POO II

Exercício de Exception

```
package exception;

import java.util.InputMismatchException; // EXCEÇÃO GERADA PELOS MÉTODOS DA CLASSE SCANNER
import java.util.Scanner;

public class Exercicio1 {
	public static void main(String args[]) { // PODERIA DEFINIR AQUI UM THROWS EXCEPTION E IRIA COMPILAR MAS NÃO PEGARIA OS ERROS DE ENTRADA DE DADOS ERRADA COM SCANNER
		int dividendo, divisor, quociente;   // VOU DEFIIR O TRY OU NÃO VAI COMPILAR DEVIDO AO MÉTODO RETORNAQUOCIENTE
		try (Scanner in = new Scanner(System.in)) {     // DEFININDO O SCANNER NO TRY NÃO PRECISAMOS FECHAR, ELE FARÁ SOZINHO
			System.out.println("Entre com o dividendo: ");
			dividendo = in.nextInt();  // PODE DAR ERRO AQUI !
			System.out.println("Entre com o divisor: ");
			divisor = in.nextInt();  // PODE DAR ERRO AQUI !
			quociente = LancaException.retornaQuociente(dividendo, divisor);  // AQUI TEM QUE TRATAR OU REPASSAR NA DEFINIÇÃO DO MÉTODO COM THROWS OU NÃO COMPILARÁ
			System.out.println("O valor do quociente é: "+quociente);
		}
		catch (InputMismatchException e) {   // PARA PEGAR ERRO DE ENTRADA DE DADOS
			System.out.println("Valor digitado não numérico!");
		}
		catch (Exception e) {  // PARA PEGAR O ERRO DO MÉTODO RETORNEQUOCIENTE SE OCORRER
			System.out.println(e.getMessage()); // ESSE MÉTODO DA CLASSE EXCEPTION EXIBE A MSG DE ERRO
		}
	}
}


public class LancaException {
	public static int retornaQuociente(int dividendo, int divisor) throws Exception {   // AQUI ELE AVISA QUE NESSE MÉTODO PODE OCORRER UMA EXCEÇÃO DO TIPO EXCEPTION
		if (divisor == 0)
			throw new Exception("O divisor não pode ser igual a zero!");  // DISPARA O ERRO E NÃO TRATA COM TRY. VAI TER QUE DECLARAR THROWS NA DEFINIÇÃO DO MÉTODO ACIMA
		return dividendo / divisor;
	}
}
```