

PROGRAMAÇÃO ORIENTADA A OBJETOS

PRINCÍPIOS DA POO



1/5/2008

Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.

1



AGENDA



1. Paradigmas de Programação
2. Programação Orientada a Objetos
3. Vantagens da POO
4. Modularização, Abstração e Encapsulamento
5. Classes e Objetos
6. Atributos, Métodos e Construtores
7. Modificadores de Atributos e Métodos
8. Sobrecarga de Métodos e Construtores
9. Outros Métodos

1/5/2008

Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.

2



PARADIGMAS DE PROGRAMAÇÃO



- **Programação Binária**
 - Grande propensão a erros;
 - Manutenção difícil por falta de estrutura;
- **Programação Procedural**
 - Adição de estruturas de apoio (procedimentos);
 - Sem reutilização;
 - Sem encapsulamento;

1/5/2008

Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.

3



PARADIGMAS DE PROGRAMAÇÃO



- **Programação Modular**
 - Divide os programas em vários componentes;
 - Não é extensível;
- **Programação Orientada a Objetos**
 - Adiciona **herança** e **polimorfismo** ao módulo;
 - Objetos interagindo entre si, orientando o fluxo global do programa;

1/5/2008

Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.

4



PROGRAMAÇÃO ORIENTADA A OBJETOS



- Coloca as estruturas de dados em primeiro lugar, examinando depois os algoritmos que vão processar os dados;
- Não existe “topo” ou “início”;
- Inicia encontrando ou determinando as classes e atributos e depois acrescentam-se métodos a cada classe.

1/5/2008

Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.

5



PROGRAMAÇÃO ORIENTADA A OBJETOS




- Uma regra simples:
 - Na identificação de **classes**, procurar por **substantivos** na análise do problema;
 - Na identificação de **métodos**, procurar por **verbos** na análise do problema;


1/5/2008

Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.

6



PROGRAMAÇÃO ORIENTADA A OBJETOS




- Em vez de modelar o programa como um conjunto de procedimentos e dados separados, a modelagem se dá através de substantivos, verbos e adjetivos do **domínio** do problema;
- Nova maneira de ver o software:
 - Ver o todo de forma conceitual;
 - O que o objeto **faz**, seu **comportamento**;
 - Programas naturais e reais.


• **Domínio:**

- Espaço onde o problema reside.
- Conjunto de conceitos que representam os aspectos importantes do problema.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 7




PROGRAMAÇÃO ORIENTADA A OBJETOS




- Orientação a Objetos (OO)** é um termo geral que inclui qualquer estilo de desenvolvimento que seja baseado no conceito de **objeto**;
- Um **objeto** é uma construção de software que encapsula estado e comportamento.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 8



PROGRAMAÇÃO ORIENTADA A OBJETOS




- Os objetos permitem a modelagem de software em **termos reais e abstrações**.

• **Objeto:**


- Tudo que é manipulável / manufaturável;
- Tudo que é perceptível por qualquer dos sentidos.

• **Abstração** é o processo mental em que as idéias estão distanciadas dos objetos.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 9




VANTAGENS DA PROGRAMAÇÃO ORIENTADA A OBJETOS




- Natural:** Não precisa saber como o software funciona;
- Confiável:** Quantas vezes o seu microondas quebrou?
- Reutilizável:** Não é necessário reinventar a roda;
- Manutenível:** 20% do tempo de vida do software é em desenvolvimento;
- Extensível:** Software não é estático;
- Oportuno:** Dividir para conquistar.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 10




NATURAL




- Não precisa saber como o software funciona.**
- Em vez de ajustar o seus programas para a linguagem do mundo dos computadores (regiões de memória), pode-se programar usando expressões do domínio do problema;
- Modelagem no nível funcional e não de implementação;
- “Não interessa o como, somente o que é feito.”

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 11




CONFIÁVEL



- Quanta vezes seu microondas quebrou?**
- A natureza modular dos objetos permite que se façam alterações em partes do programa sem que isso afete outras partes;
- Os objetos isolam o conhecimento e a responsabilidade de onde pertencem;
- Por serem isolados, os objetos podem ser testados separadamente, validando assim cada componente do software;


1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 12



REUTILIZÁVEL

- ***Não é necessário reinventar a roda.***
- Possibilita a extensão de objetos existentes (herança);
- Possibilita a escrita de códigos genéricos (polimorfismo);
- Exemplo:
 - Biblioteca Java.
- **Observação:**
 - A OO não garante código genérico.
 - Isso é obrigação do programador.


