



MaKe a NoTe oF THis!

# *Tratamento de exceções*

*Prof. Msc Gustavo Molina*

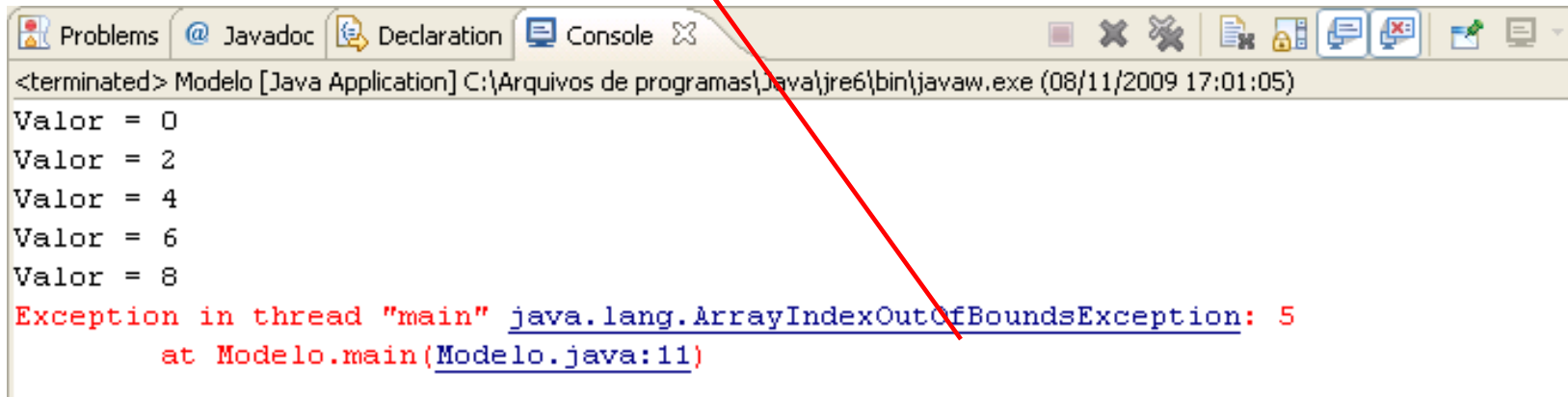
# Quando o JAVA encontra um erro...

```
public class Modelo {  
  
    public static void main(String[] args) {  
  
        int n1=10,n2=0;  
        int n3 = n1/n2;  
        System.out.println ("Resultado =" + n3);  
  
    }  
}
```



# Quando o JAVA encontra um erro...

```
public class Modelo {  
  
    public static void main(String[] args) {  
  
        int[] nro = new int[5];  
        for (int i=0;i<5;i++)  
            nro[i] = i*2;  
  
        for (int i=0;i<6;i++)  
            System.out.println("Valor = " + nro[i]);  
    }  
}
```



Problems Javadoc Declaration Console

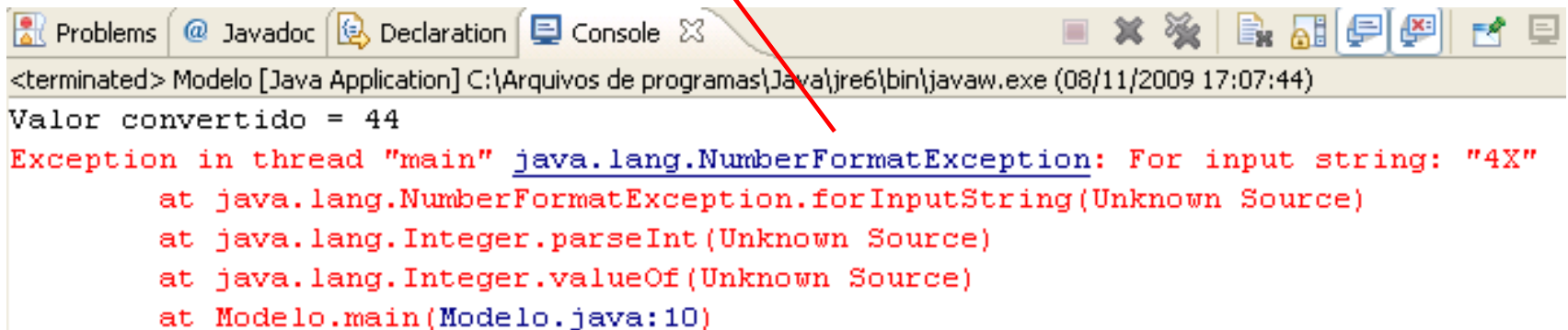
<terminated> Modelo [Java Application] C:\Arquivos de programas\Java\jre6\bin\javaw.exe (08/11/2009 17:01:05)

