Manual de Conceitos Java para Exercícios

**1. Criação de Classes e Atributos**

A criação de classes é fundamental para organizar o código. Uma classe deve conter atributos e métodos que representam o comportamento de seus objetos. Veja um exemplo de uma classe com atributos e métodos simples:

* Exemplo de classe Circulo:

public class Circulo {  
 private double raio;  
  
 // Construtor  
 public Circulo(double raio) {  
 this.raio = raio;  
 }  
  
 // Getter  
 public double getRaio() {  
 return raio;  
 }  
  
 // Setter  
 public void setRaio(double raio) {  
 this.raio = raio;  
 }  
  
 // Método para calcular a área  
 public double calcularArea() {  
 return Math.PI \* Math.pow(raio, 2);  
 }  
}

**2. Herança e Polimorfismo**

Herança é um princípio onde uma classe herda propriedades e métodos de outra. Polimorfismo permite que métodos de uma subclasse sobrescrevam os da classe pai.

* Exemplo de classe Veiculo e Carro:

public class Veiculo {  
 private String modelo;  
 private String marca;  
  
 public Veiculo(String modelo, String marca) {  
 this.modelo = modelo;  
 this.marca = marca;  
 }  
  
 public String getModelo() {  
 return modelo;  
 }  
  
 public String getMarca() {  
 return marca;  
 }  
  
 public void exibirDetalhes() {  
 System.out.println("Modelo: " + modelo + ", Marca: " + marca);  
 }  
}  
  
public class Carro extends Veiculo {  
 private int numeroDePortas;  
  
 public Carro(String modelo, String marca, int numeroDePortas) {  
 super(modelo, marca);  
 this.numeroDePortas = numeroDePortas;  
 }  
  
 @Override  
 public void exibirDetalhes() {  
 super.exibirDetalhes();  
 System.out.println("Número de Portas: " + numeroDePortas);  
 }  
}

**3. Encapsulamento**

Encapsulamento protege os dados de uma classe tornando seus atributos privados e fornecendo métodos públicos para acessá-los e modificá-los.

* Exemplo de encapsulamento em Pessoa:

public class Pessoa {  
 private String nome;  
 private int idade;  
  
 public Pessoa(String nome, int idade) {  
 this.nome = nome;  
 this.idade = idade;  
 }  
  
 public String getNome() {  
 return nome;  
 }  
  
 public void setNome(String nome) {  
 this.nome = nome;  
 }  
  
 public int getIdade() {  
 return idade;  
 }  
  
 public void setIdade(int idade) {  
 this.idade = idade;  
 }  
}

**4. Tratamento de Exceções**

Exceções lidam com erros inesperados no código. Para evitar que o programa falhe, usamos blocos 'try-catch' para capturar e tratar esses erros.

* Exemplo de tratamento de exceção ao ler arquivos:

**5. Laços de Repetição**

Laços de repetição como 'for', 'while', e 'do-while' são usados para executar código repetidamente enquanto uma condição for verdadeira.



**6. Uso de Map e HashMap**

Map e HashMap são estruturas que armazenam pares de chave e valor. Útil para associar, por exemplo, nomes de produtos e suas quantidades no estoque.

* Exemplo de uso de HashMap:



**7. Manipulação de Datas com LocalDate**

LocalDate é uma classe para trabalhar com datas no Java. Ela é útil para fazer cálculos como a diferença entre duas datas ou verificar dias específicos.

* Exemplo de manipulação de datas com LocalDate:



**8. Sobrecarga de Métodos**

Sobrecarga de métodos permite definir múltiplos métodos com o mesmo nome, mas parâmetros diferentes. Isso é útil para quando queremos que o método se comporte de maneira diferente dependendo dos parâmetros.

* Exemplo de sobrecarga de métodos:

