



# Noções Básicas de Java

8-3

Tratamento de Exceções

**ORACLE**  
Academy



Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

# Objetivos

- Esta lição abrange os seguintes objetivos:
  - Explicar a finalidade do tratamento de exceções
  - Tratar exceções com uma construção try/catch
  - Descrever exceções comuns lançadas em Java



# Tópicos

- **O que É uma Exceção?**
- Tratamento de Exceções com o Bloco try/catch
- Exceções Comuns



**ORACLE**  
Academy

JFo 8-3  
Tratamento de Exceções

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

4

# O que É uma Exceção?



- Para compreender o tratamento de exceções, você precisa primeiro compreender o que é uma exceção
- Uma exceção é um erro que ocorre durante a execução de um programa (run-time) que interrompe o fluxo normal do programa Java
- No entanto, você pode lidar com essas condições dentro do seu programa e tomar as ações corretivas necessárias para que o programa continue a ser executado (tratamento de exceções)

## Por que Devo Tratar Exceções?

- Se uma exceção ocorrer enquanto seu programa está sendo executado...
- A execução do programa é terminada
- Um rastreamento de pilha (stack trace), com detalhes da exceção, é impresso no console

## Quando Você Não Trata Exceções: Exemplo

- No Java, o código a seguir lança uma exceção porque você não pode dividir um número inteiro por zero:

```
1 public class ExceptionHandling {  
2  
3     public static void main(String args[]) {  
4         int d = 0;  
5         int a = 10 / d;  A exceção ocorre nesta instrução  
6         System.out.print(a);  Esta instrução não é executada  
7     } //fim do método main  
8 } //fim da classe ExceptionHandling
```

- Um rastreamento de pilha, com detalhes da exceção, é impresso no console
- A execução do programa termina na linha 4 e, portanto, a instrução na linha 5 não é executada

Neste exemplo, o rastreamento de pilha a seguir é impresso:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero at  
com.example.ExceptionHandling.main(ExceptionHandling.java:4)
```

## Quando Você Não Trata Exceções

- Quando o Java encontra um erro ou uma condição que faz com que a execução não continue normalmente, o Java "lança" uma exceção
- Se a exceção não for "pega" pelo programador, o programa travará
- A descrição da exceção e o rastreamento da pilha atual são impressos no console



# Tratando Exceções

- Uma maneira de tratar exceções é, em primeiro lugar, simplesmente evitá-las
- Por exemplo, evite uma `ArithmeticException` usando uma lógica condicional: Antes de tentar executar uma operação potencialmente arriscada, teste para ver se a condição ocorrerá

```
int divisor = 0;  
if(divisor == 0){  
    System.out.println("Não pode ser zero!");  
}  
else {  
    System.out.println(5 / divisor);  
} //fim if
```

# Categorias de Exceção

- As exceções do Java enquadram-se em duas categorias:
- Exceções Verificadas:
  - O compilador verifica e trata as exceções
  - Se as exceções não forem tratadas no programa, ele produzirá um erro de compilação
  - Exemplos:
    - `FileNotFoundException`, `IOException`
- Exceções Não Verificadas:
  - O compilador não verifica nem trata as exceções
  - Exemplos:
    - `ArrayIndexOutOfBoundsException`,  
`NullPointerException`, `ArithmeticException`



## Exercício 1

- Importe e abra o projeto `ExceptionsEx`
- Examine `ExceptionEx1.java`:
  - Execute o programa e observe a saída:
  - `ArrayIndexOutOfBoundsException` ocorre
  - É uma prática interessante tratar a exceção desse programa?
  - Modifique o programa para calcular a soma da matriz

# Tópicos

- O que É uma Exceção?
- **Tratamento de Exceções com o Bloco try/catch**
- Exceções Comuns



**ORACLE**  
Academy

JFo 8-3  
Tratamento de Exceções

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

12

## Tratamento de Exceções com o Bloco try/catch

- Mas nem todas as exceções podem ser evitadas porque, antes de chamar uma operação, nem sempre você sabe se ela falhará
- Outra estratégia é usar o bloco try/catch para o tratamento de exceções

## Entendendo o Bloco try/catch

- No caso de um código que tende a causar uma exceção, você pode escrevê-lo dentro de um bloco "try" especial
- Você associa os handlers de exceção com um bloco try fornecendo um ou mais blocos catch depois do bloco try
- Cada bloco catch trata o tipo de exceção indicado por seu argumento
- O argumento `ExceptionType` declara o tipo de exceção

## Controle de Fluxo em Blocos try/catch: Sucesso

- Se o bloco try for bem-sucedido, nenhuma exceção ocorrerá

