

Exercício 1 (10 minutos)

- Faça uma classe chamada `Exercicio1` para ler o nome completo de uma pessoa e imprimir o primeiro e último nomes
 - o nome completo deve ter pelo menos dois nomes

```
Digite o nome completo: Sérgio Castelo Branco Soares  
Primeiro: Sérgio  
Último: Soares
```

<http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html>

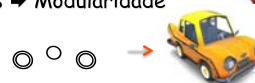
Utilizando OO para desenvolver programas

- Desenvolvimento de uma solução é mais fácil quando quebramos esta solução em módulos gerenciáveis
- Desenvolver módulos separados, onde cada um é responsável por uma certa parte da solução
 - **Programação Modular**
- OO facilita a programação modular
 - Módulos são as classes e objetos



Algumas Considerações sobre OO

- Orientação a Objetos ➔ Modularidade
 - Reusabilidade
 - Extensibilidade
- Linguagens OO têm em objetos, classes, ..., elementos fundamentais para construir programas
 - Estruturas da linguagem permitem mapeamento direto dos conceitos de OO



Os conceitos de Orientação a Objetos são independentes da linguagem de programação

Exercício 2 (10 minutos)

- Crie a classe `Conta` conforme apresentado em sala
 - atributos (`numero` e `saldo`)
 - métodos (`creditar`, `debitar`, `getSaldo`, `getNumero`)

Exercício 3 (15 minutos)

- Crie uma classe `Produto` para representar os produtos de um supermercado
- Cada produto tem uma descrição (texto), valor (real) e quantidade em estoque (inteiro)
- Lembre-se de definir métodos para ler e alterar os atributos dos produtos