

Fundamentos (parte 2)



Prof. Dr. João Paulo Lemos Escola

Copyright© 2022

Tópicos da aula

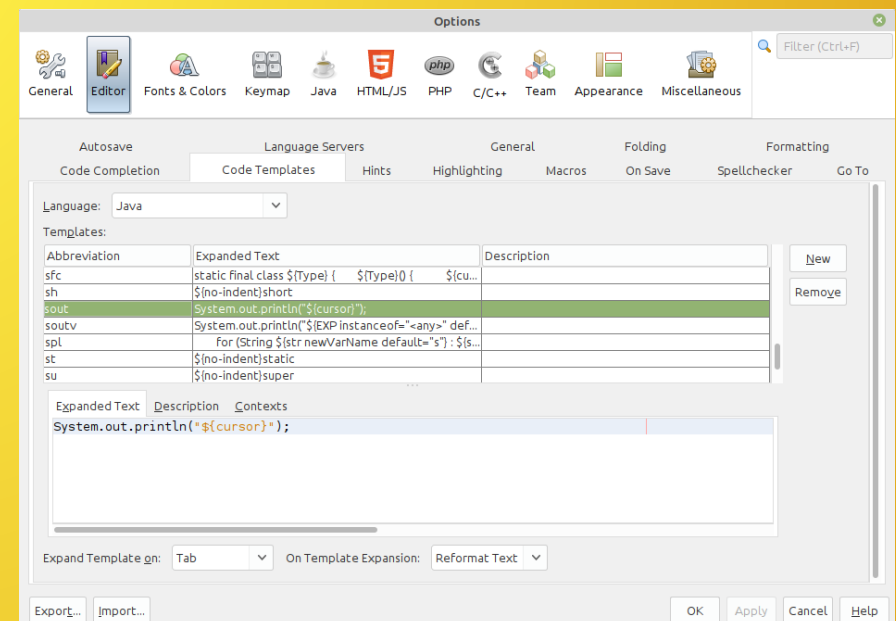
- Modelos de códigos, autocompletar e atalhos;
- Entrada de dados;
- Exceções;
- Janelas gráficas;
- Sobrecarga de métodos;
- Estruturas de decisão.

NetBeans: Dicas úteis

- O editor do NetBeans possibilita aumentar a produtividade do programador;
- As dicas e auxílios do NetBeans, além de alguns atalhos podem ser úteis na hora de programar;
- A seguir apresentaremos algumas dicas para melhorar a sua vida com o NetBeans:

NetBeans: modelos de código

- Menu Tools > options > Editor > Code Templates;
- Digite uma das palavras da lista e pressione TAB para gerar o código correspondente:
 - **sout** gera um `System.out.println("");`;



Exemplos de modelos de código

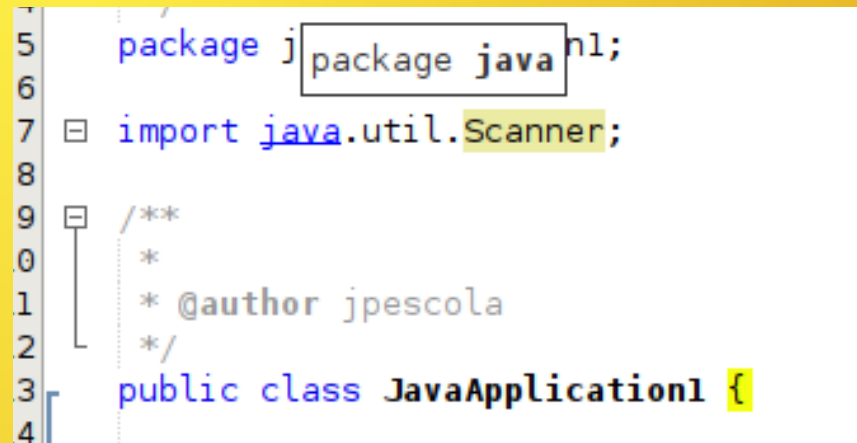
- **sout** = `System.out.println`
- **trycatch** gera a cláusula try-catch pronta
- **wh** = gera uma cláusula while
- **for** = gera uma cláusula for

