Encapsulamento e Atributos de Classes em Java

1. Introdução ao Encapsulamento

O encapsulamento é uma das quatro principais características da programação orientada a objetos. Ele envolve o conceito de esconder os detalhes internos de uma classe e expor apenas o que é necessário. O objetivo é proteger os dados da classe e garantir que eles só possam ser alterados de maneira controlada.

Benefícios do Encapsulamento:

- Protege os dados de acessos indevidos.

- Facilita a manutenção do código.

- Permite mudar a implementação interna sem impactar outros componentes.

- Controla o modo como os atributos são acessados ou modificados.

2. Atributos de Classes e Métodos Getters/Setters

Em Java, os atributos de uma classe devem ser privados para seguir o princípio do encapsulamento. O acesso a esses atributos é fornecido por meio de métodos públicos chamados getters (para ler o valor) e setters (para alterar o valor).

Exemplo básico de encapsulamento:

```
public class ContaBancaria {
   private String titular;
   private double saldo;

   public ContaBancaria(String titular, double saldoInicial) {
      this.titular = titular;
      this.saldo = saldoInicial;
   }


   public void setTitular(String titular) {
      this.titular = titular;
   }

  public String getTitular() {
      return titular;
```

```
    }


    public double getSaldo() {
        return this.saldo;
    }

    public void setSaldo(double saldo) {
        if (saldo >= 0) {
            this.saldo = saldo;
        } else {
            System.out.println("O saldo não pode ser negativo.");
        }
    }

    public void depositar(double valor) {
        if (valor > 0) {
            this.saldo += valor;   // saldo= saldo + valor
        } else {
            System.out.println("O valor de depósito deve ser positivo.");
        }
    }

    public boolean sacar(double valor) {
        if (valor <= saldo) {
            saldo -= valor; // saldo = saldo - valor
            return true;
        } else {
            System.out.println("Saldo insuficiente.");
            return false;
        }
    }
}
```

3. Encapsulamento com Validações e Lógica de Negócio

Um dos maiores benefícios do encapsulamento é poder adicionar lógica de negócios na classe para garantir que as operações sejam realizadas corretamente. Por exemplo, ao fazer um saque ou depósito, podemos incluir validações.

Exemplo com mais complexidade:

```
public class ContaBancaria {
    private String titular;
```

```
private double saldo;
private double limiteSaque = 1000.00;

public ContaBancaria(String titular, double saldoInicial) {
   this.titular = titular;
   this.saldo = saldoInicial;
}

public String getTitular() {
   return titular;
}

public void setTitular(String titular) {
   this.titular = titular;
}

public double getSaldo() {
   return saldo;
}

public void setSaldo(double saldo) {
   if (saldo >= 0) {
      this.saldo = saldo;
   } else {
      System.out.println("O saldo não pode ser negativo.");
   }
}

public double getLimiteSaque() {
   return limiteSaque;
}

public void setLimiteSaque(double limiteSaque) {
   if (limiteSaque > 0) {
      this.limiteSaque = limiteSaque;
   }
}

public void depositar(double valor) {
   if (valor > 0) {
      this.saldo += valor;
   } else {
      System.out.println("O valor de depósito deve ser positivo.");
```

```
    }
  }

  public boolean sacar(double valor) {
    if (valor <= saldo && valor <= limiteSaque) {
      saldo -= valor;
      return true;
    } else if (valor > limiteSaque) {
      System.out.println("O valor excede o limite de saque.");
      return false;
    } else {
      System.out.println("Saldo insuficiente.");
      return false;
    }
  }
}
```

4. Exemplo Prático para os Alunos

Criem uma classe Produto com atributos privados como nome, preco, e quantidade. Eles devem implementar métodos para modificar e acessar esses atributos, além de adicionar lógica para verificar se a quantidade não pode ser negativa e se o preço deve ser maior que zero.

Exemplo:

```
public class Produto {
  private String nome;
  private double preco;
  private int quantidade;

  public Produto(String nome, double preco, int quantidade) {
    this.nome = nome;
    setPreco(preco);
    setQuantidade(quantidade);
  }

  public String getNome() {
    return nome;
  }

  public void setNome(String nome) {
    this.nome = nome;
  }
```

```
    public double getPreco() {
        return preco;
    }

    public void setPreco(double preco) {
        if (preco > 0) {
            this.preco = preco;
        } else {
            System.out.println("O preço deve ser maior que zero.");
        }
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        if (quantidade >= 0) {
            this.quantidade = quantidade;
        } else {
            System.out.println("A quantidade não pode ser negativa.");
        }
    }

    public double calcularValorTotal() {
        return preco * quantidade;
    }
}
```

Conclusão

Encapsulamento é um conceito essencial em Java para proteger dados e organizar melhor o código. Ao implementar getters e setters, os desenvolvedores têm maior controle sobre como os dados de um objeto são acessados e modificados. Além disso, adicionar validações dentro desses métodos permite garantir a integridade dos dados.