

# Linguagem de Programação

---

ADS – Análise e Desenvolvimento de Sistemas  
Prof. Vagner Macedo

Aula 15  
Tratamento de Exceções

# Definição

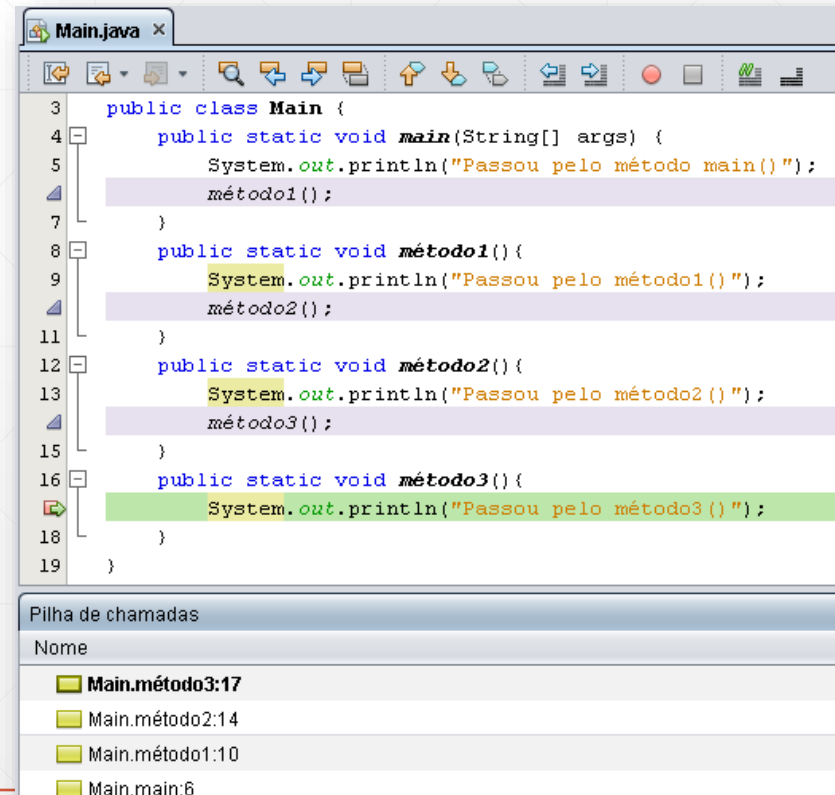
- Evento que ocorre durante a execução de um programa, que interrompe o fluxo normal de instruções.

# Funcionamento

- Quando uma exceção ocorre em um método, o método cria um objeto que contém informações para manipular a exceção:
  - Tipo de Exceção;
  - Estado do programa;
  - Mensagem;
  - Exceção Ligada.
- Ao disparar uma exceção, o sistema procura na pilha de chamada de métodos o primeiro método que previu e capturou essa exceção.

# Pilha de Chamada de Métodos (Call Stack)

- Essa pilha representa as chamadas internas de métodos do programa
- Quando estamos depurando um programa na IDE, podemos visualizar essa pilha em um ponto de interrupção através da janela “Pilha de Chamadas”.

