Classes, Métodos e Atributos

Introdução

- A orientação a objetos visa prover mecanismos de abstração e de de composição para o desenvolvimento de software
- São mecanismos de abstração:
  - Classes, atributos, métodos, interfaces
- São mecanismos de composição:
  - Herança, associação, agregação

Mecanismos de Abstração

- Abstrações visam representar conceitos do mundo real ou do mundo computacional de forma a
  - Prover estado
  - Prover comportamento
  - Ocultar detalhes de implementação
  - Encapsular conceitos relacionados

Classes

- O principal mecanismo de abstração das linguagens orientadas a objetos são as classes
- Classes englobam estado e comportamento relacionados.
- Exemplos de possíveis classes:
  - Pessoa, Conta, Veículo, Aluno, Matrícula, ...
- As classes são compostas básicamente de atributos e métodos (e construtores).

Classes em Java

- A linguagem Java permite a criação de classes públicas e classes privadas.
- Classes públicas podem ser acessadas por outras classes definidas em outros arquivos.
- Classes privadas são acessíveis somente no arquivo no qual elas estão definidas.
- É comum a criação de uma classe por arquivo (no caso, sendo uma classe pública).

Definição de uma Classe

```
public class Conta {

}
```

Declarando variáveis de uma classe

- A classe Conta recém criada pode ser usada em outras classes.
- Exemplo:

```
public class TesteConta{
    public static void main(String[] args){
        Conta c;
    }
}
```

Criando objetos da classe

- Uma classe serve como um *template* para a criação de objetos.
- Para criar objetos de uma classe são usados os chamados **construtores**

```
public class TesteConta{
    public static void main(String[] args){
        Conta c = new Conta();
    }
}
```

Adicionando estado

- Para que uma classe seja útil, ela deve ter formas de:
  - Manter estado e/ou
  - Desempenhar alguma funcionalidade
- Objetos armazenam estado através de atributos
  - Assim como nos programas estruturados isso era feito através de variáveis globais, por exemplo

Atributos - Sintaxe

- Os atributos possuem a seguinte sintaxe:

  [modificadores] tipo nome [= valor];

Adicionando estado

- Considere, por exemplo, que a classe Conta possua um atributo saldo:

```
public class Conta {
    double saldo;
}
```

Usando atributos

- Os atributos podem ser usados para leitura e para escrita.
- A sintaxe para acesso a atributos é:

  objeto.atributo

- No entanto, normalmente os atributos são acessados indiretamente, através de métodos de leitura e escrita (get e set).

Acessando atributos com get e set

- O uso de atributos diretamente pelos clientes de uma classe (TesteConta é um cliente de Conta) é desencorajado.
  - Quaisquer mudanças na estrutura interna da classe acarretariam em mudanças nos clientes
- O uso de métodos de leitura (get) e escrita (set) visam desacoplar os atributos de uma classe dos clientes que a utilizam.

Exemplo – Usando Atributos

```
public class TesteConta{
    public static void main(String[] args){
        Conta c = new Conta();
        c.setSaldo(100);
        System.out.println("Saldo: " + c.getSaldo());
    }
}
```

Modificadores de visibilidade

- Existem alguns modificadores de visibilidade para classes, métodos e atributos em Java.
- Os principais são:
  - public – Visível para todas as classes
  - private – Visível apenas dentro da classe
  - protected – Visível somente para as sub-classes
  - (sem modificador) Visível para todas as classes no pacote (diretório)

Exemplo – Conta com get e set

```
public class Conta {
    private double saldo;
    public void setSaldo(double aSaldo){
        saldo = aSaldo;
    }
    public double getSaldo(){
        return saldo;
    }
}
```

Exemplo – TesteConta com get e set

```
public class TesteConta{
    public static void main(String[] args){
        Conta c = new Conta();
        c.setSaldo(100);
        System.out.println("Saldo: " + c.getSaldo());
    }
}
```

Adicionando comportamento

- Métodos get e set representam comportamentos, embora simples.
- No entanto, as classes precisam normalmente de mais métodos
  - De forma a implementar as funcionalidades para as quais ela foi criada.
- Para a classe Conta, por exemplo, métodos precisariam ser criados para: sacar, transferir, depositar, ...