```
try {  
    // um código arriscado que tende a  
    // causar uma exceção  
}  
catch(ExceptionType ex) {  
    // código de tratamento de exceções  
}  
2 ➤ System.out.println("Conseguimos");
```

Primeiro o bloco try é executado e, em seguida, é executado o código depois do bloco catch

O controle de fluxo ignora o bloco `catch`. A execução continua com o restante do código fora do bloco `catch`.

# Controle de Fluxo em Blocos try/catch: Falha

- Se o bloco try falhar, uma exceção ocorrerá

```
try {  
1 // um código arriscado que tende a  
  // causar uma exceção  
}  
2 catch(ExceptionType ex) {  
  //código de tratamento de exceções  
}  
3 System.out.println("Conseguimos");
```

O bloco try é executado, a exceção ocorre e o restante desse bloco não é executado

O bloco catch é executado; em seguida, o restante do código é executado

O controle de fluxo move imediatamente para o bloco `catch`. Quando o bloco `catch` é concluído, a execução do restante do código continua.



# Controle de Fluxo em blocos try/catch: Exemplo

```
1 public static void main(String args[]) {
2     int a = 100, res;
3     try{
4         System.out.println("Informe o valor de b");
5         Scanner console = new Scanner(System.in);
6         int b = console.nextInt();
7         System.out.println("Informe o valor de c");
8         int c = console.nextInt();
9         res = 10 / (b - c);
10        System.out.println("O resultado é " + res);
11    }
12    catch(Exception e){
13        String errMsg = e.getMessage();
14        System.out.println(errMsg);
15    } //fim try catch
16    System.out.println("Depois do bloco catch");
17 } //fim do método main
```

**ORACLE**  
Academy

JFo 8-3  
Tratamento de Exceções

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

17

Neste exemplo, um bloco `try/catch` foi adicionado para pegar `ArithmeticException`. O exemplo ilustra o fluxo de programa quando a exceção é tratada com `try/catch`.

`ArithmeticException` ocorre na linha 9.

- O controle passa imediatamente para o bloco `catch`.
- A Instrução nº 10 no bloco `try` não é executada.
- Em vez disso, são executadas as instruções no bloco `catch`.
- A execução do programa continua com a instrução fora do bloco `catch`, e a mensagem "Depois do bloco `catch`" é exibida no console.

# Tópicos

- O que É uma Exceção?
- Tratamento de Exceções com o Bloco try/catch
- **Exceções Comuns**



**ORACLE**  
Academy

JFo 8-3  
Tratamento de Exceções

Copyright © 2020, Oracle e/ou suas empresas afiliadas. Todos os direitos reservados.

18

## Exemplos de Exceções

- `java.lang.ArrayIndexOutOfBoundsException`
  - Tentativa de acessar um índice de matriz não existente
- `java.lang.NullPointerException`
  - Tentativa de usar uma referência de objeto que não foi instanciada
- `java.io.IOException`
  - Operações de E/S com falha ou interrompidas

Veja a seguir algumas das exceções que o Java pode lançar. É provável que você tenha visto uma ou mais dessas exceções quando trabalhou nos exercícios desta classe.

# Entendendo Exceções Comuns

- Exceções Não Verificadas - devido a erro de programação:
  - Exemplo: Exceção `ArrayIndexOutOfBoundsException`

```
01 int[] intArray = new int[5];  
02 intArray[5] = 27;
```

- Rastreamento de pilha:

```
Exception in thread "main"  
    java.lang.ArrayIndexOutOfBoundsException: 5  
        at TestErrors.main(TestErrors.java:17)  
)
```

Este código mostra um erro comum cometido ao acessar uma matriz. Lembre-se de que as matrizes são baseadas em zero (o primeiro elemento é acessado por um índice zero). Por isso, em uma matriz com cinco elementos, o último elemento na verdade é `intArray[4]`.

`intArray[5]` tenta acessar um elemento que não existe. O Java responde a esse erro de programação lançando uma `ArrayIndexOutOfBoundsException` e o rastreamento da pilha é impresso no console.

Como o acesso a um índice inválido na matriz é uma exceção não verificada, você não precisa tratar a exceção com o bloco `try/catch`.

# Identificando NullPointerException

- Essa exceção não verificada é lançada quando um aplicativo tenta usar nulo quando um objeto é obrigatório
- São elas:
  - Chamar o método de instância de um objeto nulo
  - Acessar ou modificar o campo de um objeto nulo

Chamar o método length em um objeto nulo

