Aulas 2 e 3

RESUMO

Aula 2

Atributos e Métodos de Classe

Atributos Estáticos

-Não faz sentido ter esse valor repetido em todos os objetos, já que ele é único para todos os objetos.
-Devemos aplicar o modificador static na declaração do atributo
-Podemos acessar um atributo de classe através de uma referência de um objeto da classe na qual o atributo foi definido.

```
1  Funcionario f = new Funcionario();
2  // Válido, mas conceitualmente incorreto
3  f.valeRefeicaoDiario = 15;
```

Código Java 6.5: Acessando um atributo de classe

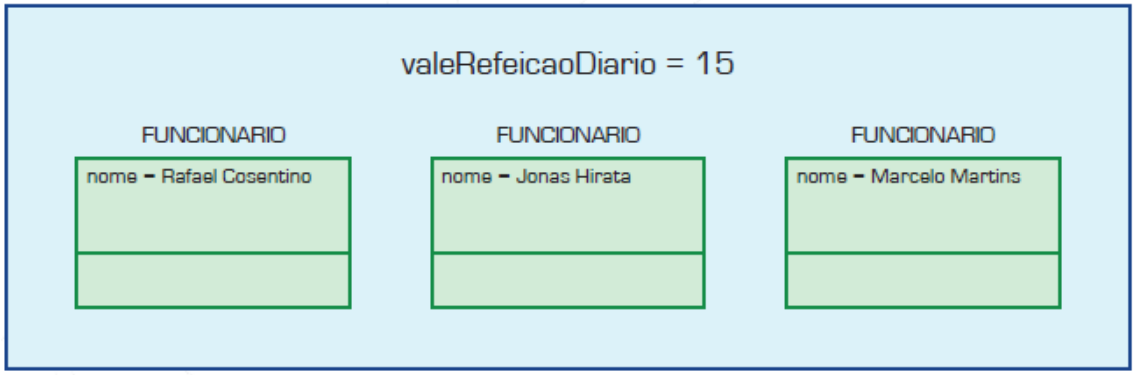

Figura 6.2: Atributos de classe

OBS: Estáticos são valores definidos que é atribuída para todas as classes com o mesmo valor.

Métodos Estáticos

-Definimos métodos para implementar as lógicas que manipulam os valores dos atributos de instância.
-Podemos fazer o mesmo para os atributos de classe.
-Um método de classe deve ser chamado através do nome da classe na qual ele foi definido.

```
1  static void reajustaValeRefeicaoDiario(double taxa) {
2    Funcionario.valeRefeicaoDiario += Funcionario.valeRefeicaoDiario * taxa;
3  }
```

Código Java 6.7: Método que reajusta o valor do vale refeição

---

Aula 4

Encapsulamento

Atributos Privados

-Podemos obter um controle centralizado tornando o atributo salario privado e definindo métodos para implementar todas as lógicas que utilizam ou modificam o valor desse atributo.
-Podemos torná-lo privado, acrescentando o modificador *private*.
-Um atributo privado só pode ser acessado ou alterado por código escrito dentro da classe na qual ele foi definido.
-Definir todos os atributos como privado e métodos para implementar as lógicas de acesso e alteração é quase uma regra da orientação a objetos.
-O intuito é ter sempre um controle centralizado dos dados dos objetos para facilitar a manutenção do sistema e a detecção de erros.

```java
class Funcionario {
  private double salario;

  void aumentaSalario(double aumento) {
    // lógica para aumentar o salário
  }
}
```

Código Java 7.2: Funcionario.java

Métodos Privados

-O papel de alguns métodos pode ser o de auxiliar outros métodos da mesma classe.
-Um atributo privado só pode ser acessado ou alterado por código escrito dentro da classe na qual ele foi definido.
-Definir todos os atributos como privado e métodos para implementar as lógicas de acesso e alteração é quase uma regra da orientação a objetos.
-O intuito é ter sempre um controle centralizado dos dados dos objetos para facilitar a manutenção do sistema e a detecção de erros.

```java
class Conta {
  private double saldo;

  void deposita(double valor) {
    this.saldo += valor;
    this.descontaTarifa();
  }

  void saca(double valor) {
    this.saldo -= valor;
    this.descontaTarifa();
  }

  void descontaTarifa() {
    this.saldo -= 0.1;
  }
}
```

Código Java 7.3: Conta.java

-Para garantir que métodos auxiliares não sejam chamados por códigos escritos fora da classe na qual eles foram definidos, podemos torná-los privados, acrescentando o modificador ***private.***

Implementação e Interface de Uso

-Dentro de um sistema orientado a objetos, cada objeto realiza um conjunto de tarefas de acordo com as suas responsabilidades.
-Por exemplo, os objetos da classe Conta realizam as operações de saque, depósito, transferência e geração de extrato

Por quê encapsular?

-Uma das ideias mais importantes da orientação a objetos é o encapsulamento.
-Encapsular significa esconder a implementação dos objetos.
-O encapsulamento favorece principalmente dois aspectos de um sistema:
Manutenção
Desenvolvimento.
-A manutenção é favorecida pois, uma vez aplicado o encapsulamento, quando o funcionamento de um objeto deve ser alterado, em geral, basta modificar a classe do mesmo.
-Aplicando a ideia do encapsulamento, os atributos deveriam ser todos privados.
-Consequentemente, os atributos não podem ser acessados ou modificados por código escrito fora da classe na qual eles foram definidos.

Acessando ou modificando atributos

-Acessar ou modificar as propriedades de um objeto manipulando diretamente os seus atributos é uma abordagem que normalmente gera problemas.
-Por isso, é mais seguro para a integridade dos objetos e, consequentemente, para a integridade da aplicação, que esse acesso ou essa modificação sejam realizados através de métodos do objeto

Getters e Setters

-Na linguagem Java, há uma convenção de nomenclatura para os métodos que têm como finalidade acessar ou alterar as propriedades de um objeto.
-Segundo essa convenção, os nomes dos métodos que permitem a consulta das propriedades de um objeto devem possuir o prefixo get.
-Analogamente, os nomes dos métodos que permitem a alteração das propriedades de um objeto devem possuir o prefixo set

OBS: Get = consultar, Set = atribuir e alterar valores.

```
1  class Cliente {
2    private String nome;
3
4    public String getNome() {
5      return this.nome;
6    }
7
8    public void setNome(String nome) {
9      this.nome = nome;
10   }
11 }
```

Código Java 7.8: Cliente.java