NetBeans: teclas de atalho

- CTRL+SHIFT+ para baixo
 - Duplica a linha atual
- SHIFT+DEL
 - Exclui a linha atual
- CTRL+espaço
 - Pede ajuda para o NetBeans (mostra javadoc ou uma sugestão)
- SHIFT+ALT+F
 - formata o código
- SHIFT+ALT+ para cima ou para baixo
 - Move a linha atual
- SHIFT+ALT+ direita ou esquerda
 - Indenta o código para a direita ou esquerda

CTRL+cursor sobre a palavra

- Segure o CTRL e mova o cursor sobre as palavras-chave;
- Quando possível, o Netbeans mostrará uma dica sobre cada item que você apontar:



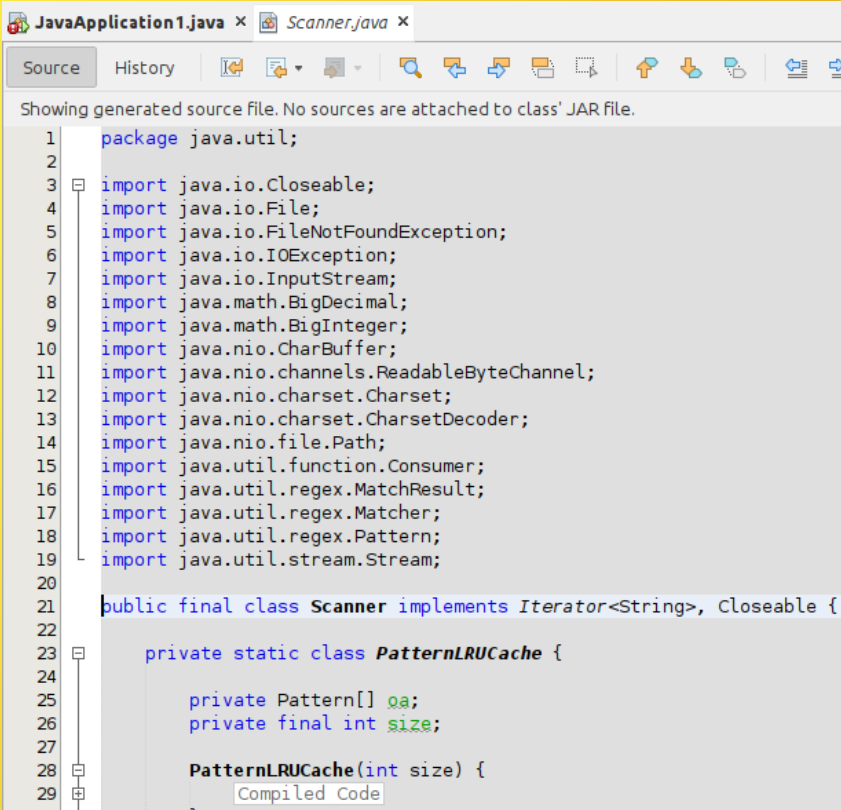
The screenshot shows a Java code editor with the following code:

```
5 package java;
6
7 import java.util.Scanner;
8
9 /**
10  *
11  * @author jpescola
12  */
13 public class JavaApplication1 {
14
```

A tooltip is displayed over the word 'package' on line 5, showing the text 'package java'.

