

Tratamento de Exceções

Prof. Ricardo P. Mesquita

Exceções

- Aplicações, durante a execução, podem incorrer em muitas espécies de erros de vários graus de severidade
- Quando métodos são invocados sobre um objeto:
 - Problemas de estado interno
 - Erros com objetos ou dados que eles manipulam
 - Ele pode estar violando seu contrato básico

Exceções

- Acontece quando encontra algo inesperado:
 - Problemas no hardware
 - Arrays fora de faixa
 - Valores de variáveis
 - Divisão por zero
 - Erro de entrada e saída (IO)
 - Erros da aplicação
 - Saldo insuficiente
 - Parâmetros de métodos
 - Falha de Memória
 - Usuário não existe
 - Nota inválida

Exceções

- É desagradável encontrar erros...
- O que deve ser feito:
 - Notificar o usuário de um erro;
 - Conseguir salvar todo o trabalho
 - Permitir que usuários saiam elegantemente do programa

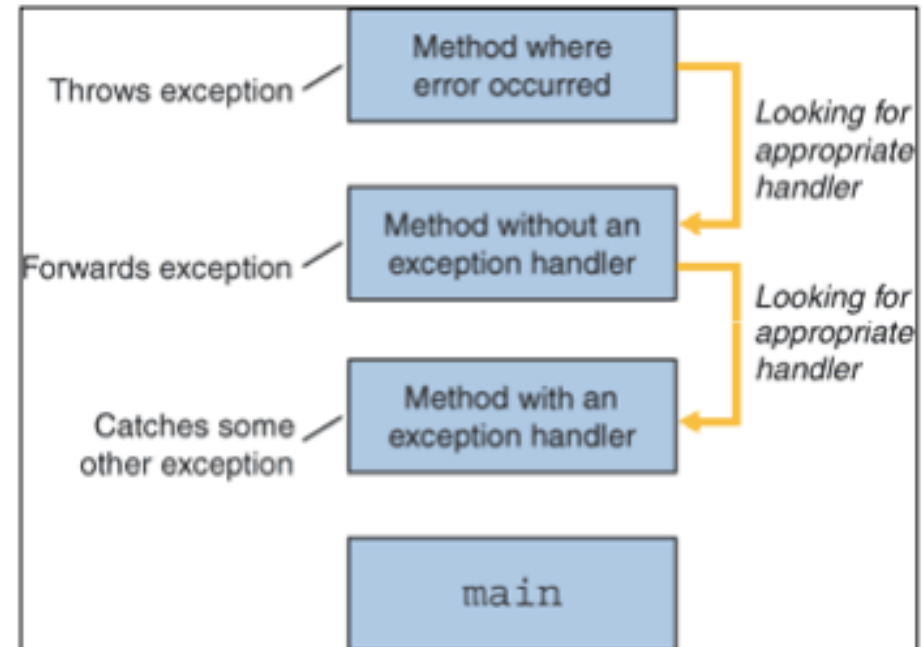
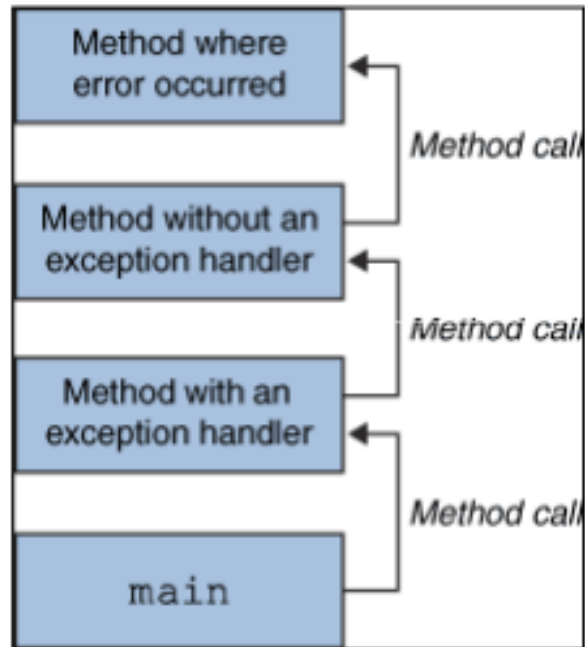
Exceções

- Exceção é um desvio no fluxo de execução normal do programa
- Indica que houve problema na execução de um bloco do programa
- Se não for tratada, programa pode parar
- O uso correto de exceções torna o programa mais robusto e confiável