Valor = 0  
Valor = 2  
Valor = 4  
Valor = 6  
Valor = 8

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5  
at Modelo.main(Modelo.java:11)

# Quando o JAVA encontra um erro...

```
public class Modelo {  
  
    public static void main(String[] args) {  
  
        String texto = "44";  
        int nro = Integer.valueOf(texto);  
        System.out.println ("Valor convertido = " + nro);  
  
        texto = "4X";  
        nro = Integer.valueOf(texto);  
        System.out.println ("Valor convertido = " + nro);  
    }  
}
```



<terminated> Modelo [Java Application] C:\Arquivos de programas\Java\jre6\bin\javaw.exe (08/11/2009 17:07:44)

Valor convertido = 44

Exception in thread "main" java.lang.NumberFormatException: For input string: "4X"  
 at java.lang.NumberFormatException.forInputString(Unknown Source)  
 at java.lang.Integer.parseInt(Unknown Source)  
 at java.lang.Integer.valueOf(Unknown Source)  
 at Modelo.main(Modelo.java:10)

# *No mundo real....*



# Patroa “sem tratamento de exceções”

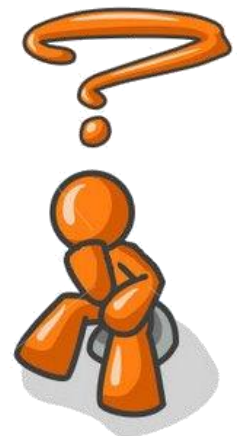
<i>Ordens da patroa</i>	Possível Problema	Consequência
Fazer o almoço	Acabar o gás	Hoje não tem almoço
Chegar às 7h30	Greve de ônibus	Empregada chega atrasada
Levar o cachorro para passear	Estar chovendo	Cachorro preso e estressado

# Patroa que “trata as exceções”

<i>Ordens da patroa</i>	Possível Problema	Tratamento da exceção
Fazer o almoço	Se... Acabar o gás	Pedir almoço no delivery
Chegar às 7h30	Se... Tiver greve de ônibus	Telefonar para patroa que ela vai buscá-la
Levar o cachorro para passear	Se... Estiver chovendo	Deixe o cachorro solto dentro da casa

# Em programação... como administrar os erros?

- Exibir mensagem de erro e sair
- Retornar um valor especial que indique a ocorrência de um determinado erro (por exemplo: -1)
- Usar uma variável global para administrar os erros através de códigos
- Usar as *exceptions*...





## *Em JAVA*

**Exceção** = possível erro e/ou condição anormal de funcionamento do programa.



**Tratar as exceções significa antecipar e lidar** com estes erros de forma a controlar as suas consequências.

**Tratar as exceções** deixa o software mais robusto, seguro e estruturado.

# Cozinhando sem tratar as exceções

```
cortar(cebola) ;  
panela.adicionar(cebola) ;  
cortar(tomate) ;  
panela.adicionar(tomate) ;  
panela.adicionar(Oleo.medida(colher)) ;  
  
comer() ;
```



**IMPREVISTO: CORTAR DEDO...**

# Idéia principal

Quando um erro acontece, uma exceção é lançada. Isso significa que o código que causou o erro tem a sua execução interrompida imediatamente **(PARA DE COZINHAR)** e o controle é transferido para o tratamento adequado deste erro **(APLICAR CURATIVO)**.

# Cozinhando com tratamento das exceções

```
tente
{
    cortar(cebola) ;
    panela.adicionar(cebola) ;
    cortar(tomate) ;
    panela.adicionar(tomate) ;
    panela.adicionar(Oleo.medida(colher)) ;
}

imprevisto(CortarDedo e)
{
    dedo.aplicar(curativo) ;
}

finalmente {
    comer() ;
}
```

# Em código JAVA...

```
try {  
    instruções  
}
```

Instruções normais do programa

```
catch ( <exception-class-name> <variable-name> ) {  
    instruções  
}
```

Instruções se um determinado ocorrer

```
finally {  
    instruções  
}
```

Instruções após gerenciamento do erro

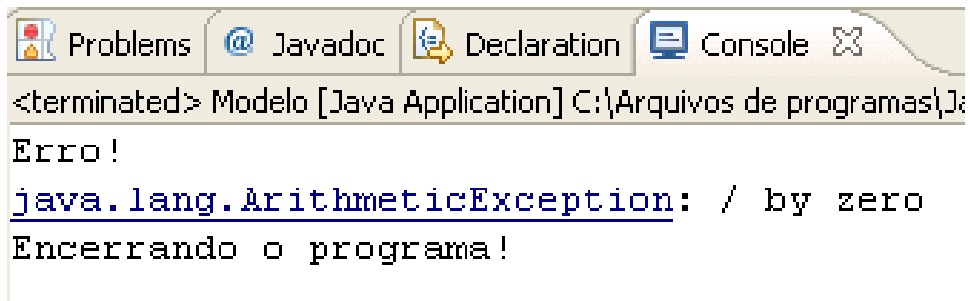
# Comando `try...catch...finally...`

- O comando `try` pode ter mais de uma cláusula `catch` e opcionalmente pode ter a cláusula `finally`
- Cada comando `try` deve ter uma das duas opções: `catch` ou `finally`
- A cláusula `finally` indica um grupo de instruções que serão executadas como última etapa do comando `try` (acontecendo ou não a exceção)

CORTANDO OU NÃO CORTANDO O DEDO, EU VOU COMER!!

# Tratando o erro (I)

```
public class Modelo {  
  
    public static void main(String[] args) {  
  
        int n1=10,n2=0;  
        try {  
            int n3 = n1/n2;  
            System.out.println ("Resultado = " + n3);  
        }  
        catch (Exception e){  
            System.out.println ("Erro!");  
            System.out.println (e);  
        }  
        finally {  
            System.out.println ("Encerrando o programa!");  
        }  
    }  
}
```



# Tratando o erro (II)

```
public class Modelo {  
    public static void main(String[] args) {
```

```
        try {  
            int n1 = Integer.valueOf(args[0]);  
            int n2 = Integer.valueOf(args[1]);  
            System.out.println ("Resultado = " + n1/n2);  
        }
```

```
        catch (ArrayIndexOutOfBoundsException e1){  
            System.out.println ("Erro na quantidade de argumentos");  
        }  
        catch (NumberFormatException e2){  
            System.out.println ("Argumento não é um número inteiro");  
        }  
        catch (ArithmeticException e3){  
            System.out.println ("Erro...Divisão por zero");  
        }
```

```
        finally {  
            System.out.println ("Encerrando o programa!");  
        }
```

```
    }
```

```
}
```



# Primeiro teste:

Parâmetros fornecidos: 20 10

**Execução OK**

```
<terminated> Modelo [Java Application] C:\Arquivos de pr  
Resultado = 2  
Encerrando o programa!
```

# Segundo teste:

Parâmetros fornecidos: 20 A

**Erro**

```
<terminated> Modelo [Java Application] C:\Arquivos de programas\Java\j  
Argumento fornecido não é um número inteiro  
Encerrando o programa!
```

# Terceiro teste:

Parâmetros fornecidos: 20

**Erro**

```
<terminated> Modelo [Java Application] C:\Arquivos de programas\Java\  
Erro na quantidade de argumentos fornecidos  
Encerrando o programa!
```

## Quarto teste:

Parâmetros fornecidos: 20 0

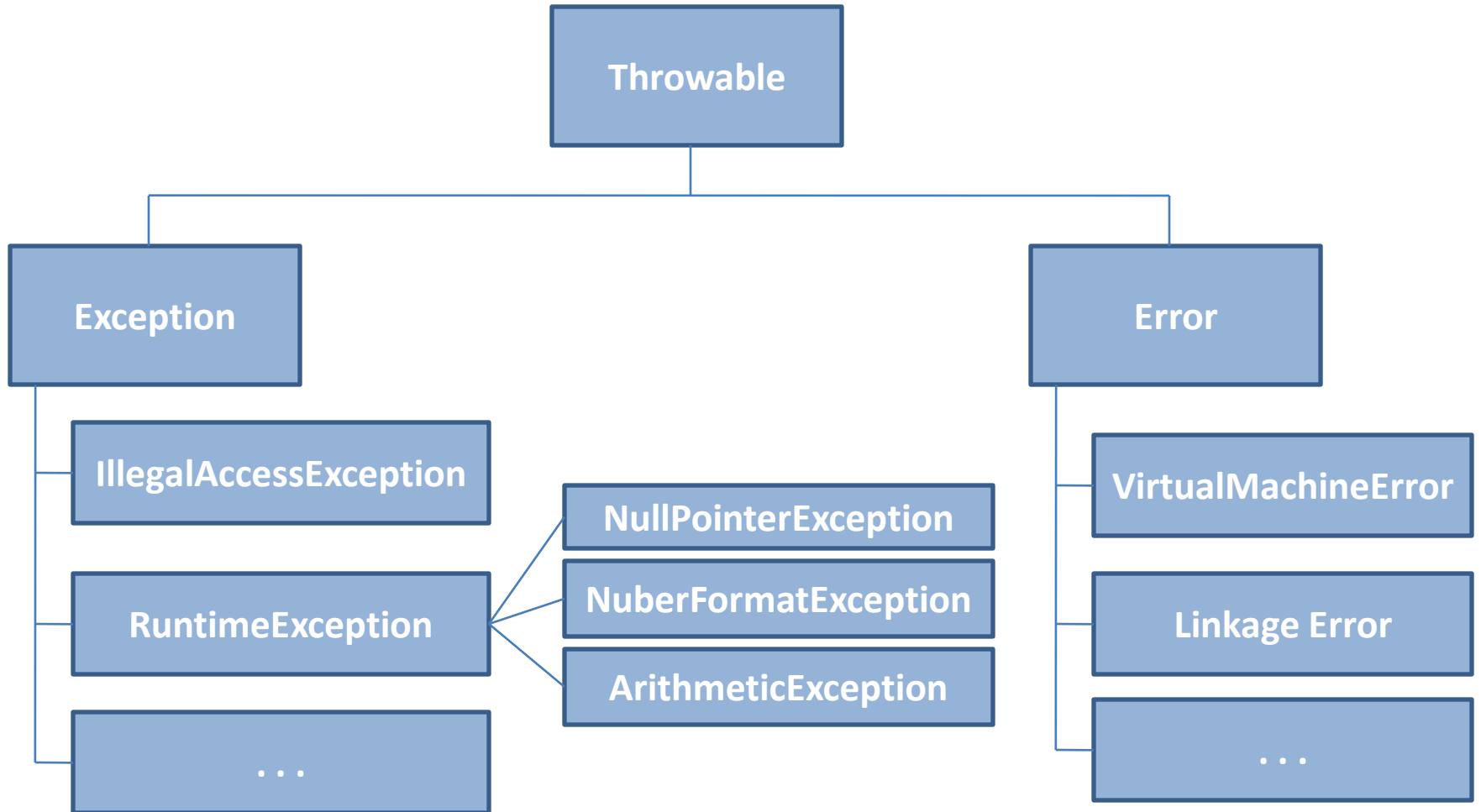
**Erro**

```
<terminated> Modelo [Java Application] C:\Arquivo
```

```
Erro...Divisão por zero
```

```
Encerrando o programa!
```

# Classes de exceção em JAVA



## Portanto...

- Uma exceção é um objeto (alguns métodos: `getMessage()`, `getClass()`, etc.)
- A superclasse de todas as classes responsáveis pelo tratamento de erros é a classe `Throwable`
- As classes `Exception` e `Error` são suas subclasses diretas
- A classe `Error` deve ser usada para tratar erros mais graves, como por exemplo o mau funcionamento de dispositivos de hardware (falta de memória)
- A classe `Exception` é a superclasse de todas as exceções possíveis durante a execução de um programa, inclusive as exceções criadas para atender necessidades específicas do programador

# Subclasses de RuntimeException

Exceção	Significado
<code>ArithmeticException</code>	Erros aritméticos tais como divisão por zero
<code>ArrayIndexOutOfBoundsException</code>	Índice do vetor está fora dos limites aceitáveis
<code>ArrayStoreException</code>	Atribuição para um elemento de um vetor de um tipo incompatível
<code>ClassCastException</code>	Inválido Cast
<code>IllegalArgumentException</code>	Argumento ilegal na chamada de um método
<code>IndexOutOfBoundsException</code>	Algum índice está fora dos limites aceitáveis
<code>NegativeArraySizeException</code>	Vetor criado com tamanho negativo
<code>NullPointerException</code>	Uso inválido de uma referência null
<code>NumberFormatException</code>	Conversão inválida de uma String para um formato numérico
<code>StringIndexOutOfBounds</code>	Tentativa de indexação fora dos limites de uma String

# Criando suas próprias exceções (I)

- O programador pode estender a classe `Exception` ou uma das suas subclasses para construir as suas próprias exceções.
- Uma exceção é lançada usando-se a palavra chave `throw` seguida da referência à exceção. Exemplo:

```
Exception opa = new Exception("deu zebra");  
...  
if (temProblema)  
    throw opa;
```





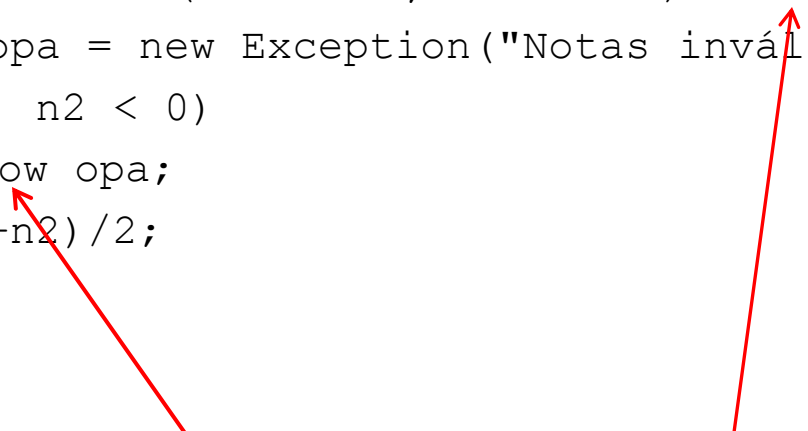
# Criando suas próprias exceções (II)

- Se o programador desejar que a exceção lançada seja tratada fora do método que a gerou, ele deve explicitar isto usando a palavra chave `throws` seguida do tipo de exceção, na declaração do método.

```
TipoDeRetorno nomeDoMetodo( ) throws Exception {  
  
    Exception opa = new Exception("deu zebra");  
    ...  
    if (temProblema)  
        throw opa;  
}
```

# Criando suas próprias exceções (III)

```
public class Notas {  
  
    public float calcMedia(float n1, float n2) throws Exception {  
        Exception opa = new Exception("Notas inválidas");  
        if (n1<0 || n2 < 0)  
            throw opa;  
        return (n1+n2)/2;  
    }  
}
```



**NÃO CONFUNDA THROW COM THROWS**

# Criando suas próprias exceções (IV)

```
try {  
    float media = minhas.calcMedia(nota1,nota2);  
    System.out.println("Media Final:" + media);  
}  
catch (Exception e){  
    System.out.println(e.getMessage());  
}
```

Como o método `calcMedia()` tem em sua definição o lançamento da exceção, caso sejam passados parâmetros negativos, o `catch()` pegará o erro e exibirá a mensagem “Notas inválidas”

You just *had* to divide  
by zero, didn't you?

It's okay!  
I wrote an  
exception!

