void main(List<String> arguments) {

  // Ler o console do usuario

  print("Digite um numero: ");

  var line = stdin.readLineSync(encoding: utf8);

  // Como estamos fazendo aqui: Line e diferente de null e nao esta vazio?

  // Se a condição for verdade usamos o proprio line para colocar dentro da variavel

  // Se a condição for falsa colocamos o "0"

  double numero = double.parse(line != null && line.isNotEmpty ? line : "0"); // Convertendo para um double

  print(numero);

}

Aqui se o usuario escrever numeros, vai da certo, mas se ele digitar letras, vamos ter problemas

Então para isso vamos fazer um try e catch

Agora dessa forma esta corrigido

void main(List<String> arguments) {

  // Ler o console do usuario

  print("Digite um numero: ");

  var line = stdin.readLineSync(encoding: utf8);

  try {

    // TENTE CONVERTER PARA UM DOUBLE

    // Como estamos fazendo aqui: Line e diferente de null e nao esta vazio?

    // Se a condição for verdade usamos o proprio line para colocar dentro da variavel

    // Se a condição for falsa colocamos o "0"

    double numero = double.parse(

      line != null && line.isNotEmpty ? line : "0",

    ); // Convertendo para um double

    print(numero);

  } catch (e) {

    // SE NAO CONSEGUIR CAIMOS AQUI.

    print("Número invalido: $line ");

  }

}

O bloco finally vai executar independente se deu erro ou não

void main(List<String> arguments) {

  // Ler o console do usuario

  print("Digite um numero: ");

  var line = stdin.readLineSync(encoding: utf8);

  try {

    // TENTE CONVERTER PARA UM DOUBLE

    // Como estamos fazendo aqui: Line e diferente de null e nao esta vazio?

    // Se a condição for verdade usamos o proprio line para colocar dentro da variavel

    // Se a condição for falsa colocamos o "0"

    double numero = double.parse(

      line != null && line.isNotEmpty ? line : "0",

    ); // Convertendo para um double

    print(numero);

  } catch (e) {

    // SE NAO CONSEGUIR CAIMOS AQUI.

    print("Número invalido: $line ");

  } finally {

    print("Executando independente se deu erro ou não");

  }

}

Em **Dart**, o bloco finally é usado em conjunto com blocos try e catch para **garantir que um trecho de código seja executado independentemente de ocorrer ou não uma exceção** (erro) durante a execução.

**Sintaxe:**

### Explicação:

* O **try** é usado para envolver o código que pode lançar (gerar) uma exceção.
* O **catch** trata a exceção, se ela ocorrer.
* O **finally** **sempre** é executado, mesmo que:
  + uma exceção tenha sido lançada;
  + o try tenha sido concluído com sucesso;
  + o catch tenha lidado com a exceção;
  + tenha um return dentro do try ou catch.

void main() {

try {

int resultado = 10 ~/ 0; // Divisão por zero lança uma exceção

print("Resultado: $resultado");

} catch (e) {

print("Ocorreu um erro: $e");

} finally {

print("Este bloco sempre será executado");

}

}

SAIDA:

Ocorreu um erro: IntegerDivisionByZeroException

Este bloco sempre será executado

**Para que serve?**

* Liberar recursos (como fechar arquivos, conexões com banco de dados, etc.)
* Garantir que determinada lógica sempre seja executada (por exemplo, mostrar uma mensagem ou limpar memória).

Se quiser, posso montar um exemplo com arquivos, rede ou banco de dados para ilustrar o uso do finally num contexto real.