Try Catch

**Exercício 1 - Divisão por Zero**

Crie um programa que solicite ao usuário dois números inteiros e realize a divisão entre eles. Utilize um bloco try-catch para capturar e tratar a exceção de divisão por zero (ArithmeticException), exibindo uma mensagem de erro caso o usuário tente dividir por zero.

import java.util.InputMismatchException;

import java.util.Scanner;

public class ExerciciosTryCatch {

public static void divisaoPorZero() {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Digite o primeiro número inteiro: ");

int num1 = scanner.nextInt();

System.out.print("Digite o segundo número inteiro: ");

int num2 = scanner.nextInt();

int resultado = num1 / num2;

System.out.println("Resultado: " + resultado);

} catch (ArithmeticException e) {

System.out.println("Erro: Divisão por zero não é permitida.");

}

}

**Exercício 2 - Conversão de String para Número**

Crie um programa que peça ao usuário para inserir um número. Use try-catch para capturar e tratar a exceção NumberFormatException caso o usuário insira um valor inválido.

public static void conversaoStringParaNumero() {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Digite um número: ");

String entrada = scanner.nextLine();

int numero = Integer.parseInt(entrada);

System.out.println("Número inserido: " + numero);

} catch (NumberFormatException e) {

System.out.println("Erro: Valor inserido não é um número válido.");

}

}

**Exercício 3 - Raiz Quadrada de um Número**

**Crie um programa que peça ao usuário para inserir um número e calcule sua raiz quadrada. Use try-catch para capturar e tratar a exceção IllegalArgumentException caso o usuário insira um número negativo.**

**public static void raizQuadrada() {**

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

**try {**

**System.out.print("Digite um número para calcular a raiz quadrada: ");**

**double numero = scanner.nextDouble();**

**if (numero < 0) {**

**throw new IllegalArgumentException("Número negativo não permitido.");**

**}**

**System.out.println("Raiz quadrada: " + Math.sqrt(numero));**

**} catch (IllegalArgumentException e) {**

**System.out.println("Erro: " + e.getMessage());**

**}**

**}**

**Exercício 4 - Soma de Dois Números**

**Crie um programa que solicite ao usuário dois números e realize a soma. Utilize try-catch para capturar e tratar a exceção InputMismatchException caso o usuário insira um valor que não seja um número.**

public static void somaDeDoisNumeros() {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Digite o primeiro número: ");

int a = scanner.nextInt();

System.out.print("Digite o segundo número: ");

int b = scanner.nextInt();

System.out.println("Soma: " + (a + b));

} catch (InputMismatchException e) {

System.out.println("Erro: Insira apenas números.");

}

}

**Exercício 5 - Acesso a Índice Inválido em um Array**

Crie um programa que solicite ao usuário um índice e tente acessar um elemento de um array fixo de números inteiros. Utilize try-catch para capturar e tratar a exceção ArrayIndexOutOfBoundsException, caso o usuário informe um índice fora dos limites do array.

public static void acessoIndiceInvalido() {

int[] array = {10, 20, 30, 40, 50};

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Digite o índice que deseja acessar: ");

int indice = scanner.nextInt();

System.out.println("Valor no índice " + indice + ": " + array[indice]);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Erro: Índice fora dos limites do array.");

}

}

**Exercício 6 - Cadastro de Usuário com Validação**

Crie um programa que solicite ao usuário um nome e uma idade. Se o usuário inserir um nome vazio ou uma idade negativa, o programa deve lançar e capturar uma exceção personalizada chamada DadosInvalidosException.

**static class DadosInvalidosException extends Exception {**

**public DadosInvalidosException(String mensagem) {**

**super(mensagem);**

**}**

**}**

**public static void cadastroUsuario() {**

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

**try {**

**System.out.print("Digite o nome: ");**

**String nome = scanner.nextLine();**

**System.out.print("Digite a idade: ");**

**int idade = scanner.nextInt();**

**if (nome.trim().isEmpty() || idade < 0) {**

**throw new DadosInvalidosException("Nome vazio ou idade negativa.");**

**}**

**System.out.println("Usuário cadastrado: " + nome + ", " + idade + " anos.");**

**} catch (DadosInvalidosException e) {**

**System.out.println("Erro: " + e.getMessage());**

**}**

**}**

**Exercício 7 - Calculadora de Média de Notas**

Crie um programa que calcule a média de três notas inseridas pelo usuário. Utilize try-catch para capturar e tratar exceções, como InputMismatchException, caso o usuário insira um valor que não seja um número.

