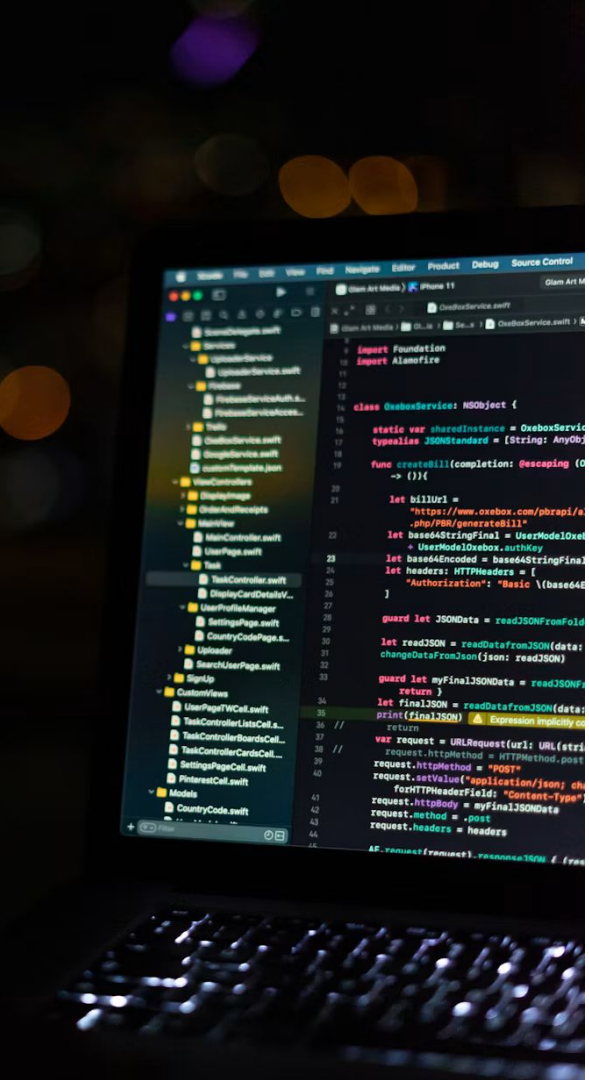

5954018 - Programação Orientada a Objetos

Aula 6 - Manipulação de Objetos: Entrada de Dados, Campos Estáticos e Constantes

Prof. Dr. Denis Martins

DCM | FFCLRP | USP

Objetivos de Aprendizagem | Autoavaliação de progresso

Utilize esta lista como um guia para verificar se você consolidou os conceitos essenciais desta aula.

1. **Entrada de Dados com Scanner**
   Utilizar a classe Scanner para capturar e processar dados informados pelo usuário via console.
2. **Criação Dinâmica de Objetos**
   Criar novos objetos instanciados em tempo de execução utilizando os dados fornecidos pelo usuário.
3. **Atributos de Instância vs. Campos Estáticos**
   Diferenciar e compreender o ciclo de vida e escopo de atributos de instância em relação a campos estáticos.
4. **Declaração de Constantes**
   Definir constantes imutáveis no escopo da classe fazendo uso da combinação de modificadores static final.
5. **Métodos de Instância vs. Métodos Estáticos**
   Identificar e aplicar a diferença conceitual e prática no acesso e chamada de métodos de instância e estáticos.

Créditos: **slides baseados no material de aula original do Prof. Clever Farias.**

Entrada de Dados

Até agora

Os objetos eram instanciados com valores pré-definidos diretamente no código fonte ou passador por argumentos de linha de comando.

```
ContaBancaria conta =
      new ContaBancaria("001", 100.0f);
```

Entrada de Dados

- A classe Scanner permite ler dados diretamente do teclado.
- Pertence ao pacote **java.util**.
- Instanciar objeto Scanner com **new**.

```
import java.util.Scanner;

Scanner scanner =
      new Scanner(System.in);
```

Classe Scanner

Métodos Auxiliares

Essa mesma classe define um conjunto de métodos auxiliares para facilitar a leitura de diferentes tipos de dados. Exemplos:

- Leitura de tipos primitivos numéricos e lógicos.
- Captura de texto formatado ou linhas completas.
- Conversão e tratamento automático de tipos.

```
scanner.nextBoolean(); // boolean
scanner.nextInt(); // int
scanner.nextByte(); // byte
scanner.nextFloat(); // float
scanner.nextLine(); // String (linha)
```

Exemplo Prático

```java
import java.util.Scanner;

/** Lê dados de uma conta. */
public class Principal {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Número: ");
        String numero = scanner.nextLine();

        System.out.print("Saldo: ");
        float saldo = scanner.nextFloat();

        scanner.close();
    }
}
```

Atividade Prática

Tarefa

Modifique o programa ao lado para ler:

- número da conta;
- saldo inicial;
- valor para crédito.

Depois, crie a conta, realize o crédito e exiba o saldo.

```
ContaBancaria conta =
  new ContaBancaria(numero, saldo);

System.out.print("Crédito: ");
float valor = scanner.nextFloat();

conta.creditar(valor);
```

Instâncias

Considere o trecho de código ao abaixo. Alterar **`bobConta`** altera **`adaConta`**?

```
ContaBancaria bobConta =
  new ContaBancaria("001", 100.0f);
ContaBancaria adaConta =
  new ContaBancaria("002", 500.0f);
```

Estado do Objeto

Cada objeto possui seu próprio estado.

Instâncias da mesma classe compartilham a estrutura de atributos, mas mantêm valores independentes na memória.

```
ContaBancaria bobConta =
  new ContaBancaria("001", 100.0f);
ContaBancaria adaConta =
  new ContaBancaria("002", 500.0f);
```

Estado Interno dos Objetos

bobConta : ContaBancaria

numero: "001"
saldo: 100.0f

adaConta : ContaBancaria

numero: "002"
saldo: 500.0f

Escopo de Classe vs. Instância
Embora os saldos sejam individuais, algumas informações podem ser compartilhadas por toda a classe.

E se quisermos que diferentes instâncias de uma mesma classe compartilhem uma mesma informação?

Exemplo: Número de Contas Criadas

```
// Três instâncias são criadas:

ContaBancaria conta1 =
  new ContaBancaria("001", 100.0f);

ContaBancaria conta2 =
  new ContaBancaria("002", 200.0f);

ContaBancaria conta3 =
  new ContaBancaria("003", 300.0f);
```

Quantas contas foram criadas?

- Como armazenar a quantidade total de contas criadas no sistema?
- Onde e como esse valor deve ser incrementado e mantido?

Campos Estáticos

O que são campos estáticos?

Campos estáticos (`static`) em uma classe são campos **compartilhados** por todas as instâncias dessa classe.

- **Pertencem à Classe:** Não pertencem a nenhum objeto individual, mas sim à própria estrutura da classe.
- **Cópia Única:** Apenas um único valor é armazenado e mantido em memória, independente do número de instâncias criadas.

Qualquer modificação feita por uma das instâncias será **refletida imediatamente** em todas as outras instâncias da mesma classe.

```
public class ContaBancaria {
    private String numero;
    private float saldo;
    private static int quantidadeContas = 0;

    public ContaBancaria(
         String numero,
         float saldo) {
            this.numero = numero;
            this.saldo = saldo;
            quantidadeContas++;
    }
}
```

Exercício Rápido

Cenário de Execução

```
// Três contas são criadas:
ContaBancaria conta1 =
  new ContaBancaria("001", 100.0f);
ContaBancaria conta2 =
  new ContaBancaria("002", 200.0f);
ContaBancaria conta3 =
  new ContaBancaria("003", 300.0f);
```

Responda:

> **Quantos valores diferentes de `saldo` existem?**

> **Quantos valores diferentes de `quantidadeContas` existem?**

> **Qual deve ser o valor de `quantidadeContas`?**

Atividade Prática: Poupança (Parte 1)

Estrutura da Classe

Crie a classe **Poupança** contendo os seguintes elementos:

- Uma Poupança possui **número** e **saldo**. O saldo é inicializado com o valor do depósito inicial do correntista.
- **Atributo Estático:** Campo `taxaJuros` para armazenar a taxa de todos os correntistas.
- **Método de Cálculo:** Estimar o valor poupado após um período de **X meses**.
- **Método de Atualização:** Atualizar a taxa de juros comum da classe.

Aplicação de Teste

Escreva um aplicativo (classe com método `main`) para testar a lógica desenvolvida:

Instancie Objetos:

Crie instâncias de contas poupança com depósitos iniciais diferentes.

Valide o Fluxo:

Altere a taxa de juros estática e verifique se o rendimento de todas as contas é atualizado proporcionalmente.

