Tratamento de Exceções

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Abril de 2019

Conceito

- **Robustez**: capacidade de um *software* continuar seu processamento mesmo sob condições anormais;
- Problemas: Como prevenir-se de erros em tempo de execução? Como prevenir-se de situações anormais?
- Exemplos de situações anormais:
  - Divisão por zero, fim de arquivo, *overflow*, utilização de um objeto não instanciado, acesso a um índice inválido de um vetor.

Exceção

- É um evento ocorrido durante a execução normal de um programa que desvia o fluxo normal de execução;

- É uma condição provocada por uma situação excepcional, que requer uma ação específica imediata.

(Varejão, 2004)

Mecanismo para Tratar Exceções

- Existem linguagens de programação que não possuem tais mecanismo. Como por exemplo:
  - Linguagem C e
  - Linguagem Pascal;
- Neste caso o programador deve produzir código para fazer o tratamento das possíveis exceções.
- Exemplo:

```
if(x != 0)
   res = y / x;
else
   printf("Erro de divisão por zero!");
```

Mecanismo para Tratar Exceções

- Por outro lado, existem linguagens que possuem tais mecanismos. Como:
  - Delphi,
  - C#,
  - Java e etc.;
- Benefícios:
  - Diminuição da sobrecarga do texto do programa;
  - Maior comodidade, para o programador, no tratamento das condições anormais.

Exemplo de Exceções em Java

- **`NullPointerException`** – ocorre quando a operação realizada gera um acesso a uma área não alocada (nula);

- **`IndexOutOfBoundException`** – ocorre quando se tenta acessar um índice de um vetor que está fora de seus limites.

Lançamento de Exceções

- O lançamento (ou sinalização) de exceções pode ser realizado da seguinte forma:
  - **Automaticamente**: o próprio mecanismo existente na linguagem é o responsável por lançar a exceção. Pode ocorrer em qualquer ponto do programa passível de geração de situação anormal.
    - Exemplo: o usuário do *software* entra com um valor do tipo string num campo que espera um valor do tipo float.
  - **Explicitamente**: o programador escreve o código para lançar uma possível exceção e consequentemente capturá-la.

Lançamento de Exceções

- Em casos específicos um programador pode fazer o lançamento de exceções que não são tratadas pela JVM;
- O exemplo a seguir mostra um lançamento de exceção explícito em Java. Para tal, emprega-se o comando **`throw`**;

```
throw new Exception();
```

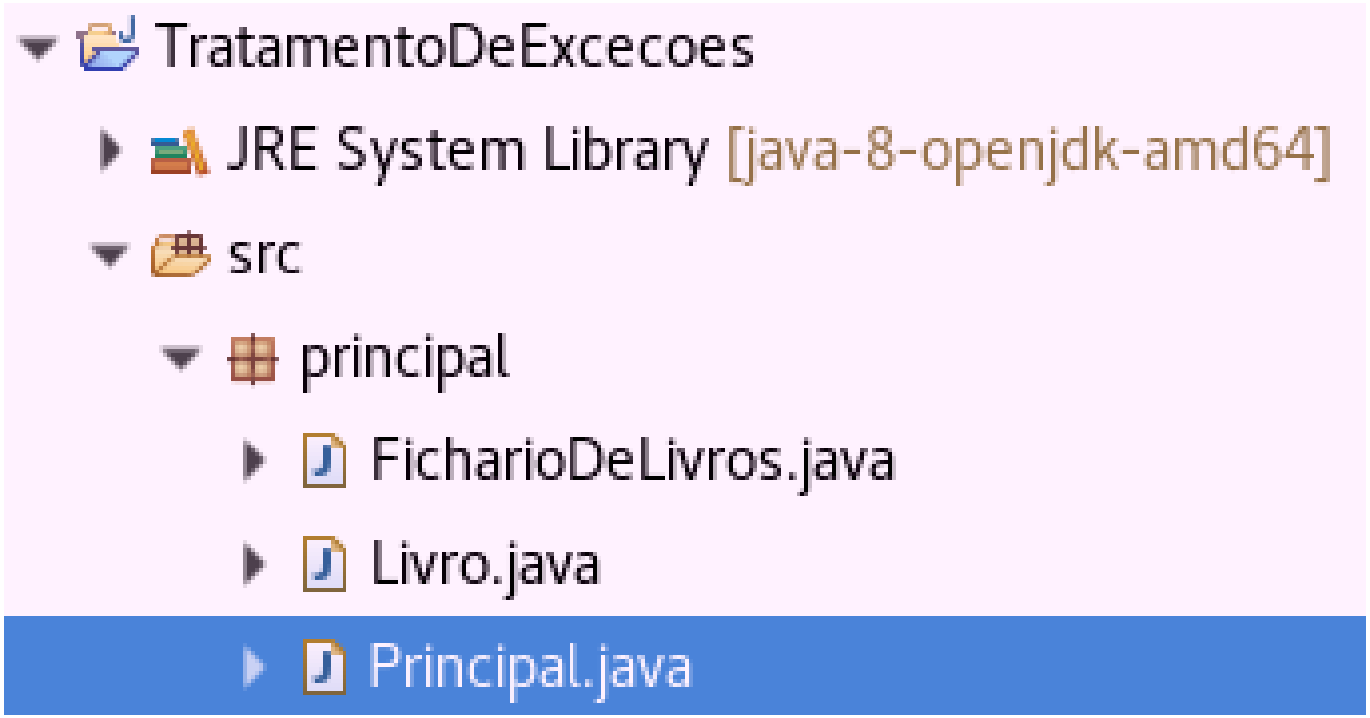
- Mais adiante veremos um exemplo prático.

Tratamento de Exceções

- Exceções podem ser tratadas a partir de trechos de códigos que tomam determinadas atitudes quando da ocorrência da anormalidade;
- Em Java para tratar uma exceção usa-se a instrução `try`:
  - A instrução `try` é composta por três blocos:
    - bloco **`try`**,
    - bloco **`catch`** e o
    - bloco **`finally`**. (Facultativo.)

Atenção ao Código!

- Projeto criado no Eclipse:



Classe Livro

Livro.java | FicharioDeLivros.java | Principal.java

```
1  package principal;
2
3  public class Livro {
4      private String titulo;
5      private String autor;
6
7      public Livro() {
8      }
9
10     public Livro(String titulo, String autor) {
11         this.titulo = titulo;
12         this.autor = autor;
13     }
14
15     public String getTitulo() {
16         return titulo;
17     }
18
19     public void setTitulo(String titulo) {
20         this.titulo = titulo;
21     }
22
23     public String getAutor() {
24         return autor;
25     }
26
27     public void setAutor(String autor) {
28         this.autor = autor;
29     }
30 }
```

Classe FicharioDeLivros

Livro.java | FicharioDeLivros.java | Principal.java

```
1  package principal;
2
3  import java.util.ArrayList;
4
5  public class FicharioDeLivros {
6      private ArrayList<Livro> livros;
7
8      public FicharioDeLivros(ArrayList<Livro> livros) {
9          this.livros = livros;
10     }
11
12     public void inserir() {
13         livros.add(new Livro("AAA","BBB"));
14         livros.add(new Livro("CCC","DDD"));
15         livros.add(new Livro("EEE","FFF"));
16         livros.add(new Livro("GGG","HHH"));
17         livros.add(new Livro("III","JJJ"));
18     }
19
20     public void imprimir() {
21         for(int i = 1; i <= 5; i++) {
22             mostraLivro(livros.get(i));
23         }
24         System.out.println("Todos os livros foram impressos!");
25     }
26
27     private void mostraLivro(Livro l) {
28         System.out.println("Título: " + l.getTitulo());
29         System.out.println("Autor : " + l.getAutor());
30     }
31 }
```

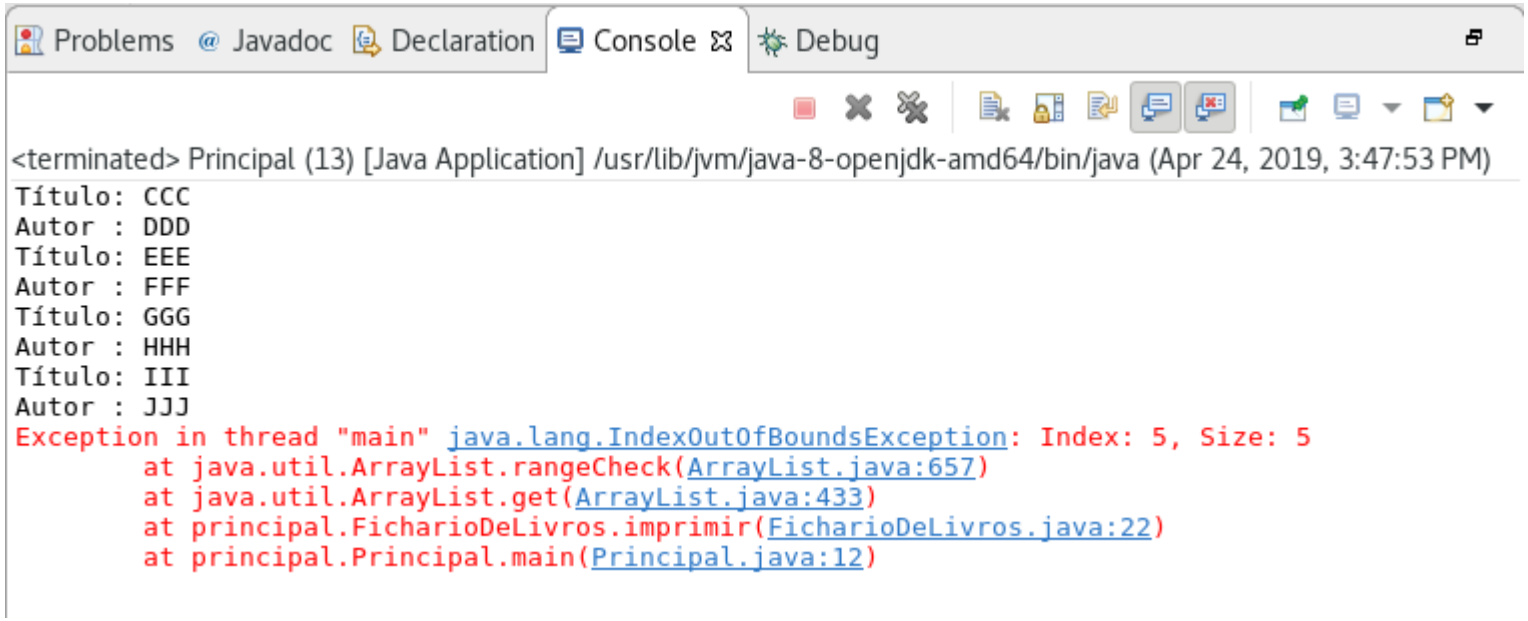
Classe Principal

Livro.java | FicharioDeLivros.java | Principal.java

```
1  package principal;
2
3  import java.util.ArrayList;
4
5  public class Principal {
6
7      public static void main(String[] args) {
8          ArrayList<Livro> aLivros = new ArrayList<>();
9          FicharioDeLivros fl = new FicharioDeLivros(aLivros);
10
11         fl.inserir();
12         fl.imprimir();
13     }
14 }
```

Erro!

- Existe um erro proposital no código que foi apresentado.
- E ao executar este programa o mesmo será finalizado com o seguinte “erro”:

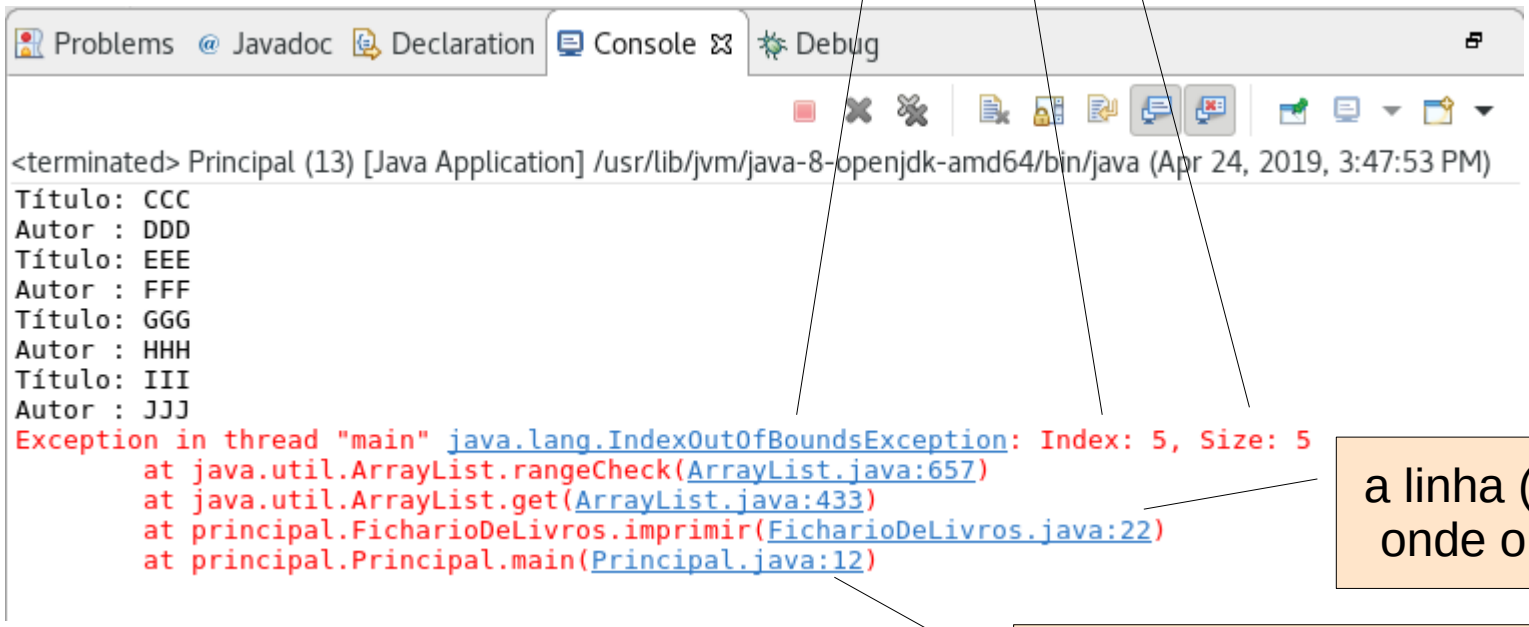


O Erro nos diz:

que o índice do vetor está fora do limite,

o índice que extrapolou o limite,

o tamanho do vetor,



a linha (22) do método onde o erro ocorreu e

a linha (12) que chamou o método onde o erro aconteceu.

Analisando o Código

```
12      public void inserir() {
13          livros.add(new Livro("AAA","BBB"));
14          livros.add(new Livro("CCC","DDD"));
15          livros.add(new Livro("EEE","FFF"));
16          livros.add(new Livro("GGG","HHH"));
17          livros.add(new Livro("III","JJJ"));
18      }
```

Foram inseridos 5 livros no ArrayList. Desta forma, o tamanho do ArrayList é 5 e seus índices vão de 0 a 4.

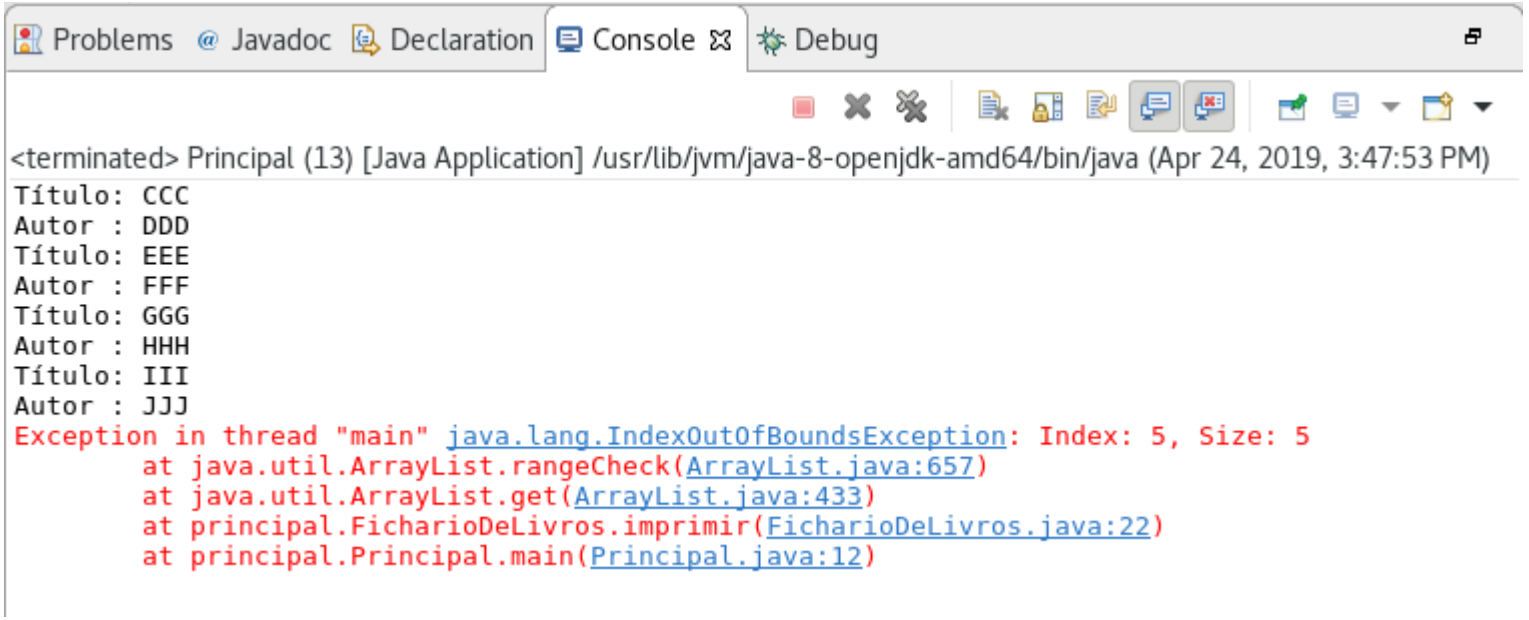
No “for” os índices percorridos vão de 1 a 5.

```
20      public void imprimir() {
21          for(int i = 1; i <= 5; i++) {
22              mostraLivro(livros.get(i));
23          }
24          System.out.println("Todos os livros foram impressos!");
25      }
```

Atenção!

```
20    public void imprimir() {
21        for(int i = 1; i <= 5; i++) {
22            mostraLivro(livros.get(i));
23        }
24        System.out.println("Todos os livros foram impressos!");
25    }
```

Após o “for” existe uma mensagem que não foi impressa.
Isto ocorreu porque o programa terminou antes de imprimi-la.

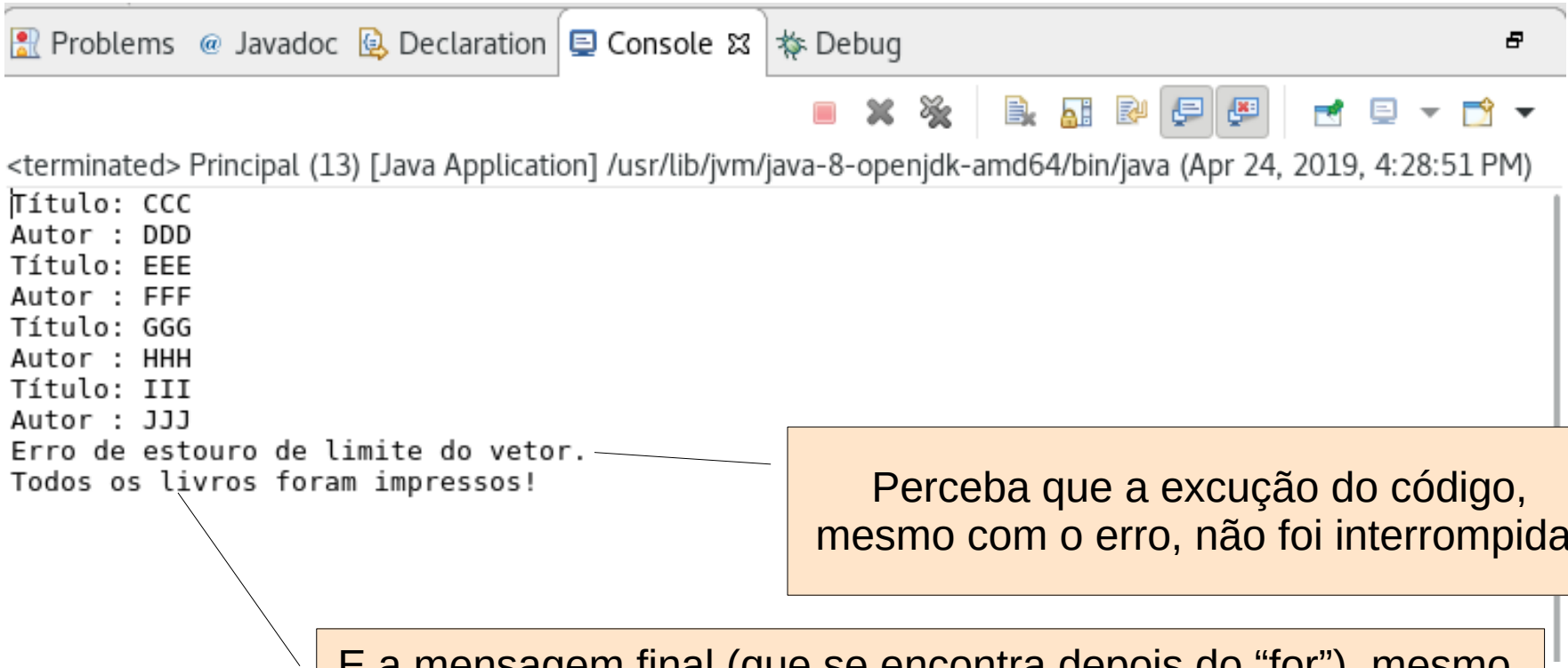


Blindando o Código

Livro.java | FicharioDeLivros.java | Principal.java

```
1  package principal;
2
3  import java.util.ArrayList;
4
5  public class FicharioDeLivros {
6      private ArrayList<Livro> livros;
7
8      public FicharioDeLivros(ArrayList<Livro> livros) {
9          this.livros = livros;
10     }
11
12     public void inserir() {
13         livros.add(new Livro("AAA","BBB"));
14         livros.add(new Livro("CCC","DDD"));
15         livros.add(new Livro("EEE","FFF"));
16         livros.add(new Livro("GGG","HHH"));
17         livros.add(new Livro("III","JJJ"));
18     }
19
20     public void imprimir() {
21         try {
22             for(int i = 1; i <= 5; i++) {
23                 mostraLivro(livros.get(i));
24             }
25         }
26         catch(IndexOutOfBoundsException e) {
27             System.out.println("Erro de estouro de limite do vetor.");
28         }
29         System.out.println("Todos os livros foram impressos!");
30     }
31
32     private void mostraLivro(Livro l) {
33         System.out.println("Título: " + l.getTitulo());
34         System.out.println("Autor : " + l.getAutor());
35     }
36 }
```

Execução Após Blindagem



Perceba que a excução do código, mesmo com o erro, não foi interrompida.

E a mensagem final (que se encontra depois do “for”), mesmo não condizente com a realidade, ainda assim, foi impressa. Ou seja, a execução do programa não foi interrompida.

Exemplo

Principal.java

```
1  import java.util.Scanner;
2
3  public class Principal {
4
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7
8          String n, d;
9          int num, den, div;
10
11         System.out.println("Digite o valor do numerador: ");
12         n = sc.nextLine();
13         System.out.println("Digite o valor do denominador: ");
14         d = sc.nextLine();
15
16         try {
17             num = Integer.valueOf(n);
18             den = Integer.valueOf(d);
19
20             div = num / den;
21
22             System.out.println("\nResultado: " + div);
23         }
24         catch(NumberFormatException x) {
25             System.out.println("Erro de formatação. Error: " + x.getMessage());
26         }
27         catch(ArithmeticException j) {
28             System.out.println("Erro de divisão por zero. Error: " + j.getMessage());
29         }
30         catch(Exception f) {
31             System.out.println("Erro no cálculo. Error: " + f.getMessage());
32         }
33         sc.close();
34     }
35 }
```

Explicando:

- No exemplo do *slide* anterior a cláusula `catch(NumberFormatException x)` captura exceções de formatação numérica;
- A cláusula `catch(ArithmeticException j)` captura exceções ocorridas em operações aritméticas, por exemplo o erro de divisão por zero e a
- Cláusula `catch(Exception f)` captura qualquer tipo de exceção ocorrida dentro da instrução `try`;
- Os comandos dentro dos blocos `catch` são o tratamento das possíveis exceções;
- Já as variáveis `x`, `j` e `f`, cada qual, contém a sua respectiva exceção no idioma da linguagem de programação, ou seja, o inglês, mais os detalhes técnicos desta.

Propagação de Exceções

- Quando uma exceção ocorre a JVM busca pela cláusula `catch` associada ao seu tipo. A sequência para realização da busca é a que aparece no código (aninhamento de `catch`'s);
- **Quando uma exceção não é tratada no bloco em que ocorreu, ela é propagada para o bloco mais externo;**
- Se for uma chamada de método, então é propagada para o método chamador.

Exemplo Hipotético

```
...
try{
    try{
        // Aqui pode ocorrer uma exceção do tipo IOException.
    }
    catch(IOException e){
        ...         // Tratamento parcial.
    }
}
catch(IOException e){
    ...   // Restante do tratamento.
}
```

Exemplo em Código

Principal.java

```
4  public class Principal {
5
6      public static int quociente(int num, int den){
7          return num / den;
8      }
9
10     public static void main(String[] args) {
11         Scanner sc = new Scanner(System.in);
12         int num, den, div;
13         boolean resp;
14
15         do {
16             System.out.println("Digite o valor do numerador: ");
17             num = sc.nextInt();
18             System.out.println("Digite o valor do denominador: ");
19             den = sc.nextInt();
20
21             try {
22                 div = quociente(num, den);
23
24                 System.out.printf("\nResultado da divisão de %d / %d = %d", num, den, div);
25             }
26             catch(ArithmeticException j) {
27                 System.out.println("\nErro de divisão por zero. Error: " + j.getMessage());
28             }
29             catch(InputMismatchException f) {
30                 System.out.println("\nDigite um número inteiro. Error: " + f.getMessage());
31             }
32
33             System.out.println("\n\nDeseja fazer outro cálculo? (true/false): ");
34             resp = sc.nextBoolean();
35         }while(resp);
36         sc.close();
37     }
38 }
```

Problems | Javadoc | Declaration | Console | Debug

```
Principal (15) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/j
Digite o valor do numerador:
5
Digite o valor do denominador:
0
Erro de divisão por zero. Error: / by zero


Deseja fazer outro cálculo? (true/false):
```

Propagação de Exceções

- Se o bloco ou método para o qual a exceção foi propagada não fizer seu tratamento, então a exceção continua sendo propagada;
- Se a exceção chegar ao método principal (`main`) e também não for tratada, então o programa é abortado.

Exemplo Sem Tramamento 1

Principal.java

```
1  import java.util.Scanner;
2
3  public class Principal {
4
5      public static int quociente(int num, int den){
6          return num / den;
7      }
8
9      public static void main(String[] args) {
10         Scanner sc = new Scanner(System.in);
11         int num, den, div;
12         boolean resp;
13
14         do {
15             System.out.println("Digite o valor do numerador: ");
16             num = sc.nextInt();
17             System.out.println("Digite o valor do denominador: ");
18             den = sc.nextInt();
19
20             div = quociente(num, den);
21
22             System.out.printf("\nResultado da divisão de %d / %d = %d", num, den, div);
23
24             System.out.println("\n\nDeseja fazer outro cálculo? (true/false): ");
25             resp = sc.nextBoolean();
26         }while(resp);
27         sc.close();
28     }
29 }
```

Problems | Javadoc | Declaration | Console | Debug

```
<terminated> Principal (16) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/ja
Digite o valor do numerador:
5
Digite o valor do denominador:
0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Principal.quociente(Principal.java:6)
        at Principal.main(Principal.java:20)
```

Exemplo Sem Tratamento 2

Principal.java

```
1  import java.util.Scanner;
2
3  public class Principal {
4
5      public static int quociente(int num, int den){
6          return num / den;
7      }
8
9      public static int metodoIntermediario(int v1, int v2) {
10         return quociente(v1,v2);
11     }
12
13     public static void main(String[] args) {
14         Scanner sc = new Scanner(System.in);
15         int num, den, div;
16         boolean resp;
17
18         do {
19             System.out.println("Digite o valor do numerador: ");
20             num = sc.nextInt();
21             System.out.println("Digite o valor do denominador: ");
22             den = sc.nextInt();
23
24             div = metodoIntermediario(num, den);
25
26             System.out.printf("\nResultado da divisão de %d / %d = %d", num, den, div);
27
28             System.out.println("\n\nDeseja fazer outro cálculo? (true/false): ");
29             resp = sc.nextBoolean();
30         }while(resp);
31         sc.close();
32     }
33 }
```

Problems | Javadoc | Declaration | Console | Debug

```
<terminated> Principal (17) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/jav
Digite o valor do numerador:
5
Digite o valor do denominador:
0
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Principal.quociente(Principal.java:6)
        at Principal.metodoIntermediario(Principal.java:10)
        at Principal.main(Principal.java:24)
```

A Cláusula `throws`

- A cláusula **`throws`** faz parte da assinatura do método;
- Ela foi criada para ser um indicativo de que o método pode gerar uma exceção do tipo que foi declarado na cláusula **`throws`**;
- Desta forma, um trecho de código que venha a chamar este método deve obrigatoriamente capturar uma possível exceção que o método poderá lançar.

Exemplo

Principal.java

```
4  public class Principal {
5
6      public static int quociente(int num, int den) throws ArithmeticException{
7          return num / den;
8      }
9
10     public static void main(String[] args) {
11         Scanner sc = new Scanner(System.in);
12         int num, den, div;
13         boolean resp;
14
15         do {
16             System.out.println("Digite o valor do numerador: ");
17             num = sc.nextInt();
18             System.out.println("Digite o valor do denominador: ");
19             den = sc.nextInt();
20
21             try {
22                 div = quociente(num, den);
23
24                 System.out.printf("\nResultado da divisão de %d / %d = %d", num, den, div);
25             }
26             catch(ArithmeticException j) {
27                 System.out.println("\nErro de divisão por zero. Error: " + j.getMessage());
28             }
29             catch(InputMismatchException f) {
30                 System.out.println("\nDigite um número inteiro. Error: " + f.getMessage());
31             }
32
33             System.out.println("\n\nDeseja fazer outro cálculo? (true/false): ");
34             resp = sc.nextBoolean();
35         }while(resp);
36         sc.close();
37     }
38 }
```

Problems | Javadoc | Declaration | Console

```
Principal (15) [Java Application] /usr/lib/jvm/java-8-openj
Digite o valor do numerador:
5
Digite o valor do denominador:
0

Erro de divisão por zero. Error: / by zero


Deseja fazer outro cálculo? (true/false):
```

Atenção!

- O fato de existir a cláusula **`throws`** na assinatura do método não eximi o programador de ter que colocar a chamada do método dentro de uma instrução **`try … catch`**.

Relançamento de Exceções

- Em algumas situações, pode ser necessário que o local onde ocorreu a exceção a trate de maneira parcial, deixando o restante para blocos mais externos;
- Neste caso, utiliza-se o recurso de relançamento de exceções.

Exemplo Hipotético

```
...
try{
    try{
        // Aqui pode ocorrer uma exceção do tipo IOException.
    }
    catch(IOException e){
        ...          // Tratamento parcial.
        throw e;     // Relançamento da exceção.
    }
}
catch(IOException e){
    ...   // Restante do tratamento.
}
```

Exemplo em Código

```java
import java.util.InputMismatchException;
import java.util.Scanner;

public class Principal {

    public static int quociente(int num, int den) throws ArithmeticException{
        return num / den;
    }

    public static int metodoIntermediario(int v1, int v2) throws Exception {
        try {
            return quociente(v1,v2);
        }
        catch(Exception e) {
            throw e;  // Relançando a exceção.
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int num, den, div;
        boolean resp;

        do {
            System.out.println("Digite o valor do numerador: ");
            num = sc.nextInt();
            System.out.println("Digite o valor do denominador: ");
            den = sc.nextInt();

            try {
                div = quociente(num, den);

                System.out.printf("\nResultado da divisão de %d / %d = %d", num, den, div);
            }
            catch(ArithmeticException | InputMismatchException j) {
                System.err.println("\n\nErro de divisão por zero. Error: " + j.getMessage());
            }

            System.out.println("\n\nDeseja fazer outro cálculo? (true/false): ");
            resp = sc.nextBoolean();
        }while(resp);
        sc.close();
    }
}
```

O método não trata, mas relança a exceção

```
adoc  Declaration  Console
Principal (18) [Java Application] /usr/lib/jvm/java-8-openjdk-am
Digite o valor do numerador:
5
Digite o valor do denominador:
0

Deseja fazer outro cálculo? (true/false):

Erro de divisão por zero. Error: / by zero
```

Continuação após o Tratamento de Exceções

- Em algumas situações pode ser necessária a execução de um conjunto de comandos, independentemente de ter ocorrido uma exceção na instrução **`try`**;
- A cláusula **`finally`** de Java provê este recurso;
- Em geral, este recurso é utilizado quando deseja-se restabelecer o estado de algum objeto de forma independente da ocorrência e da propagação de exceções;
- Exemplo:
  - encerramento de conexões com banco de dados ou fechamento de arquivos, quando houver a ocorrência de determinadas exceções.

Continuação após o Tratamento de Exceções

- É possível a existência de um bloco **`try`** sem um bloco **`catch`**. Mas não é possível existir uma instrução **`try`** sem, pelo menos, um bloco **`catch`** ou um bloco **`finally`**;
- Havendo um bloco **`finally`** na instrução **`try`** e independente do que aconteça, o código dentro do bloco **`finally`** sempre será executado.

Exemplos Hipotético

```
try{
   ...
   // Código qualquer.
   ...
}
finally{
   ...
   // Código qualquer.
   ...
}
```

```
try{
   // Código qualquer.
}
catch(Exception e){
   // Código qualquer.
}
finally{
   // Código qualquer.
}
```

Exemplo Try ... Finally

Principal.java

```
1  import java.util.Scanner;
2
3  public class Principal {
4
5      public static int quociente(int num, int den){
6          return num / den;
7      }
8
9      public static void main(String[] args) {
10         Scanner sc = new Scanner(System.in);
11         int num, den, div;
12         boolean resp;
13
14         do {
15             System.out.println("Digite o valor do numerador: ");
16             num = sc.nextInt();
17             System.out.println("Digite o valor do denominador: ");
18             den = sc.nextInt();
19
20             try {
21                 div = quociente(num, den);
22
23                 System.out.printf("\nResultado da divisão de %d / %d = %d", num, den, div);
24                 System.out.println("\n\n-----------------");
25             }
26             finally {
27                 System.err.println("Com ou sem exceção eu passo por aqui.");
28             }
29
30             System.out.println("Deseja fazer outro cálculo? (true/false): ");
31             resp = sc.nextBoolean();
32         }while(resp);
33         sc.close();
34     }
35 }
```

Problems @ Javadoc Declaration Console Debug

```
<terminated> Principal (19) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/ja
Digite o valor do numerador:
5
Digite o valor do denominador:
0
Com ou sem exceção eu passo por aqui.
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Principal.quociente(Principal.java:6)
        at Principal.main(Principal.java:21)
```

Problems @ Javadoc Declaration Console

```
Principal (19) [Java Application] /usr/lib/jvm/java-8-openjdk-a
Digite o valor do numerador:
5
Digite o valor do denominador:
2

Resultado da divisão de 5 / 2 = 2

-----------------
Deseja fazer outro cálculo? (true/false):
Com ou sem exceção eu passo por aqui.
```

Exempo Try, Catch, Finally

Principal.java

```
4  public class Principal {
5
6      public static int quociente(int num, int den){
7          return num / den;
8      }
9
10     public static void main(String[] args) {
11         Scanner sc = new Scanner(System.in);
12         int num, den, div;
13         boolean resp;
14
15         do {
16             System.out.println("Digite o valor do numerador: ");
17             num = sc.nextInt();
18             System.out.println("Digite o valor do denominador: ");
19             den = sc.nextInt();
20
21             try {
22                 div = quociente(num, den);
23
24                 System.out.printf("\nResultado da divisão de %d / %d = %d", num, den, div);
25                 System.out.println("\n\n-----------------");
26             }
27             catch(ArithmeticException | InputMismatchException x) {
28                 System.out.println("\n\nExceção. Error: " + x.getMessage());
29             }
30             finally {
31                 System.err.println("\n\nCom ou sem exceção eu passo por aqui.");
32             }
33
34             System.out.println("Deseja fazer outro cálculo? (true/false): ");
35             resp = sc.nextBoolean();
36         }while(resp);
37         sc.close();
38     }
39 }
```

Problems @ Javadoc Declaration Console

```
Principal (20) [Java Application] /usr/lib/jvm/java-8-openjd
Digite o valor do numerador:
5
Digite o valor do denominador:
2

Resultado da divisão de 5 / 2 = 2

-----------------
Deseja fazer outro cálculo? (true/false):


Com ou sem exceção eu passo por aqui.
```

Problems @ Javadoc Declaration Console

```
Principal (20) [Java Application] /usr/lib/jvm/java-8-openjd
Digite o valor do numerador:
5
Digite o valor do denominador:
0

Exceção. Error: / by zero

Com ou sem exceção eu passo por aqui.

Deseja fazer outro cálculo? (true/false):
```

Dúvidas?

Bibliografia

- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; tradução Edson Furmankiewicz; revisão técnica Fábio Lucchini. 6a. ed., São Paulo: Pearson, 2005.
- DESENVOLVE WEB. IconesBr. Disponível em: <http://www.iconesbr.net/index.php?acao=search&q=aten%E7%E3o&s=>. Acesso em: 01 maio 2019.
- FERREIRA, Kecia Aline Marques. *Slides* da disciplina de Programação de Computadores II. CEFET-MG, 2009.
- STACKOVERFLOW. Usando as palavras-chave Throws e Throw. Disponível em: <https://pt.stackoverflow.com/questions/17025/usando-as-palavras-chave-throws-e-throw>. Acesso em: 01 maio 2019.