public static void mediaNotas() {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Digite a primeira nota: ");

double n1 = scanner.nextDouble();

System.out.print("Digite a segunda nota: ");

double n2 = scanner.nextDouble();

System.out.print("Digite a terceira nota: ");

double n3 = scanner.nextDouble();

double media = (n1 + n2 + n3) / 3;

System.out.println("Média: " + media);

} catch (InputMismatchException e) {

System.out.println("Erro: Insira apenas números.");

}

}

**Exercício 8 - Verificação de Idade para Maioridade**

Crie um programa que peça a idade de uma pessoa e verifique se ela é maior de idade (18 anos ou mais). Caso o valor inserido seja negativo, lance e trate uma exceção personalizada chamada IdadeInvalidaException.

static class IdadeInvalidaException extends Exception {

public IdadeInvalidaException(String mensagem) {

super(mensagem);

}

}

public static void verificarMaioridade() {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Digite a idade: ");

int idade = scanner.nextInt();

if (idade < 0) {

throw new IdadeInvalidaException("Idade negativa não é permitida.");

}

if (idade >= 18) {

System.out.println("Você é maior de idade.");

} else {

System.out.println("Você é menor de idade.");

}

} catch (IdadeInvalidaException e) {

System.out.println("Erro: " + e.getMessage());

}

}

**Exercício 9 - Conversão de Temperatura com Verificação de Intervalo**

Crie um programa que converta a temperatura de Celsius para Fahrenheit. O programa deve solicitar ao usuário a temperatura em Celsius e, caso a temperatura fornecida seja menor que -273.15 (o zero absoluto), lance uma exceção personalizada chamada TemperaturaInvalidaException. Caso o usuário insira um valor não numérico, o programa deve capturar e tratar a exceção InputMismatchException.

static class TemperaturaInvalidaException extends Exception {

public TemperaturaInvalidaException(String mensagem) {

super(mensagem);

}

}

public static void conversaoTemperatura() {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Digite a temperatura em Celsius: ");

double celsius = scanner.nextDouble();

if (celsius < -273.15) {

throw new TemperaturaInvalidaException("Temperatura abaixo do zero absoluto.");

}

double fahrenheit = (celsius \* 9 / 5) + 32;

System.out.println("Temperatura em Fahrenheit: " + fahrenheit);

} catch (InputMismatchException e) {

System.out.println("Erro: Insira um número válido.");

} catch (TemperaturaInvalidaException e) {

System.out.println("Erro: " + e.getMessage());

}

}

**Exercício 10 - Calculadora de Fatorial com Exceção para Limite de Recursão**

Crie um programa que calcule o fatorial de um número fornecido pelo usuário. O cálculo de fatorial deve ser feito utilizando recursão. Caso o número fornecido seja negativo, o programa deve lançar uma exceção personalizada chamada NumeroNegativoException. Além disso, o programa deve garantir que o número fornecido não seja maior que 20 (para evitar problemas com a recursão e estouro de pilha) e lançar uma exceção LimiteFatorialExcedidoException caso o número seja maior que 20.

static class NumeroNegativoException extends Exception {

public NumeroNegativoException(String mensagem) {

super(mensagem);

}

}

static class LimiteFatorialExcedidoException extends Exception {

public LimiteFatorialExcedidoException(String mensagem) {

super(mensagem);

}

}

public static long fatorial(int n) throws NumeroNegativoException, LimiteFatorialExcedidoException {

if (n < 0) throw new NumeroNegativoException("Número negativo.");

if (n > 20) throw new LimiteFatorialExcedidoException("Número muito grande.");

if (n == 0) return 1;

return n \* fatorial(n - 1);

}

public static void calcularFatorial() {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Digite um número para calcular o fatorial: ");

int num = scanner.nextInt();

long resultado = fatorial(num);

System.out.println("Fatorial de " + num + " é " + resultado);

} catch (NumeroNegativoException | LimiteFatorialExcedidoException e) {

System.out.println("Erro: " + e.getMessage());

} catch (InputMismatchException e) {

System.out.println("Erro: Insira um número válido.");

}

}

}