1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 13



MANUTENÍVEL

- **20% do tempo de vida do software é em desenvolvimento.**
- 60% a 80% do tempo gasto trabalhando em um programa é em manutenção;
- Para corrigir um erro, simplesmente se corrige em um lugar;
- Como uma mudança na implementação é transparente, todos os outros objetos irão se beneficiar.


1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 14



EXTENSÍVEL

- **Software não é estático.**
- Um software tem que crescer e mudar com o passar do tempo para permanecer útil;
- Existem vários recursos para se estender o código:
 - Herança.
 - Polimorfismo.
 - Padrões de projetos.


1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 15



OPORTUNO

- **Dividir para conquistar.**
- POO diminui o tempo do ciclo de desenvolvimento;
- O software natural, transforma um grande problema complexo e vários problemas simples;
- O desenvolvimento de cada problema simples pode ser realizado em paralelo.


1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 16



MODULARIZAÇÃO

- Processo de dividir um todo em partes bem definidas, que podem ser construídas e examinadas separadamente e que interagem de maneiras bem definidas.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 17



ABSTRAÇÃO

- Capacidade de **ignorar detalhes** de partes para focalizar a atenção em um nível mais elevado de um problema;
- Analisar o contexto real das entidades e abstrair seus **dados e comportamento** de forma a representá-los computacionalmente.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 18

ABSTRAÇÃO

Mundo-Real

Abstração

Software

- **Entidade:** Conta Bancária
- **Dados:**
 - número da conta
 - saldo da conta
 - nome do cliente
 - ...
- **Comportamento:**
 - retirar valor
 - depositar valor
 - mostrar saldo
 - ...

1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
19

ABSTRAÇÃO

Mundo-Real

Abstração

Software

- **Entidade:** Aluno
- **Dados:**
 - nome
 - CPF
 - endereço
 - ...
- **Comportamento:**
 - matricular em disciplina
 - trancar disciplina
 - ver notas
 - ...

1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
20

ENCAPSULAMENTO

- Combinar **dados** e **comportamento** em um pacote e ocultar a implementação dos dados do usuário do objeto;
- Objetos ou entidades externas não precisam saber do funcionamento interno de outro objeto.

1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
21

ENCAPSULAMENTO

NOME DA CLASSE

1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
22

ABSTRAÇÃO-ENCAPSULAMENTO


- **Enunciado:**
 - Para um sistema bancário, identificou-se uma entidade chamada conta bancária. Cada conta bancária possui um número, um cliente e um saldo. É permitido depositar e retirar valores, bem com consultar o saldo de cada uma das contas bancárias. Não é permitido retirar valores superiores ao saldo disponível para saque.
- **Abstrair dados e comportamento** de uma conta bancária do enunciado acima e encapsular estes em uma classe chamada ContaBancaria.

1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
23

CLASSES E OBJETOS

- OO baseia-se em conceitos da realidade (Objetos, Estados e Estímulos) para a modelagem e construção de sistemas de software.


1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
24



OBJETO

- Pode ser qualquer coisa na natureza que possua **características e comportamentos**;
- Algo (coisa ou evento) que existe no mundo real;
- Objetos Java representam objetos a partir do domínio de um problema.


1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
25



CLASSE

- É uma **abstração de um conjunto de objetos** que possuem os mesmos tipos de características;
- Exemplo:
 - Classe: Conta Bancária
 - Objetos: Conta Bancária do João, Conta Bancária do Pedro, Conta Bancária da Maria.


1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
26



CLASSE

- Define os atributos e métodos:
 - **Atributos** : características de uma classe visível externamente;
 - **Método**: ação executada por um objeto quando passa uma mensagem ou em resposta a uma mudança de estado;
- Atua como molde para criar ou instanciar objetos.

