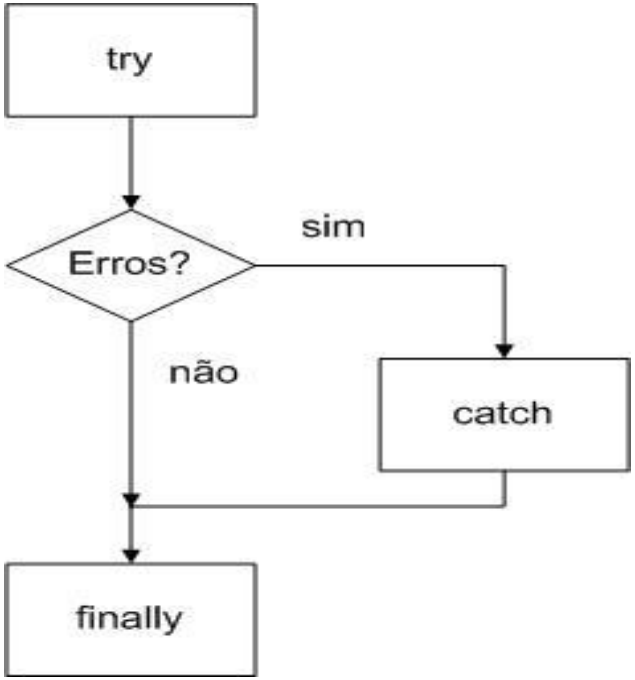
Tratamento de Exceções

Linguagem Java

Exceções

- Erros que podem acontecer durante a execução de um programa.
- Devem ser tratados (contornados) de modo a evitar que o programa termine de modo anormal!
- A estrutura **try-catch-finally** tem esta finalidade aliada a cláusula **throws**.

Uso da estrutura try-catch-finally



Uso da estrutura try-catch-finally:

- Qualquer instrução, ou bloco de instruções, que possam gerar uma exceção devem ser reunidos num bloco try.
- Um bloco try pode ter um ou vários blocos catch;
- Havendo um bloco try, deve haver pelo menos um bloco catch, embora vários sejam permitidos.

Uso da estrutura try-catch-finally:

- O bloco finally é opcional, mas se ocorrer deve ser único.
  - Um bloco finally ele será executado independentemente de ter ocorrido erro ou não.
- Se uma exceção é detectada num bloco try, o programa será imediatamente desviado para o bloco catch correspondente à exceção gerada.

Uso da estrutura try-catch-finally

- As exceções são nomeadas segundo o seu tipo:

NumberFormatException (erros de formato de dados)
ArithmeticException (divisão por zeros entre inteiros)
IOException (erros de E/S de dados)
ArrayIndexOutOfBounds (indexação fora dos limites do vetor)
InterruptedException (erro de interrupção)

Exception (exceção genérica, isto é, não particularizada)

Estrutura try-catch-finally

```
   try{
          <bloco de instruções>
       }
       catch (<nome da exceção 1>){
          <tratamento da exceção 1>
       }
...
        catch (<nome da exceção n>) {
      <tratamento da exceção n>
       }
       finally{
          <instruções finais>
     }
```

Exemplo:

```
import  javax.swing.*;
public class Ex_try_catch_finally{
    public static void main(String args[]){
       int n1, n2;
        String
aux=JOptionPane.showInputDialog(null,
                "Informe um valor inteiro");
       try{
         //transforma em inteiro
          n1=Integer.parseInt(aux);
          aux=JOptionPane.showInputDialog(null,
               "Informe outro valor inteiro");
          n2=Integer.parseInt(aux); //idem
          JOptionPane.showMessageDialog(null,
                       "Produto = "  + (n1*n2));
           JOptionPane.showMessageDialog(null,
               "Divisão = "  + (n1/n2));
}
```

```
 catch (NumberFormatException e) {
  //trata erros de formato de numeros

  JOptionPane.showMessageDialog(null,
         "Erro na entrada de dados");
}
      catch (ArithmeticException e) {
           //Divisão por zero

  JOptionPane.showMessageDialog(null,
         "Divisão por zero");          }
      finally {

  JOptionPane.showMessageDialog(null,
         "****Final de execução****");
      }
   }
}
```

Uso da cláusula ***throws*** *(repasse)*

- Certas operações requerem tratamento de exceção, senão o programa nem compila.
- O correto seria inserir no programa um bloco try-catch, mas caso não queira realizar este tratamento, deve-se explicitar através da cláusula **throws**.