```
public static void main(String[] args) {  
  
    String name = null;  
    System.out.print("Comprimento da string " + name.length());  
  
} //fim do método main
```

NullPointerException é lançada porque um método está sendo chamado em um valor nulo.

# Identificando IOException

```
public static void main(String[] args) {  
  
    try {  
        File testFile = new File("//testFile.txt");  
        testFile.createNewFile();  
        System.out.println("testFile exists:"  
                             + testFile.exists());  
    }  
    catch (IOException e) {  
        System.out.println(e);  
    }  
} //fim try catch  
} //fim do método main
```

O exemplo do slide está tratando a exceção que possivelmente foi gerada:

- Lançando a exceção do método `testCheckedException`
- Pegando a exceção no método caller

Neste exemplo, o bloco `catch` pega a exceção porque o caminho para o arquivo de texto não está formatado corretamente. `System.out.println(e)` chama o método `toString` da exceção, e o resultado é `java.io.IOException`. Ou seja, a sintaxe do nome do arquivo, do nome do diretório ou do rótulo de volume está incorreta.

# Melhores Práticas para Tratar Exceções

- Tente ser o mais específico possível com o tipo de erro que você está tentando pegar
- Assim o programa poderá fornecer um feedback específico sobre o que deu errado
- Em geral, pegar uma exceção genérica é uma ação muito imprecisa para ser útil, mas isso pode ser feito em último caso

```
catch (Exception e) {  
    System.out.println(e);  
}
```

# Exemplo de uma Prática Ruim

```
public static void main(String[] args) {  
  
    try {  
        File testFile = new File("//testFile.txt");  
        testFile.createNewFile();  
        System.out.println("testFile exists:"  
                             + testFile.exists());  
    }  
    catch (Exception e) { Pegando uma exceção  
        System.out.println("Error Creating File");  
    } //end try catch  
} //fim do método main
```

Não há processamento da classe de exceção?

O código no slide ilustra duas práticas inadequadas de tratamento de exceções.

1. A cláusula `catch` pega um tipo `Exception`, em vez de um tipo `IOException`.
2. A cláusula `catch` não analisa o objeto `Exception`. Em vez disso, ela simplesmente assume que a exceção esperada foi lançada do objeto `File`.

Como resultado desse estilo de programação descuidado, o código imprime a seguinte mensagem no console:

```
Problema ao criar o arquivo!
```

A mensagem sugere que o arquivo não foi criado e, na verdade, qualquer código adicional no bloco `catch` será executado. Mas o que realmente está acontecendo no código?



# Uma Prática um Tanto Quanto Melhor

```
public static void main(String[] args) {  
    try {  
        File testFile = new File("//testFile.txt");  
        testFile.createNewFile();  
        System.out.println("testFile exists:"  
                             + testFile.exists());  
    }  
    catch (IOException e) {  
        System.out.println(e);  
    }  
} //fim try catch  
} //fim do método main
```

Pegando a exceção específica

toString() é chamado neste objeto

O código ilustra duas boas práticas de tratamento de exceções:

1. A cláusula `catch` pega um tipo `IOException`.
2. A cláusula `catch` imprime os detalhes da exceção no console.



## Exercício 2

- Importe e abra o projeto `ExceptionsEx`
- Examine `Calculator.java` e `ShoppingCart.java`
- Modifique os programas para implementar o tratamento de exceções:
  - `Calculator.java`:
    - Identifique a exceção que poderia ocorrer
    - Altere a assinatura do método `divide` para indicar que ele lança uma exceção
  - `ShoppingCart.java`:
    - Pegue a exceção na classe que chama o método `divide`

# Resumo

- Nesta lição, você deverá ter aprendido a:
  - Explicar a finalidade do tratamento de exceções
  - Tratar exceções com uma construção try/catch
  - Descrever exceções comuns lançadas em Java