1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
27




CLASSE *versus* OBJETO

- Classes representam o **conceito geral** de uma coisa;
- Objetos representam **instâncias concretas** de uma classe.
- Exemplo:
 - **Classe**: ContaBancaria
 - **Objetos**: c1, c2 e c3

```
ContaBancaria c1, c2, c3;
c1 = new ContaBancaria(10001, "Pedro Paulo", 10.0);
c2 = new ContaBancaria(10002, "João Paulo", 20.0);
c3 = new ContaBancaria(10003, "José Paulo", 30.0);
```


1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
28



ATRIBUTOS E MÉTODOS

- **Atributos** são os dados que caracterizam um objeto:
 - Definem o estado do objeto.
- **Métodos** definem o comportamento do objeto:
 - Através deles podemos nos comunicar com os objetos;
 - Podem manipular os atributos de um objeto.

1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
29



ATRIBUTOS E MÉTODOS

```
public class ContaBancaria {
    private long numero;
    private String nome;
    private double saldo;

    public double deposito (double valor) {
        ...
    }


    public double saque(double valor) {
        ...
    }

    public double aplicaJuros(double taxa) {
        ...
    }
}
```


ATRIBUTOS

MÉTODOS

1/5/2008
Programação Orientada a Objetos
Prof. Ademir Schmitz, M.Sc.
30




CHAMADA DE MÉTODOS




- Os métodos podem chamar outros métodos da mesma classe como parte de sua implementação (**chamada de método interno**);
- Os métodos podem chamar métodos de outros objetos utilizando a notação de ponto (**chamada de método externo**).

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 31




CONSTRUTORES




- Permitem que cada objeto seja configurado adequadamente quando é criado;
- O construtor inicializa o objeto para um estado adequado.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 32




CONSTRUTORES




- Detalhes dos construtores:
 - Um construtor tem o mesmo nome da classe.
 - Um construtor por ter um ou mais parâmetros (ou mesmo nenhum).
 - Um construtor é sempre chamada através da palavra **new**.
 - Um construtor não retorna nenhum valor.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 33




CONSTRUTORES




```
public ContaBancaria(long numero, String nome, double saldo) {
    this.numero = numero;
    this.nome = nome;
    this.saldo = saldo;
}

public ContaBancaria(long numero, String nome) {
    this.numero = numero;
    this.nome = nome;
    this.saldo = 0.0;
}
```

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 34




CONSTRUTORES




- Um **construtor padrão** é um construtor sem parâmetros;
- Se não for declarado um construtor na classe, Java vai providenciar um **construtor padrão**;
- Com o construtor padrão do Java, todos os atributos recebem valores **default**.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 35




MODIFICADORES DE VISIBILIDADE




- Também conhecidos como modificadores de acesso;
- Definem a visibilidade de um atributo, método ou construtor.

1/5/2008 Programação Orientada a Objetos Prof. Ademar Schmitz, M.Sc. 36




MODIFICADORES DE VISIBILIDADE




- Os modificadores de visibilidade são:
 - public**: todas as classes externas têm acesso;
 - private**: nenhuma classe externa têm acesso;
 - protected**: apenas subclasses e classes dentro do mesmo pacote têm acesso;
 - package**: apenas classes dentro do mesmo pacote têm acesso.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 37




MODIFICADORES DE VISIBILIDADE




- Enunciado:**
 - No caso da classe *ContaBancaria*, qual deverá ser a visibilidade de cada um de seus atributos, métodos e construtores?

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 38




MODIFICADORES GERAIS DE ATRIBUTOS E MÉTODOS




- Definem comportamentos diferenciados para atributos e/ou métodos.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 39




MODIFICADORES GERAIS DE ATRIBUTOS E MÉTODOS




- static**: pertence a classe, e não a um objeto em particular:
 - Atributos Estáticos** são conhecidos como **variáveis de classe** ou **variáveis estáticas**;
 - Existe exatamente uma cópia de uma durante todo o tempo, independente do número de instâncias criadas;

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 40




MODIFICADORES GERAIS DE ATRIBUTOS E MÉTODOS




- static**: pertence a classe, e não a um objeto em particular:
 - Métodos estáticos** são conhecimentos também como **métodos de classe**;
 - Podem ser chamados a partir da classe, e não existe a necessidade de criação de uma instância para a sua utilização;
 - Podem acessar apenas atributos e métodos estáticos.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 41




MODIFICADORES GERAIS DE ATRIBUTOS E MÉTODOS




- final**: não pode ter o valor alterado (atributo), não pode ser sobrescrito (método);
- Outros**: abstract, native, synchronized, volatile, etc.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 42