Atividade Prática: Poupança (Parte 2)

Regras de Negócio

A conta Poupança agora deve suportar novas operações financeiras:

- **Aporte (Crédito):** Permite adicionar fundos à conta.
- **Resgate (Débito):** Permite retirar valores da conta.

Restrição de Valor:
O valor máximo para cada operação (aporte ou resgate) é de **R$ 500,00** por transação.

Implementação

Modifique a Classe Poupança:

Atualize a classe desenvolvida anteriormente para incluir os dois novos métodos correspondentes às operações:

- `aportar(valor)`
- `resgatar(valor)`

Garanta que as validações de limite de R$ 500,00 e saldo disponível (para resgates) sejam aplicadas corretamente.

É provável que você tenha feito algo assim:

```
public void aportar(float valor) {
    if (valor <= 500) {
        this.saldo += valor;
    }
}

public void resgatar(float valor) {
    if (valor <= 500 and valor <= this.saldo) {
        this.saldo -= valor;
    }
}
```

Boa Prática: Evite Números Mágicos

O que é?

Um **número mágico** é um valor literal inserido diretamente no código, cujo significado ou propósito não é imediatamente claro.

Problema

Dificulta a leitura do código, prejudica a manutenção e aumenta o risco de erros quando o mesmo valor é repetido em múltiplos pontos.

Solução

Prefira sempre declarar **constantes** com nomes claros e autoexplicativos que revelem o real significado do valor.

Leia Mais: https://denmartins.wordpress.com/2011/08/15/numeros-magicos/ (seu professor era um programador estagiário nesta época)

Constantes

Definição de Constantes

Valores fixos que permanecem imutáveis durante toda a execução do programa.

Declaração em Java

Utilizamos os modificadores `static final` combinados.

```
public static final float LIMITE_OPERACAO = 500.0f;
```

Padrão de Nomenclatura

Por convenção de código, os nomes devem usar `UPPER_SNAKE_CASE`.

Importante!

Qualquer tentativa de modificar uma constante gera um **erro.**

Exemplo

```
public class Poupança {
    public static final float LIMITE_OPERACAO = 500.0f;
    private float saldo;

    public void aportar(float valor) {
        if (valor > 0.0f && valor <= LIMITE_OPERACAO) {
            this.saldo += valor;
        }
    }
}
```

Acesso à Constante

Acesso Direto

Como a constante pertence à classe, realizamos o acesso diretamente utilizando o nome da classe.

Escopo de Classe

Isso deixa claro que o valor é compartilhado e não pertence a uma conta ou instância específica.

```
System.out.println(
    Poupanca.LIMITE_OPERACAO
);
```

Métodos de Instância x Métodos Estáticos

Método de Instância

Operam sobre um objeto específico e dependem do seu estado individual.

Exemplo em Java:

```
bobPoupanca.aportar(100.0f);
```

O método altera o saldo do objeto `bobPoupanca`, dependendo do estado interno daquela instância específica.

Método Estático

Pertencem à classe como um todo, e não a uma instância ou objeto específico.

Exemplo em Java:

```
Poupanca.getQuantidadeContas();
```

São chamados diretamente pelo nome da classe e podem ser executados sem a necessidade de instanciar um objeto.

Exemplo

```java
/**
 * Retorna a quantidade de contas criadas.
 *
 * @return quantidade de contas
 */
public static int getQuantidadeContas() {
    return quantidadeContas;
}
```

Conclusão

Entrada Dinâmica de Dados

O uso de `Scanner` permite criar e manipular objetos em tempo de execução com dados informados diretamente pelo usuário.

Constantes de Classe

A associação `static final` define constantes seguras, estabelecendo valores imutáveis e evitando o antipattern de **números mágicos**.

Membros de Instância

Atributos e métodos de instância pertencem a cada objeto individualmente, definindo seu comportamento e estado interno único.

Membros Estáticos (Static)

Campos e métodos `static` pertencem diretamente à classe, compartilhando o mesmo espaço de memória entre todas as instâncias.

Leitura Recomendada

- **Capítulo 6 do livro**: Deitel, P.; Deitel, H. Java: Como Programar. 10a Edição. Pearson Universidades, 2016.
- Quando Usar Classes e Variáveis Estáticas no Java? Dicas e Exemplos Práticos

Atividade Prática

Em duplas. Instruções em:

https://denmartins.github.io/labs/poo-manipulacao-objetos

Dúvidas e Discussão