Métodos - sintaxe

- Métodos são similares a procedimentos ou funções em linguagen estruturadas.
- A sintaxe é a normalmente a seguinte:

```
[modificadores] tipoRetorno nome(parametros){
   // Corpo do método ...
}
```

Métodos - Exemplo

- Considere um método para saque de valores de uma conta...

```
public class Conta {
    private double saldo;
    public void sacar(double valor){
        saldo = saldo – valor;
    }
    ...
}
```

Exemplo – Chamada a método

```
public class TesteConta{
    public static void main(String[] args){
        Conta c = new Conta();
        c.setSaldo(100);
        System.out.println("Saldo: " + c.getSaldo());
        c.sacar(50);
        System.out.println("Saldo: " + c.getSaldo());
    }
}
```

Chamada a métodos - Sintaxe

- A sintaxe padrão para chamada a métodos é:

  objeto.método(parametro1, parametro2, ...);

Adicionando Métodos

- Vamos adicionar um método para a transferência de uma conta para outra, chamado transferirPara:

```
public class Conta {
    public void transferirPara (Conta destino, double valor){
        destino.saldo = destino.saldo + valor;
        saldo = saldo – valor;
    }
    …
}
```

Exemplo – Transferindo...

```
public class TesteConta{
    public static void main(String[] args){
        Conta c = new Conta();
        c.setSaldo(100);
        Conta c2 = new Conta();
        c.setSaldo(200);
        c.transferirPara(c2, 50);
        System.out.println("Saldo origem: " + c.getSaldo());
        System.out.println("Saldo destino: " + c2.getSaldo());
    }
}
```

Exemplo – Transferindo...

```
public class TesteConta{
    public static void main(String[] args){
        Conta c = new Conta();
        c.setSaldo(100);
        Conta c2 = new Conta();
        c.setSaldo(200);
        c.transferirPara(c2, 50);
        System.out.println("Saldo origem: " + c.getSaldo());
        System.out.println("Saldo destino: " + c2.getSaldo());
    }
}
```

Exercícios

1. Modifique a classe Conta de forma a adicionar os seguintes atributos:
   - Agencia
   - Número da Conta

   Exemplifique o uso destes atributos.
2. Modifique a classe Conta de forma a adicionar os seguintes métodos:
   - depositar, efetuarPagamento, colocarCreditosCelular

Exercícios

3. Crie uma classe Retangulo. A classe possui os atributos altura e largura, cujo valor padrão é 1. Ela possui métodos para calcular a área e o perímetro do retângulo. Devem existir métodos get e set para todos os atributos. Os valores de altura e largura devem estar no intervalo [0, 20]. Escreva um programa para testar a classe Retangulo.

Exercícios

4. Crie uma classe Tempo que guarde as horas, minutos e segundos em atributos separados. Crie métodos get e set para cada um destes atributos e certifique-se que as faixas de valores estão entre as permitidas para cada um deles. Crie uma classe de exemplo para Tempo.
5. Modifique a classe Tempo para que o tempo seja contado em segundos a partir da meia-noite. Mantenha os mesmos métodos get e set. Veja como a classe de exemplo não precisa ser mudada.

Exercícios

6. Modifique a classe Tempo de forma que existam os seguintes métodos:
   - incrementarSegundos(int nr);
   - incrementarMinutos(int nr);
   - incrementarHoras(int nr);

- Após o incremento, a classe tempo deve estar em um estado consistente.
- Forneça também um método toString(), que mostre o valor da hora como uma String

Exercícios

7. Crie uma classe para manter os dados de um professor. Devem ser mantidos os seguintes dados: matrícula, nome, endereço, titulação, área de atuação. Exemplifique o uso da classe.
8. Crie uma classe para manter os dados de um aluno. Devem ser mantidos os seguintes dados: matrícula, nome, endereço, ano de ingresso, semestre de ingresso. Exemplifique o uso da classe.

Exercícios

9. Crie uma classe que mantenha os dados de um curso de graduação. São necessários os seguintes dados: sigla, nome, turno, ingressantes por ano.
10. Crie uma classe que mantenha os dados de uma disciplina de graduação. São necessários os seguintes dados: código, nome, pré-requisitos.