MODIFICADORES GERAIS DE ATRIBUTOS E MÉTODOS




- **Enunciado:**
 - Como definir uma taxa de juros única a ser aplicada para todas as contas bancárias instanciadas pela classe ContaBancaria?
 - Como definir uma sequência de números para as contas bancárias?
 - Como estabelecer um limite mensal de operações sem cobrança de taxa?
- **Observação:** métodos com o modificador *final* serão vistos quando falarmos sobre herança.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 43




SOBRECARGA




- Métodos e construtores podem ser sobrecarregados;
- Sobrecarga significa dar o **mesmo nome** a mais de um método ou construtor;
- Também conhecida como **overloading**;

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 44




SOBRECARGA




- A **assinatura** deve ser diferente;
- Fazem parte da assinatura de um método:
 - Nome;
 - Tipo dos parâmetros;
 - Quantidade de parâmetros.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 45




SOBRECARGA DE MÉTODOS




- Pode ocorrer entre métodos da mesma classe;
- Pode ocorrer entre métodos de classes diferentes (quando um método é herdado e outro é declarado).
- **Observação:** o segundo caso será visto quando trabalharmos com herança.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 46



SOBRECARGA DE MÉTODOS




```

public double adicionaJuros(double taxa) {
    saldo = saldo + (saldo * (taxa/100.0));
    return saldo;
}


public double adicionaJuros() {
    saldo = saldo + (saldo * (taxaJuros/100.0));
    return saldo;
}

```

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 47



SOBRECARGA DE CONSTRUTORES




```

public ContaBancaria(long numero, String nome,
double saldo) {
    this.numero = numero;
    this.nome = nome;
    this.saldo = saldo;
}


public ContaBancaria(long numero, String nome) {
    this.numero = numero;
    this.nome = nome;
    this.saldo = 0.0;
}

```

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 48




TIPOS DE VARIÁVEIS




1. Variáveis de instância (**atributos**):
 - Pertencem a um objeto;
 - São inicializados com um valor default;
2. Variáveis de classe (**static**):
 - Pertencem a uma classe;
 - São inicializados com um valor default;
3. Variáveis locais:
 - Pertencem a um método ou bloco de código;
 - Devem ser inicializadas explicitamente;
4. Variáveis de parâmetros:
 - Pertencem a um método;
 - Devem ser inicializadas explicitamente.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 49




MÉTODOS gets() e sets()




- Implementados para permitir que atributos com **visibilidade private** possam ser alterados ou consultados por entidades externas:
 - **tipo getNomeDoAtributo()**: retorna o valor do atributo;
 - **setNomeDoAtributo(valor)**: atualizar o valor do atributo com o valor passado como parâmetro.
- Normalmente todos os atributos privados possuem o respectivos métodos get e set.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 50




OUTROS MÉTODOS




- **public boolean equals(Object obj)**
 - Deve ser implementado em cada classe para definir a igualdade de objetos instanciados com a respectiva classe;
- **public String toString()**
 - Deve ser implementado em cada classe para definir a representação string de objetos instanciados com a respectiva classe;
- **public Object clone()**
 - Deve ser implementado em cada classe para permitir a clonagem (cópia idêntica) de objetos instanciados com a respectiva classe;
- **public int compareTo(Comparable obj)**
 - Deve ser implementado em cada classe para definir o relacionamento igual, maior ou menor entre objetos instanciados com a respectiva classe;

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 51




PROJETO DE CLASSES




1. Descubra o que lhe é pedido para fazer com um objeto da classe em questão;
2. Determine os atributos;
3. Dê nomes aos métodos;
4. Documente a interface pública;
5. Determine os construtores;
6. Implemente os métodos;
7. Teste a classe.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 52



PROJETO DE CLASSES



1. Sempre mantenha dados privados;
2. Sempre inicialize os dados (atributos);
3. Não use tipos básicos em demasia numa classe;
4. Nem todos os campos precisam de “acessadores” e “modificadores” de campos individuais;
5. Use uma forma padrão de definição das classes;
6. Divida classes com tarefas demais;
7. Dê nomes às classes e métodos que representam suas tarefas.

1/5/2008 Programação Orientada a Objetos Prof. Ademir Schmitz, M.Sc. 53