Exceções

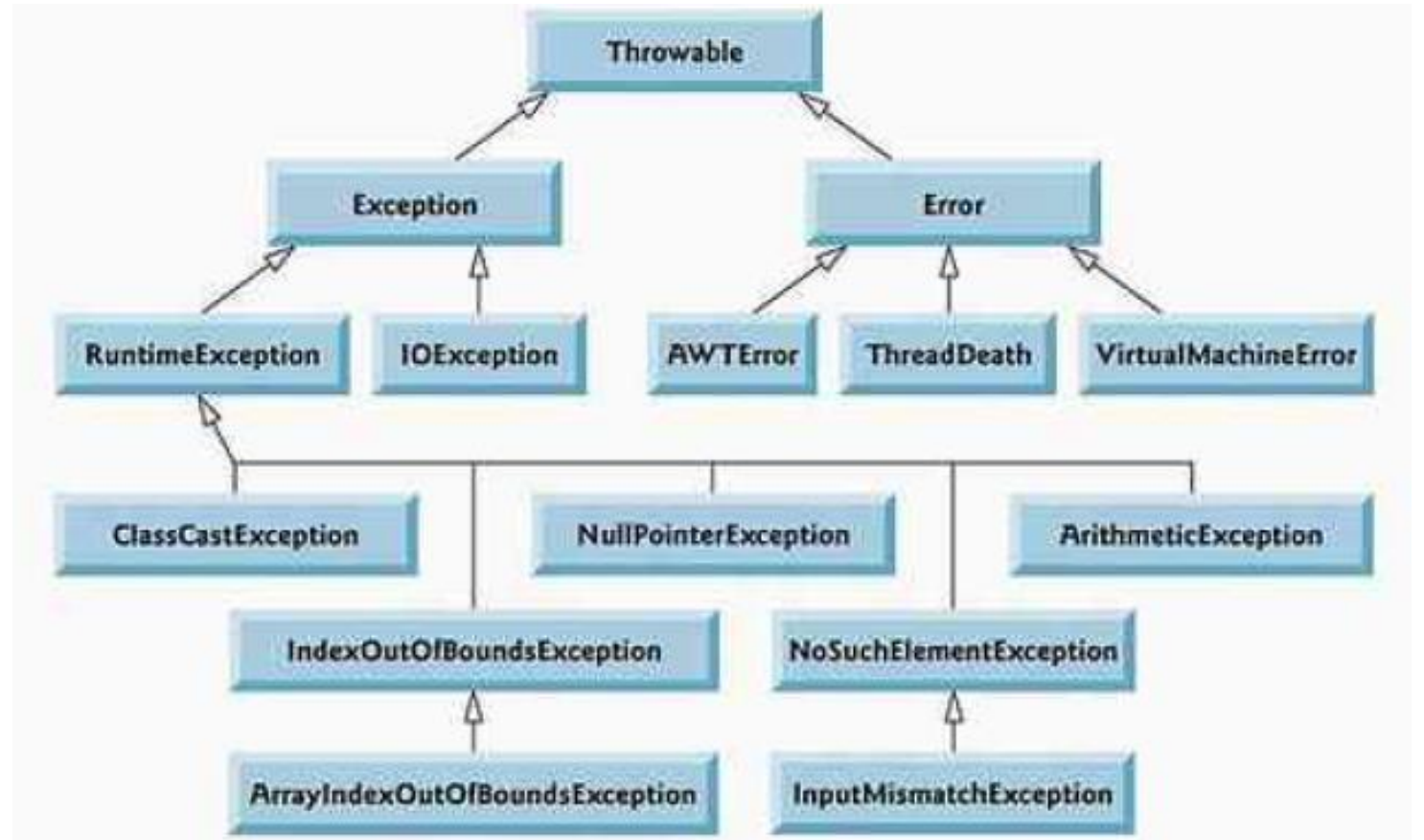
- Quando um erro ocorre dentro de um método:
 - Um objeto é criado
 - O objeto contém informações sobre o erro, assim como o tipo e o estado do programa
 - A ação de capturar esse objeto chamamos de ***throwing an exception***
 - O ambiente de execução (***runtime system***) tenta encontrar algum tratamento a exceção
 - A busca segue a ***call stack*** – lista de chamadas de métodos

Exceções



Hierarquia de Exceções

- Em Java, um objeto de exceção é sempre uma instância de uma classe derivada ***Throwable***



Hierarquia de Exceções

- **Error:**
 - Ocorrem devido a problemas de SO ou hardware
 - Não deve lançar um objeto desse tipo
- Ao se fazer um programa Java, deve-se focar na hierarquia **Exception**
 - Divide-se em dois ramos: as que derivam de **RuntimeException** e as que não derivam

Hierarquia de Exceções

- ***RuntimeException:***

- Acontece porque se fez um erro de programação

- Exemplos:

- Acesso de array proibido;
 - acesso de ponteiro nulo.

- **Qualquer outra Exceção:**

- Ocorre porque algo ruim, como um erro de E/S, aconteceu com o programa bom em outros aspectos

- Exemplos:

- Tentar ler além do fim de um arquivo;
 - tentar abrir um URL malformatado.

Tipos de Exceções

- A *Java Language Specification* define dois tipos de exceções:
 - **Exceção Não Verificada (*unchecked*):**
 - Deriva da classe `Error` ou da `RuntimeException`
 - Não precisam ser tratadas, mas podem ser
 - `NullPointerException`, `NumberFormatException`, `ArrayIndexOutOfBoundsException`
 - Se uma exceção não verificada ocorrer e não for tratada, o programa pode parar (console)
 - **Exceção Verificada (*checked*):**
 - Exceções previsíveis
 - Devem ser tratadas pelo programa