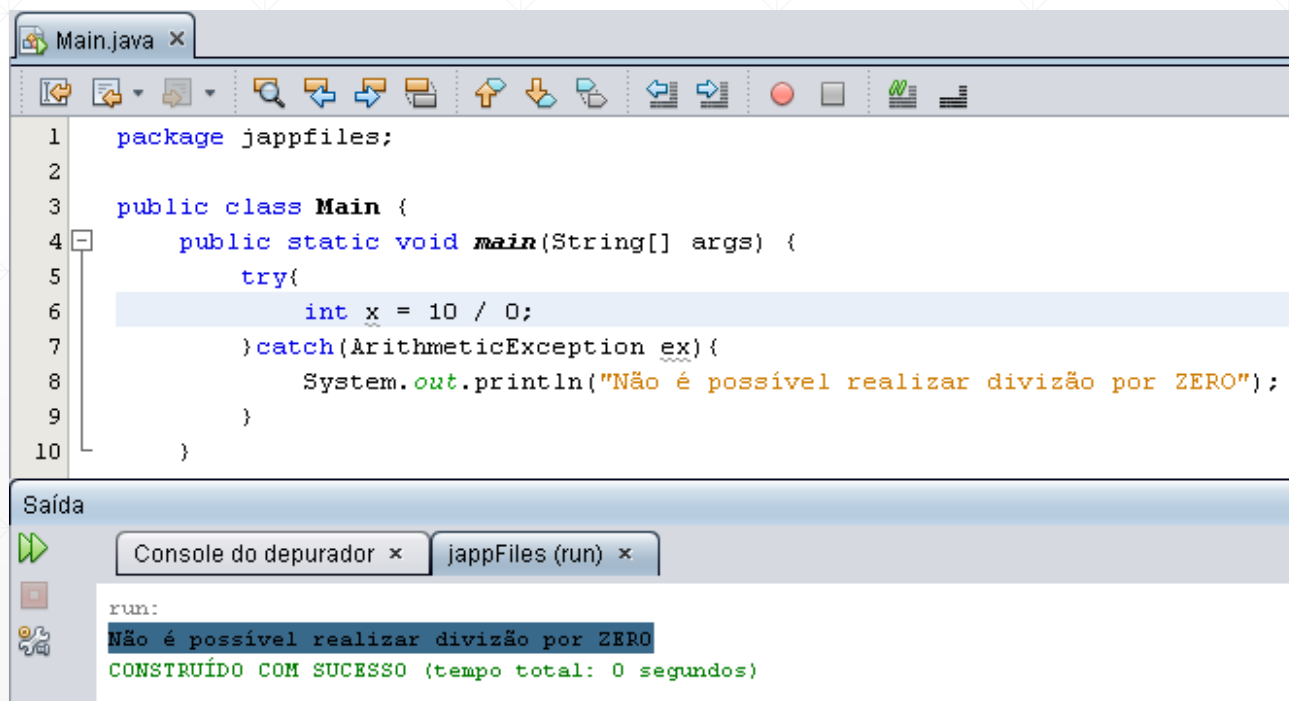


# Manipulador de Exceção

- Para capturar uma exceção, utilizamos um manipulador de exceção
- Ao encontrar na pilha de chamadas um método com um manipulador de exceção, o sistema passa ao manipulador o objeto criado na exceção
- Se nenhum manipulador de exceção for encontrado na pilha de chamadas de métodos, a aplicação é encerrada

# A instrução try-catch

- A instrução permite capturar uma exceção e executar um bloco de instruções caso ela aconteça



```
1 package jappfiles;
2
3 public class Main {
4     public static void main(String[] args) {
5         try{
6             int x = 10 / 0;
7         } catch (ArithmeticException ex) {
8             System.out.println("Não é possível realizar divisão por ZERO");
9         }
10    }
```

Saída

Console do depurador x jappFiles (run) x

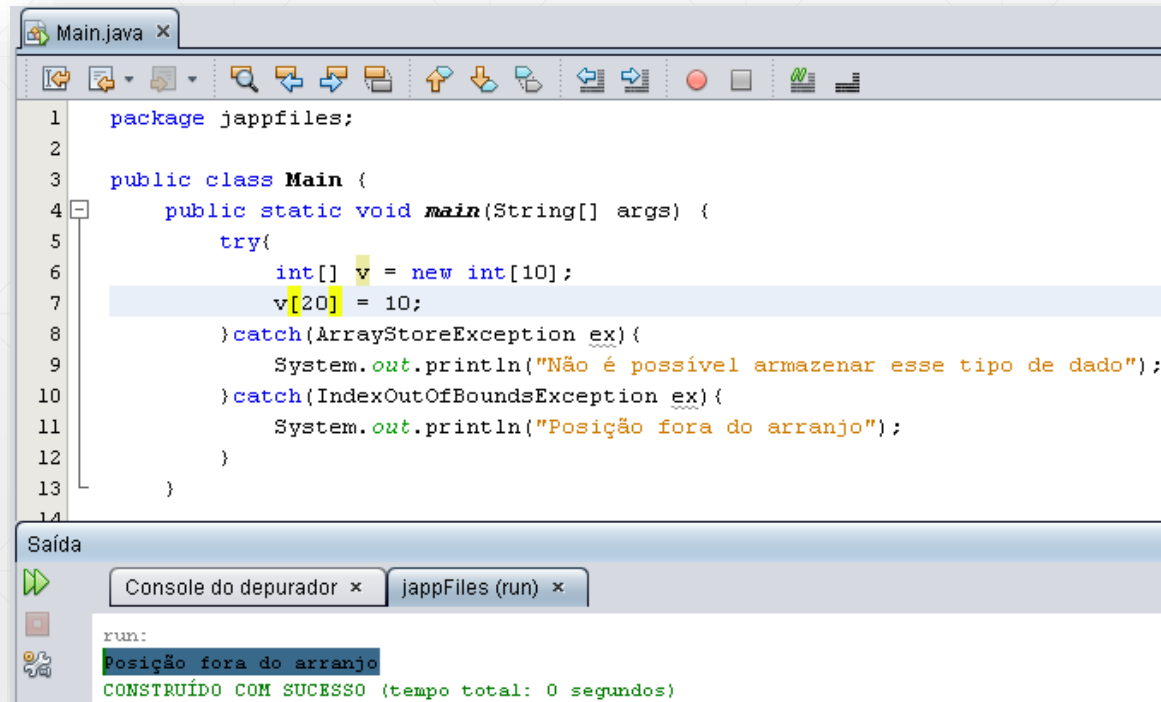
run:  
Não é possível realizar divisão por ZERO  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