Uso da cláusula *throws*

```
import java.io.*;
class IgnoraErro{
    public static void main(String args[]) throws IOException { //descarta o erro
      BufferedReader dado; //tipo de dado que será buferizado durante a leitura
      System.out.println("entre com seu nome");
      dado=new BufferedReader(new InputStreamReader(System.in));
      System.out.println("Seu nome eh " + dado.readLine());
     }
}
```

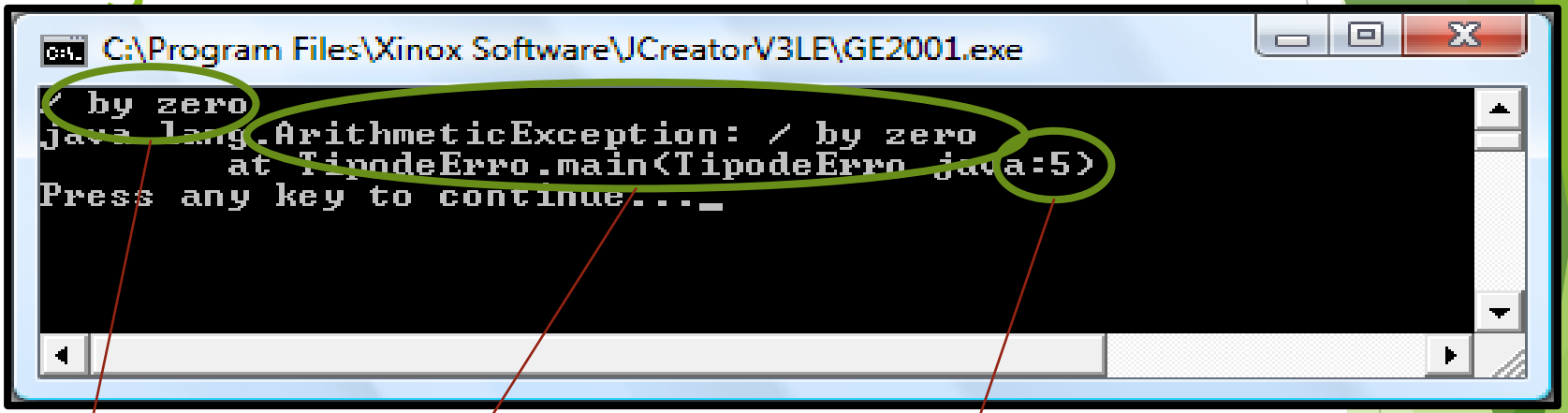
Uso do **getMessage** e **printStackTrace**

- Quando precisamos saber qual foi o erro e em que linha ocorreu podemos recorrer a este 2 métodos.
  - O método **getMessage** retorna uma string informando a exceção ocorrida
  - O método printStackTrace retorna o tipo de exceção e em que linha do programa ocorreu

Exemplo

```
1.class TipodeErro {
2.  public static void main(String args[]){
3.    int x=10, y=0, z=0;
4.    try{
5.          z=x/y; // gera uma arithmetic exception (divisão por zero)
6.    }
7.    catch(Exception erro) { //exceção genérica!!!
8.       System.out.println(erro.getMessage()); //mostra a mensagem de erro
9.       erro.printStackTrace(); //mostra a exceção e a linha onde ocorreu
10.         }
11.    }
12. }
```

Resultado da execução do programa



Linha do programa onde ocorreu o erro

Tipo de erro

Mensagem de erro (getMessage)