Formas de Captura

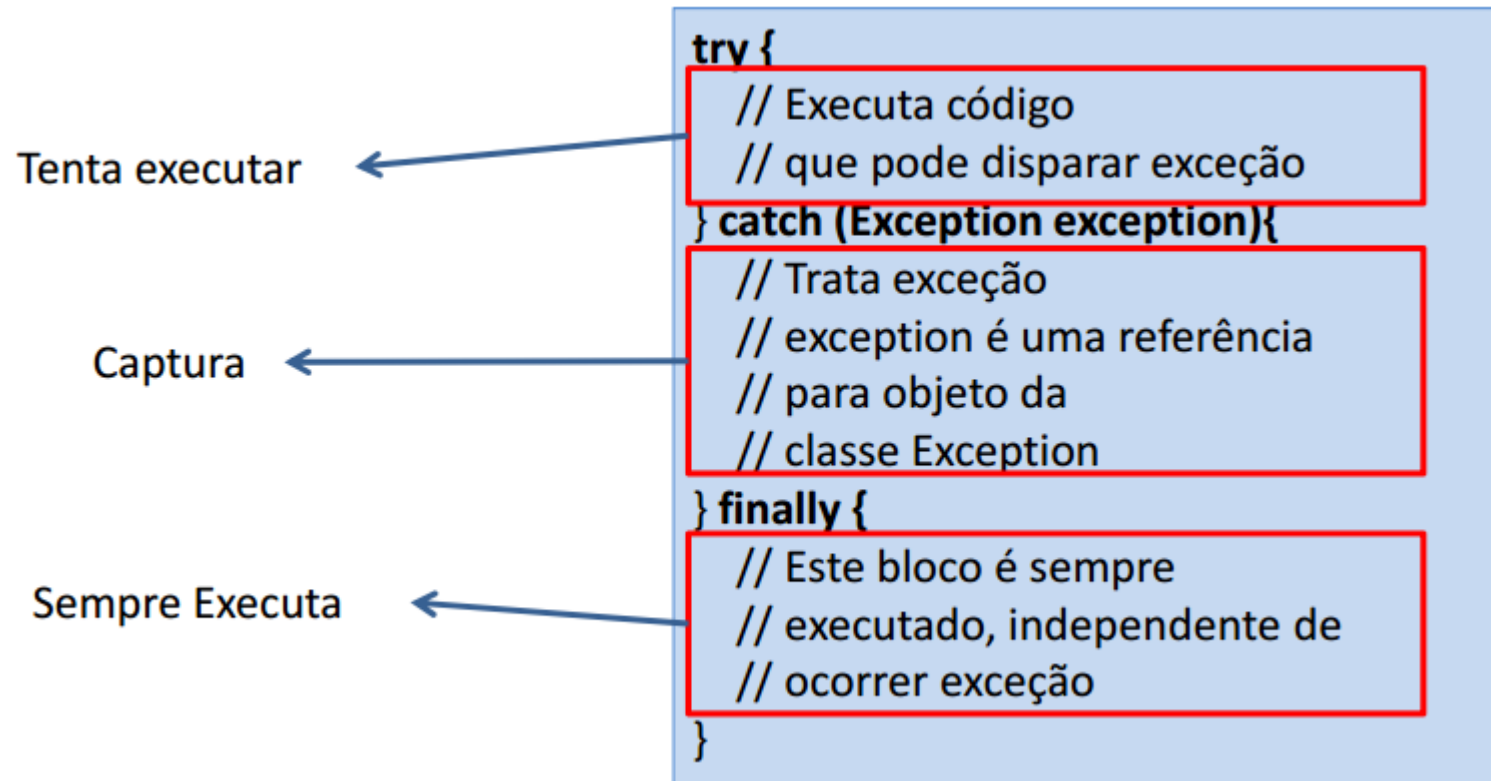
- Através de uma ***cláusula throws*** na assinatura do método
- Ou através de uma ***bloco try***

Formas de Captura

- Um método deve informar que exceções ele pode disparar (**throw**)
 - cláusula *throws* na definição do método
- Um bloco que tenta (**try**) chamar um método que pode disparar uma exceção deve tratá-la
 - Chamada normal de um método, mas que deve estar em um *bloco try* {...} **catch** {...}
- Uma exceção é um objeto que deve ser capturado (**catch**)
 - É nesse bloco que a exceção deve ser tratada
- Um trecho de código pode ser executado sempre
 - *bloco finally*

Tratando Exceções

- 1ª Forma: bloco **try-catch-finally**
 - Um bloco deve capturar uma exceção para tratá-la



Exemplo

- O método **parseInt** pode disparar exceção **NumberFormatException** (não verificada)
 - Se a exceção for disparada os comandos do bloco try não serão mais executados.
 - O fluxo de execução muda para a captura (bloco catch)
 - Ela pode não ser tratada.
 - Programa encerra se for disparada

```
System.out.print("Digite um número");
String numero = sc.nextLine();
int x;
try{
    x = Integer.parseInt(numero);
    System.out.println("Numero digitado é válido");
} catch (NumberFormatException exception){
    System.out.println("Digite um número válido");
}
```

Catch

- Pode haver *mais de um* bloco catch
 - Cada bloco trata um tipo específico de exceção
 - O bloco try contém métodos que podem disparar todas as exceções
 - Um método pode disparar mais de uma exceção

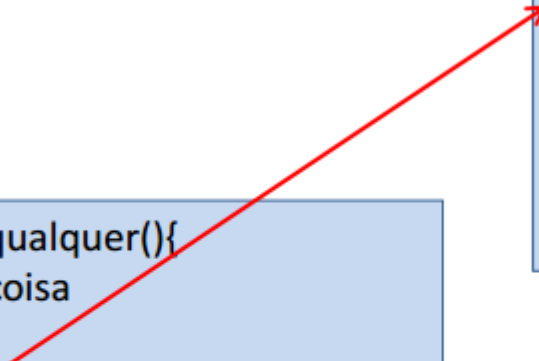
```
try {  
    // bloco de comandos  
} catch (Excecao1 ex1) {  
    // Trata exceção1  
} catch (Excecao2 ex2) {  
    // Trata exceção2  
} catch (Excecao3 ex3){  
    // Trata exceção3  
}
```


Repassando a Responsabilidade

- Um método pode repassar uma exceção
 - Ele chama um método que dispara uma exceção, mas não quer tratar
 - Ele pode repassar a exceção
- Basta colocar a cláusula **throws** na assinatura do método
- Para quem chama, é o método que dispara a exceção

Repassando a Responsabilidade

```
public void qualquer(){  
    // alguma coisa  
    try {  
        metodo();  
    } catch (AlgumaException e) {  
        e.printStackTrace();  
    }  
    // mais coisa  
}
```



```
public void metodo() throws AlgumaException{  
    // Corpo do método chama  
    // método que dispara AlgumaException  
    obj.metodoQueDisparaExcecao();  
}
```

Definindo Exceções

- **1º Passo:** Defina uma classe que herde de Exception
 - Ou RuntimeException se desejar fazer exceção não verificada

```
public class ContatoNaoEncontradoException extends Exception {  
    public ContatoNaoEncontradoException()  
    {  
        super("Contato não encontrado");  
    }  
}
```

Definindo Exceções

- **2º Passo:** No método que dispara a exceção:
 - Coloque a cláusula throws
 - Crie o objeto da classe de exceção
 - Dispare (*throw*) a exceção
 - Onde a exceção será disparada depende de cada método

```
public Contato buscar(String nome)
    throws ContatoNaoEncontradoException{
    // Laco para procurar contato pelo nome
    for (int i = 0 ; i < quantidade ; i++)
        if (contatos[i].nome().equals(nome))
            return contatos[i];
    // Se sair do laço e não tiver retornado
    // o contato não esta cadastrado
    // deve disparar excecao
    throw new ContatoNaoEncontradoException();
}
```

Definindo Exceções

- **3º Passo:** A exceção é tratada na interface com o usuário

```
private void buscarContato() {  
    System.out.print("Digite o nome: ");  
    String nome = sc.nextLine();  
    try {  
        Contato contato = agenda.buscar(nome);  
        System.out.println(contato);  
    } catch (ContatoNaoEncontradoException e) {  
        System.out.println("ERRO!!!!");  
        System.out.println(e.getMessage());  
        System.out.println("\nDigite [ENTER] para continuar....");  
        sc.nextLine();  
    }  
}
```

Exemplo 1

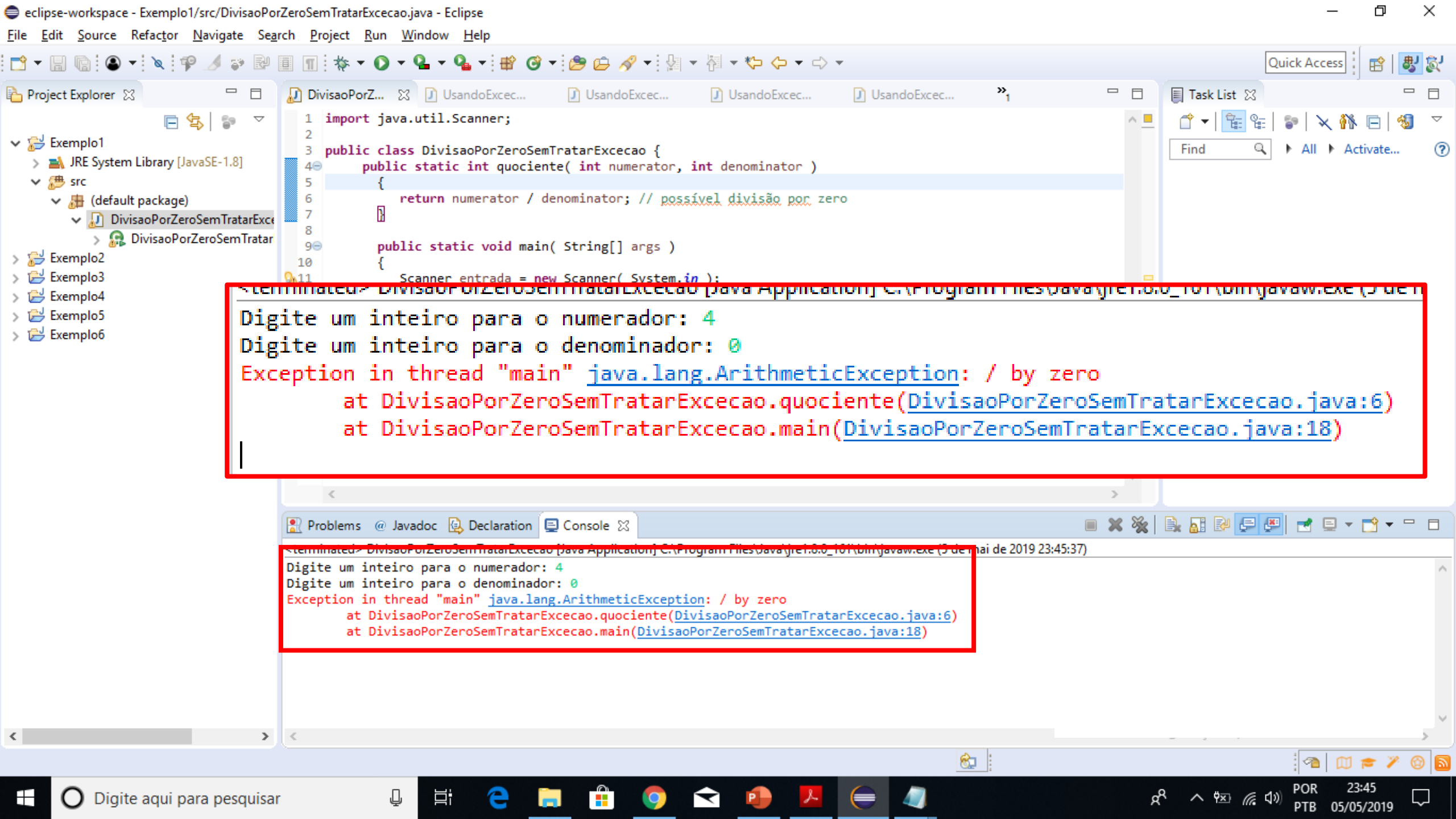
```
import java.util.Scanner;

public class DivisaoPorZeroSemTratarExcecao {
    public static int quociente( int numerator, int denominator )
    {
        return numerator / denominator; // possível divisão por zero
    }

    public static void main( String[] args )
    {
        Scanner entrada = new Scanner( System.in );

        System.out.print( "Digite um inteiro para o numerador: " );
        int numerador = entrada.nextInt();
        System.out.print( "Digite um inteiro para o denominador: " );
        int denominador = entrada.nextInt();

        int resultado = quociente( numerador, denominador );
        System.out.printf(
            "\nResultado: %d / %d = %d\n", numerador, denominador,
            resultado );
    }
}
```



Exemplo 2

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

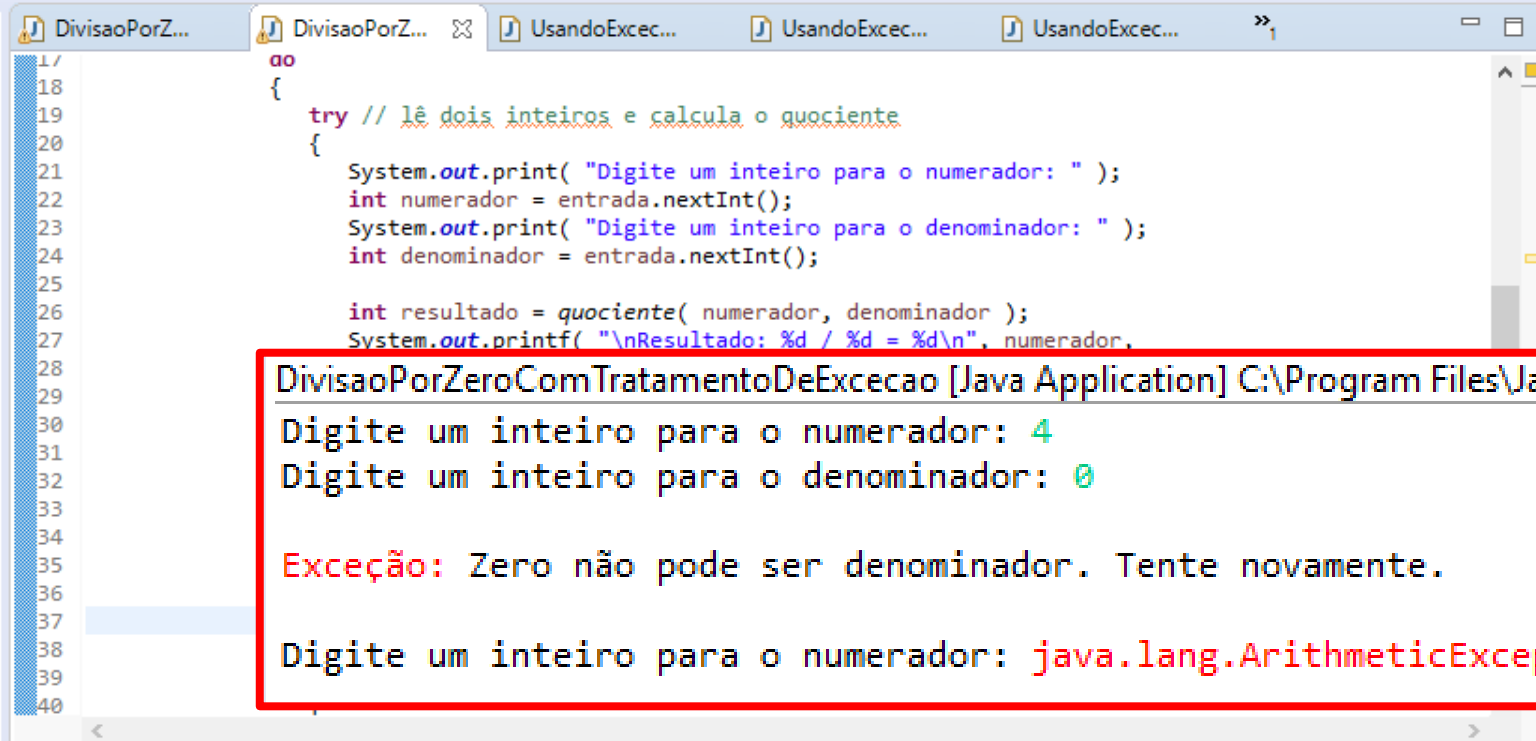
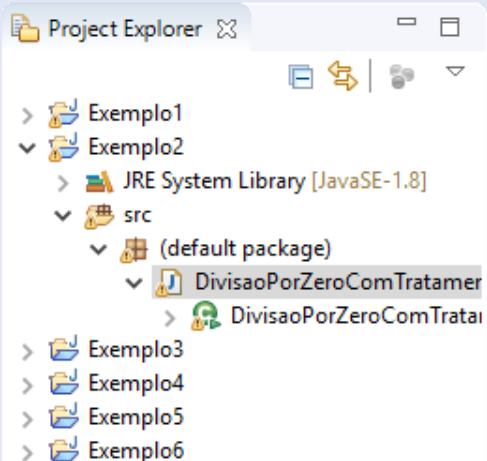
public class DivisaoPorZeroComTratamentoDeExcecao {

    public static int quociente( int numerador, int denominador )
        throws ArithmeticException
    {
        return numerador / denominador; // possível divisão por zero
    }

    public static void main( String[] args )
    {
        Scanner entrada = new Scanner( System.in );
        boolean continueLoop = true; // determina se mais entradas são necessárias

        do
        {
            try // lê dois inteiros e calcula o quociente
            {
                System.out.print( "Digite um inteiro para o numerador: " );
                int numerador = entrada.nextInt();
                System.out.print( "Digite um inteiro para o denominador: " );
                int denominador = entrada.nextInt();

                int resultado = quociente( numerador, denominador );
                System.out.printf( "\nResultado: %d / %d = %d\n", numerador,
                    denominador, resultado );
                continueLoop = false; // encerra o loop
            }
            catch ( InputMismatchException inputMismatchException )
            {
                System.out.print( "Erro: entrada inválida. Digite um inteiro. " );
                entrada.next();
            }
        } while ( continueLoop );
    }
}
```

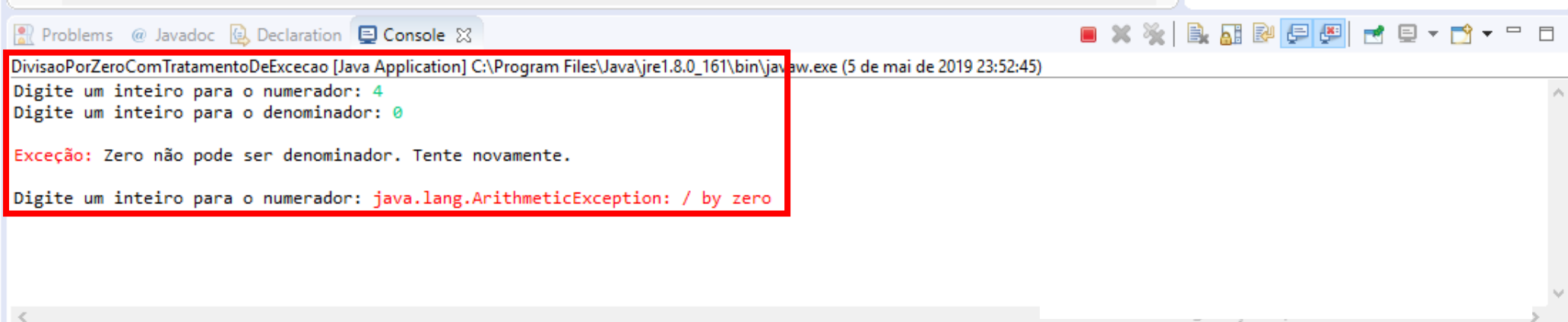



DivisaoPorZeroComTratamentoDeExcecao [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (5 de mai de 2019 23:52:45)

Digite um inteiro para o numerador: 4
Digite um inteiro para o denominador: 0

Exceção: Zero não pode ser denominador. Tente novamente.

Digite um inteiro para o numerador: java.lang.ArithmeticException: / by zero

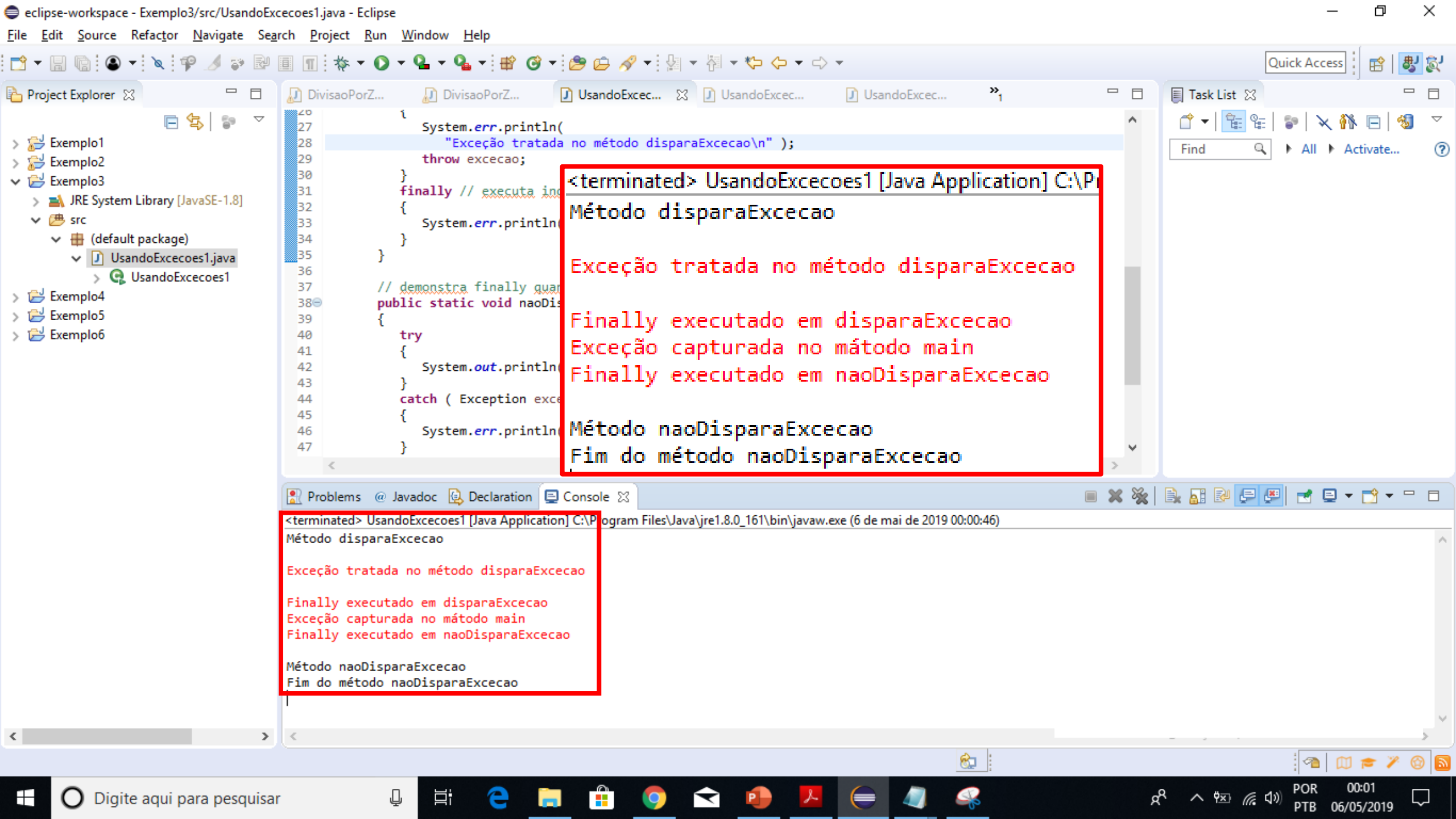


Exemplo 3

```
public class UsandoExcecoes1 {
    public static void main( String[] args )
    {
        try
        {
            disparaExcecao();
        }
        catch ( Exception exception )
        {
            System.err.println( "Exceção capturada no método main" );
        }

        naoDisparaExcecao();
    }
}
```

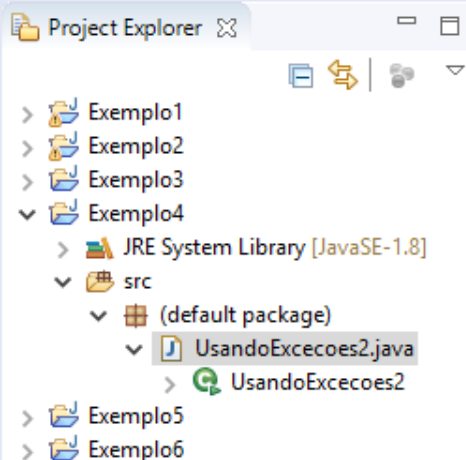
```
public static void disparaExcecao() throws Exception
{
    try
    {
        System.out.println( "Método disparaExcecao" );
        throw new Exception(); //dispara uma exceção
    }
    catch ( Exception excecao )
    {
        System.err.println(
            "Exceção tratada no método disparaExcecao" );
        throw excecao;
    }
    finally // executa independentemente do que ocorre no bloco try...catch
    {
        System.out.println( "Finally executado em disparaExcecao" );
    }
}
```



Exemplo 4

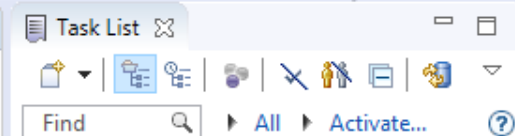
```
public class UsandoExcecoes2 {
    public static void main( String[] args )
    {
        try
        {
            disparaExcecao();
        }
        catch ( Exception e )
        {
            System.err.println( "Exceção tratada no método main" );
        }
    }

    public static void disparaExcecao() throws Exception
    {
        try
        {
            System.out.println( "Método disparaExcecao" );
            throw new Exception();
        }
        catch ( RuntimeException runtimeException ) // catch em tipo incorreto
        {
            System.err.println(
                "Exceção tratada no método disparaExcecao" );
        }
        finally
        {
            System.err.println( "\nFinally é sempre executado" );
        }
    }
}
```



```
9      catch ( Exception e )
10     {
11         System.err.println( "Exceção tratada no método main" );
12     }
13 }
14
15 public static void disparaExcecao() throws Exception
16 {
17     try
18     {
19         System.out.println( "Método disparaExcecao" );
20         throw new Exception();
21     }
22     catch ( RuntimeException runtimeException ) // catch em tipo incorreto
23     {
24         System.err.println(
25             "Exceção t
26         );
27     }
28     finally
29     {
30         System.err.pr
31     }
32 }
33 }
```

<terminated> UsandoExcecoes2 [Java App
Método disparaExcecao
Finally é sempre executado
Exceção tratada no método main



Problems @ Javadoc Declaration Console

<terminated> UsandoExcecoes2 [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (6 de mai de 2019 00:04:20)
Método disparaExcecao
Finally é sempre executado
Exceção tratada no método main

Exemplo 5

```
public class UsandoExcecoes3 {
public static void main( String[] args )
{
    try
    {
        metodo1();
    }
    catch ( Exception e )
    {
        System.err.printf( "%s\n\n", e.getMessage() );
        e.printStackTrace();

        StackTraceElement[] pilhaDeElementos = e.getStackTrace();

        System.out.println( "\nRastreo da pilha a partir de getStackTrace:" );
        System.out.println( "Classe\tArquivo\t\tLinha\tMétodo" );

        for ( StackTraceElement elemento : pilhaDeElementos )
        {
            System.out.printf( "%s\t", elemento.getClassName() );
            System.out.printf( "%s\t", elemento.getFileName() );
            System.out.printf( "%s\t", elemento.getLineNumber() );
            System.out.printf( "%s\n", elemento.getMethodName() );
        }
    }
}

public static void metodo1() throws Exception
{
    metodo2();
}
```



Project Explorer

Exemplo1
Exemplo2
Exemplo3
Exemplo4
Exemplo5
JRE System Library [JavaSE-1.8]
src
(default package)
UsandoExcecoes3
Exemplo6

DivisaoPorZ...

```
20 {  
21     System.out.printf(  
22     System.out.printf(  
23     System.out.printf(  
24     System.out.printf(  
25 }  
26 }  
27 }  
28  
29 public static void metodo1()  
30 {  
31     metodo2();  
32 }  
33  
34 public static void metodo2()  
35 {  
36     metodo3();  
37 }
```

Problems @ Javadoc Declaration Console

<terminated> UsandoExcecoes3 [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\java.exe

Exceção lançada no metodo3

java.lang.Exception: Exceção lançada no metodo3
at UsandoExcecoes3.metodo3(UsandoExcecoes3.java:41)
at UsandoExcecoes3.metodo2(UsandoExcecoes3.java:36)
at UsandoExcecoes3.metodo1(UsandoExcecoes3.java:31)
at UsandoExcecoes3.main(UsandoExcecoes3.java:7)

Rastreo da pilha a partir de getStackTrace:

| Classe | Arquivo | Linha | Método |
|-----------------|----------------------|-------|---------|
| UsandoExcecoes3 | UsandoExcecoes3.java | 41 | metodo3 |
| UsandoExcecoes3 | UsandoExcecoes3.java | 36 | metodo2 |
| UsandoExcecoes3 | UsandoExcecoes3.java | 31 | metodo1 |
| UsandoExcecoes3 | UsandoExcecoes3.java | 7 | main |

<terminated> UsandoExcecoes3 [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\java.exe

Exceção lançada no metodo3

java.lang.Exception: Exceção lançada no metodo3
at UsandoExcecoes3.metodo3(UsandoExcecoes3.java:41)
at UsandoExcecoes3.metodo2(UsandoExcecoes3.java:36)
at UsandoExcecoes3.metodo1(UsandoExcecoes3.java:31)
at UsandoExcecoes3.main(UsandoExcecoes3.java:7)

Rastreo da pilha a partir de getStackTrace:

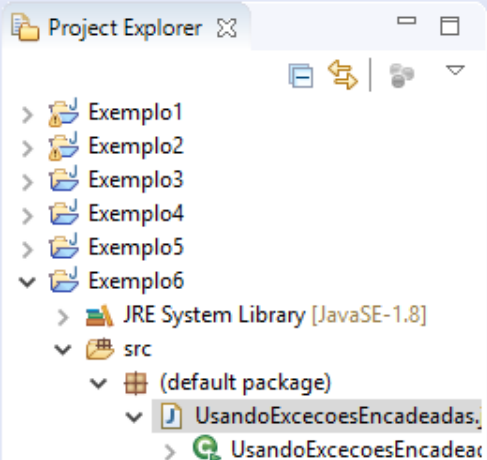
| Classe | Arquivo | Linha | Método |
|-----------------|----------------------|-------|---------|
| UsandoExcecoes3 | UsandoExcecoes3.java | 41 | metodo3 |
| UsandoExcecoes3 | UsandoExcecoes3.java | 36 | metodo2 |
| UsandoExcecoes3 | UsandoExcecoes3.java | 31 | metodo1 |
| UsandoExcecoes3 | UsandoExcecoes3.java | 7 | main |

Exemplo 6

