

Objetivos: Aprender a como realizar o tratamento de erros;  
Formatar a exibição de números com casas decimais;

Recursos: Aula em laboratório com uso do computador; Quadro e pincel; Roteiro de Aula;

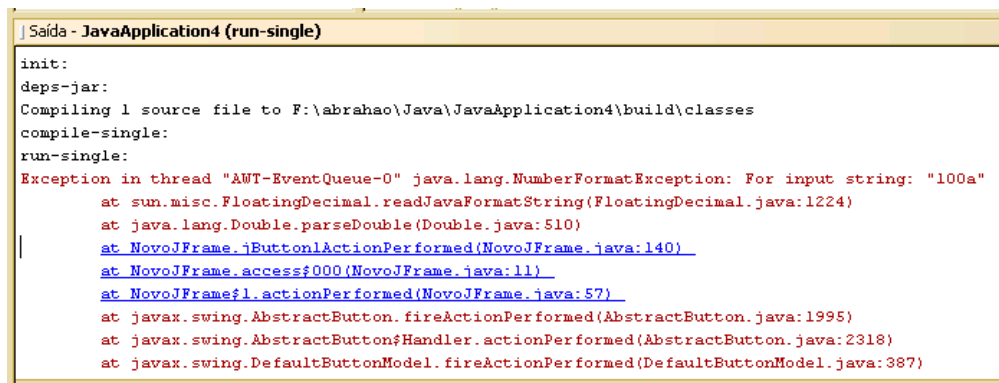
## TRATAMENTO DE ERROS (EXCEÇÕES)

Durante a execução de um programa, muitas vezes o usuário realiza uma operação que foi prevista pelo programador como sendo um erro de utilizador, ou mesmo algo que não foi previsto durante a implementação do programa. Normalmente quando tais erros acontecem, o programa é finalizado ou perde sua funcionalidade devido à quebra da seqüência de comandos. Para evitar problemas como esse e ainda poder exibir mensagens de alerta ao usuário, usamos o conjunto de comandos try / catch. Observe o código abaixo inserido num botão qualquer:

### Arquivo: Erros1.java

```
1. String s = "100a";  
2. Double numero = Double.parseDouble(s);
```

Na linha 1 do código acima definimos uma variável s com valor "100a". Na linha seguinte tentamos converter o valor de s para uma variável Double. Entretanto o programa vai retornar uma mensagem de erro no console semelhante a esta:



```
| Saída - JavaApplication4 (run-single)  
init:  
deps-jar:  
Compiling 1 source file to F:\abrahamo\Java\JavaApplication4\build\classes  
compile-single:  
run-single:  
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "100a"  
    at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1224)  
    at java.lang.Double.parseDouble(Double.java:510)  
    at NovoJFrame.jButton1ActionPerformed(NovoJFrame.java:140)  
    at NovoJFrame.access$000(NovoJFrame.java:11)  
    at NovoJFrame.j1.actionPerformed(NovoJFrame.java:57)  
    at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1995)  
    at javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2318)  
    at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:387)
```

Por existir uma letra junto dos dígitos, o número não pode ser convertido e gera uma exceção. Para avisar ao usuário sobre esse erro, podemos usar try / catch, conforme ilustrado no código abaixo.

### Arquivo: Erros2.java

```
1. String s = "100a";  
2. try {  
3.     Double numero = Double.parseDouble(s);  
4. }catch(Exception e){  
5.     JOptionPane.showMessageDialog(null,"Erro na conversão");  
6. }
```

Neste segundo caso, estamos dizendo ao programa *tente* (try) fazer o código a seguir e *capture* (catch) qualquer erro que acontecer. Assim, executamos linha a linha o que está no bloco try, e se por acaso algo inesperado ocorrer, o código salta automaticamente para a exibição da mensagem "Erro na conversão" que está escrito em catch.

Devemos usar essa construção em praticamente todos os códigos que estejam sujeitos à erros. Java inclusive requer que operações como gravação de arquivos, acesso à banco de dados, e outras sejam obrigatoriamente escritas entre try/catch.

O termo (**Exception e**) cria um objeto chamado 'e' que armazena as informações sobre o erro sem exibi-la no console. Caso deseje mostrar essas informações ao usuário, use **e.printStackTrace()** dentro do bloco catch.

Outros tipos de erros comuns:

**catch(NumberFormatException e)** – É ativada quando uma string tenta ser convertida para um valor mas esta não possui o formato correto (ex. converter '100a' para Double);

**catch(ArithmeticException e)** – É ativada quando um erro matemático ocorre, como uma divisão por zero (ex. int x = 10/0).

**catch(FileNotFoundException e)** – Usada para capturar erros ao tentar abrir um arquivo que não existe.

Esses são apenas alguns exemplos, existem centenas de exceções que podem ser usadas.

É possível capturar mais de um tipo de erro usando vários catch na construção do código. Ex.:

**Arquivo:** Erros3.java

```
1. String s = "100a";
2. try {
3.     Double numero = Double.parseDouble(s);
4. }catch(FileNotFoundException e){
5.     JOptionPane.showMessageDialog(null,"Arquivo não encontrado");
6. }catch(NumberFormatException e){
7.     JOptionPane.showMessageDialog(null,"Número digitado inválido");
8. }
```

## FORMATANDO A EXIBIÇÃO DE NÚMEROS COM CASAS DECIMAIS

Em alguns programas anteriores que fizemos, ao exibir o resultado de um cálculo, aparecem várias casas decimais onde normalmente só queremos duas. Para contornar esse problema, podemos usar a classe **DecimalFormat** que cria um objeto capaz de modificar a exibição do resultado. Observe o código abaixo usado para exibir duas casas decimais.

```
Double d = 2.1245;
```

```
DecimalFormat decimalFormat = new DecimalFormat("#.00");
```

```
JOptionPane.showMessageDialog(null, "" + decimalFormat.format(d));
```

Explore a documentação dessa classe para conhecer mais detalhes.

<http://java.sun.com/javase/6/docs/api/java/text/DecimalFormat.html>

## EXERCÍCIOS

1. Modifique o programa da lanchonete adicionando o tratamento de exceções, para que se o usuário esquecer-se de colocar a quantidade em algum dos campos do recheio, o programa deve pedir ao usuário que digite essa informação.
2. Modifique o mesmo programa de forma que o valor total seja exibido apenas com duas casas decimais.
3. Crie uma pequena calculadora que receba dois números do usuário e exiba o resultado. Faça o tratamento de exceções com mensagens individuais para erros aritméticos (divisão por zero) e erros de digitação (formato do número).