# Sintaxe do try-catch

```
try{  
    //instruções que devem ser monitoradas, pois  
    //podem disparar uma exceção  
}catch(Exception e){  
    //instruções que devem ser executadas se  
    //ocorrer uma exceção  
}finally{  
    //instruções que devem ser executadas sempre  
}
```

# O bloco “catch”

- Podemos utilizar vários blocos “catch” para o mesmo “try”.



The screenshot shows an IDE window titled 'Main.java' with the following Java code:

```
1 package jappfiles;
2
3 public class Main {
4     public static void main(String[] args) {
5         try{
6             int[] v = new int[10];
7             v[20] = 10;
8         }catch(ArrayStoreException ex){
9             System.out.println("Não é possível armazenar esse tipo de dado");
10        }catch(IndexOutOfBoundsException ex){
11            System.out.println("Posição fora do arranjo");
12        }
13    }
14 }
```

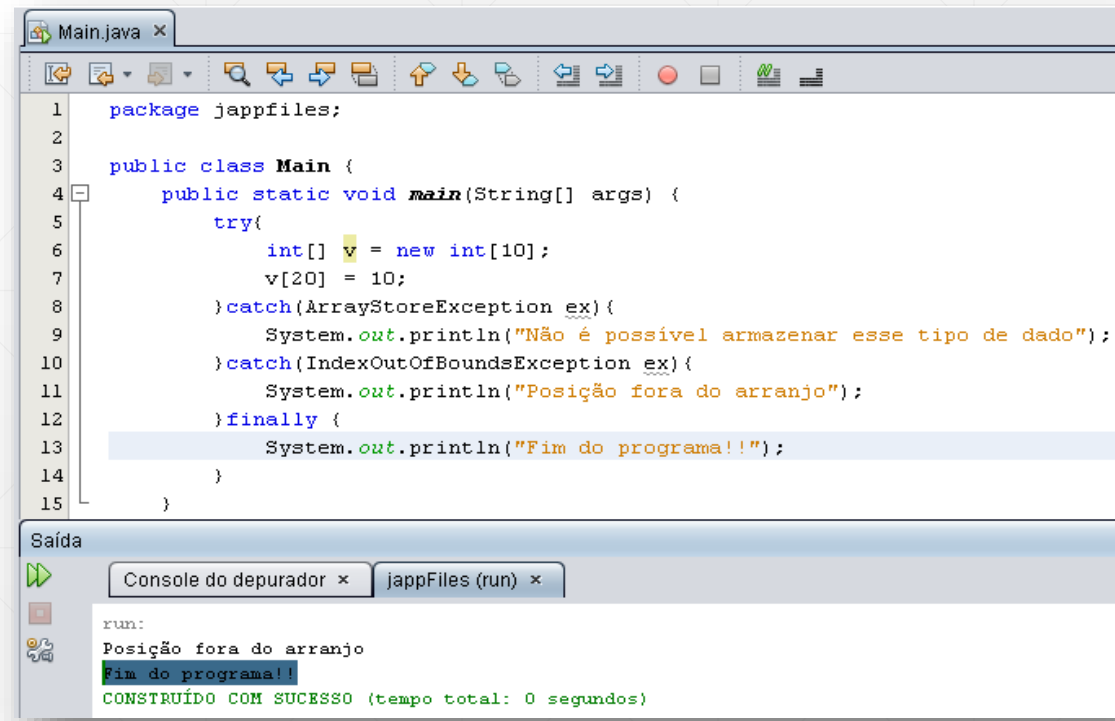
Below the code editor is a 'Saída' (Output) window. It contains two tabs: 'Console do depurador x' and 'jappFiles (run) x'. The 'jappFiles (run) x' tab is active, showing the output of the program:

```
run:
Posição fora do arranjo
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```



# O bloco “finally”

- O bloco “finally” é utilizado para que tenhamos instruções que sempre devem ser executadas, ocorrendo exceção ou não.



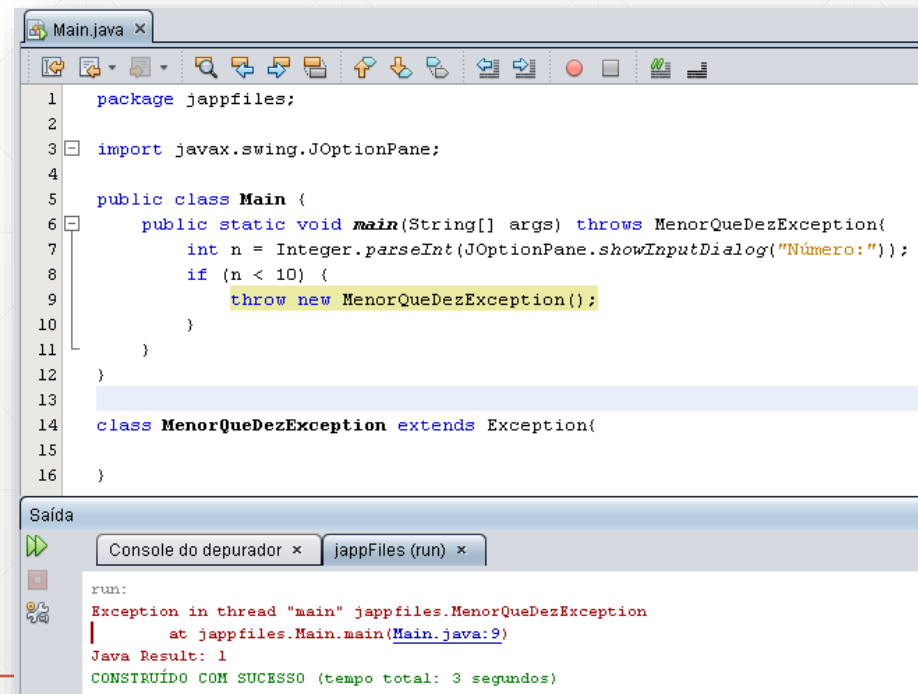
```