```
public class UsandoExcecoesEncadeadas {
    public static void main( String[] args )
    {
        try
        {
            metodo1();
        }
        catch ( Exception e )
        {
            e.printStackTrace();
        }
    }

    public static void metodo1() throws Exception
    {
        try
        {
            metodo2();
        }
        catch ( Exception e )
        {
            throw new Exception( "Exceção lançada no metodo1", e );
        }
    }

    public static void metodo2() throws Exception
    {
        try
        {
            metodo3();
        }
        catch ( Exception exc )
```

```
1 public class UsandoExcecoesEncadeadas {
2     public static void main(String[] args) {
3         {
4             try
5             {
6                 metodo1();
7             }
8             catch (Exception e) {
9                 e.printStackTrace();
10            }
11        }
12    }
13    public static void metodo1() {
14        {
15            try
16            {
17                metodo2();
18            }
19            catch (Exception e) {
20                e.printStackTrace();
21            }
22        }
23    }
24    public static void metodo2() {
25        {
26            try
27            {
28                metodo3();
29            }
30            catch (Exception e) {
31                e.printStackTrace();
32            }
33        }
34    }
35    public static void metodo3() {
36        // ...
37    }
38 }
```

<terminated> UsandoExcecoesEncadeadas [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\java
java.lang.Exception: Exceção lançada no metodo1
at UsandoExcecoesEncadeadas.metodo1(UsandoExcecoesEncadeadas.java:23)
at UsandoExcecoesEncadeadas.main(UsandoExcecoesEncadeadas.java:7)
Caused by: java.lang.Exception: Exceção lançada no metodo2
at UsandoExcecoesEncadeadas.metodo2(UsandoExcecoesEncadeadas.java:35)
at UsandoExcecoesEncadeadas.metodo1(UsandoExcecoesEncadeadas.java:19)
... 1 more
Caused by: java.lang.Exception: Exceção lançada no metodo3
at UsandoExcecoesEncadeadas.metodo3(UsandoExcecoesEncadeadas.java:41)
at UsandoExcecoesEncadeadas.metodo2(UsandoExcecoesEncadeadas.java:31)
... 2 more

Problems @ Javadoc Declaration Console
<terminated> UsandoExcecoesEncadeadas [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\java.exe (6 de mai de 2019 00:10:27)
java.lang.Exception: Exceção lançada no metodo1
at UsandoExcecoesEncadeadas.metodo1(UsandoExcecoesEncadeadas.java:23)
at UsandoExcecoesEncadeadas.main(UsandoExcecoesEncadeadas.java:7)
Caused by: java.lang.Exception: Exceção lançada no metodo2
at UsandoExcecoesEncadeadas.metodo2(UsandoExcecoesEncadeadas.java:35)
at UsandoExcecoesEncadeadas.metodo1(UsandoExcecoesEncadeadas.java:19)
... 1 more
Caused by: java.lang.Exception: Exceção lançada no metodo3
at UsandoExcecoesEncadeadas.metodo3(UsandoExcecoesEncadeadas.java:41)
at UsandoExcecoesEncadeadas.metodo2(UsandoExcecoesEncadeadas.java:31)
... 2 more

Exercício

- Crie uma Classe CalculoMatematico
 - Nela, crie um método divisao, que recebe como parâmetros os valores a serem divididos. O retorno é o resultado da divisão (todos os números devem ser do tipo inteiro)
- Crie uma classe de teste para testar a CalculoMatematico
 - Nela crie um objeto CalculoMatematico e acesse o método divisao, tentando dividir 4 por 0.
- Execute a classe e veja o que acontece

Exercício

- Crie um bloco **try...catch** no método divisao para tratar a operação realizada
- No catch:
 - Informar o objeto do tipo ArithmeticException
 - Imprimir uma mensagem informando que a operação não pode ser realizada
 - Retornar zero

Exercício

- Tire o bloco **try...catch** do método divisao
- Adicione **throws ArithmeticException** na assinatura do método
- Na primeira linha do bloco do método, faça uma verificação se o divisor é igual a 0
 - Se for, lance uma exceção
 - **throw new ArithmeticException("Texto");**
- Na classe de teste, crie um bloco try...catch, tentando executar o método divisão
 - Catch para **ArithmeticException**
 - No bloco do Catch, imprima o método getMessage() do objeto criado do tipo ArithmeticException

Exercício

- Crie uma nova Classe
 - `DivisorZeroException`
 - Implemente da mesma forma do slide 19
- Na Classe `CalculoMatematico`, troque **`ArithmeticException`** por **`DivisorZeroException`**
 - `throws DivisorZeroException`
 - `throw new DivisorZeroException();`
- Na classe de teste, troque no Catch **`ArithmeticException`** por **`DivisorZeroException`**

Dúvidas?