1 package jappfiles;
2
3 public class Main {
4     public static void main(String[] args) {
5         try{
6             int[] v = new int[10];
7             v[20] = 10;
8         }catch(ArrayStoreException ex){
9             System.out.println("Não é possível armazenar esse tipo de dado");
10        }catch(IndexOutOfBoundsException ex){
11            System.out.println("Posição fora do arranjo");
12        }finally {
13            System.out.println("Fim do programa!!");
14        }
15    }
16 }
```

Saída

run:  
Posição fora do arranjo  
Fim do programa!!  
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

# Disparando exceções

- Se quisermos “forçar a barra” e disparar uma exceção propositalmente no programa, podemos utilizar o comando “throws” como uma instrução.



```
1 package jappfiles;
2
3 import javax.swing.JOptionPane;
4
5 public class Main {
6     public static void main(String[] args) throws MenorQueDezException{
7         int n = Integer.parseInt(JOptionPane.showInputDialog("Número:"));
8         if (n < 10) {
9             throw new MenorQueDezException();
10        }
11    }
12 }
13
14 class MenorQueDezException extends Exception{
15
16 }
```

Saída

Console do depurador x jappFiles (run) x

```
run:
Exception in thread "main" jappfiles.MenorQueDezException
|   at jappfiles.Main.main(Main.java:9)
Java Result: 1
CONSTRUÍDO COM SUCESSO (tempo total: 3 segundos)
```

**Obrigado!**