CTRL+Click

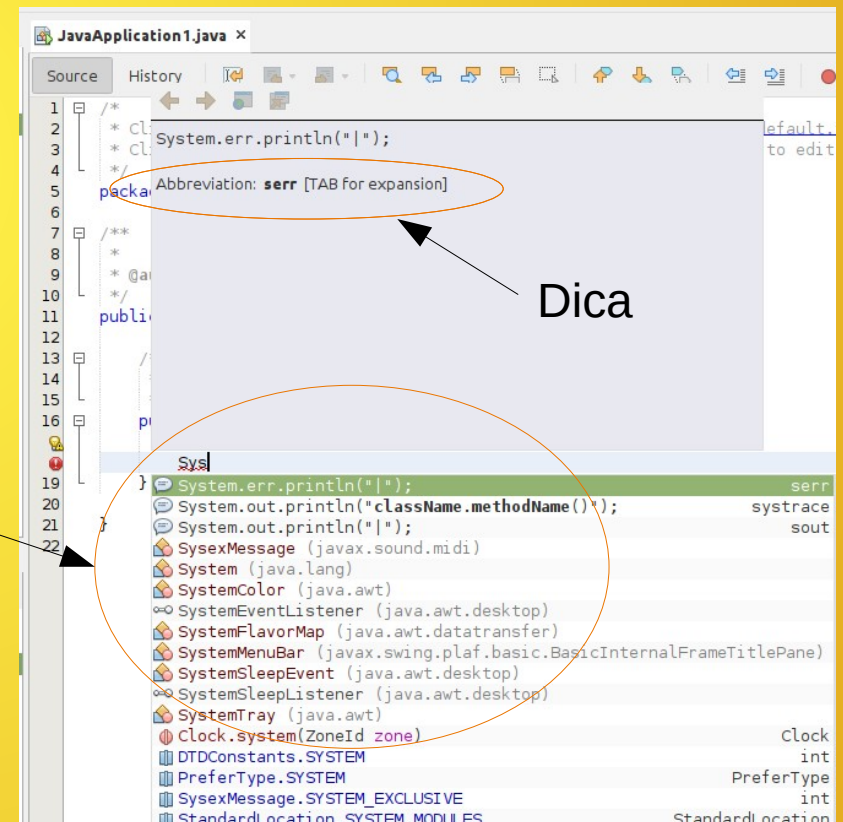
- Uma importante dica para uso do Netbeans;
- Segure a tecla CTRL e clique em uma classe ou variável para ser direcionado à sua biblioteca ou declaração:



```
JavaApplication1.java x Scanner.java x
Source History
Showing generated source file. No sources are attached to class' JAR file.
1 package java.util;
2
3 import java.io.Closeable;
4 import java.io.File;
5 import java.io.FileNotFoundException;
6 import java.io.IOException;
7 import java.io.InputStream;
8 import java.math.BigDecimal;
9 import java.math.BigInteger;
10 import java.nio.CharBuffer;
11 import java.nio.channels.ReadableByteChannel;
12 import java.nio.charset.Charset;
13 import java.nio.charset.CharsetDecoder;
14 import java.nio.file.Path;
15 import java.util.function.Consumer;
16 import java.util.regex.MatchResult;
17 import java.util.regex.Matcher;
18 import java.util.regex.Pattern;
19 import java.util.stream.Stream;
20
21 public final class Scanner implements Iterator<String>, Closeable {
22
23     private static class PatternLRUCache {
24
25         private Pattern[] oa;
26         private final int size;
27
28         PatternLRUCache(int size) {
29             Compiled Code
30         }
31     }
32 }
```


Auto completar

- Quando o Netbeans sugere um trecho de código, basta selecionar e pressionar Enter para auto completar;
- Para chamar o auto completar, utilize as teclas CTRL+espaço:

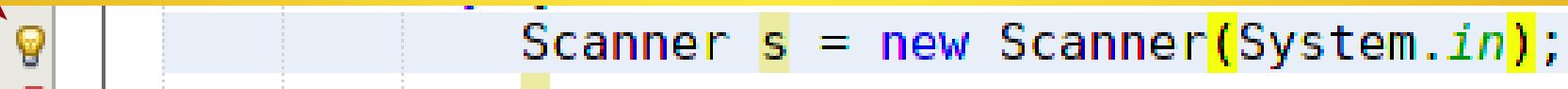



Entrada de dados

- Além do uso de argumentos, que são suportados por padrão nas aplicações Java, podemos trabalhar com entradas de dados em tempo de execução;
- Existem várias classes para entrada de dados, mas a classe **Scanner** é a mais simples e evoluída delas.

NetBeans: a lâmpada

- Quando aparecer a lâmpada ao lado da linha atual, clique nela e veja a sugestão de código do Netbeans;
- Escolhendo a sugestão, ele irá executar o ajuste do seu código automaticamente:



```
Scanner s = new Scanner(System.in);
```

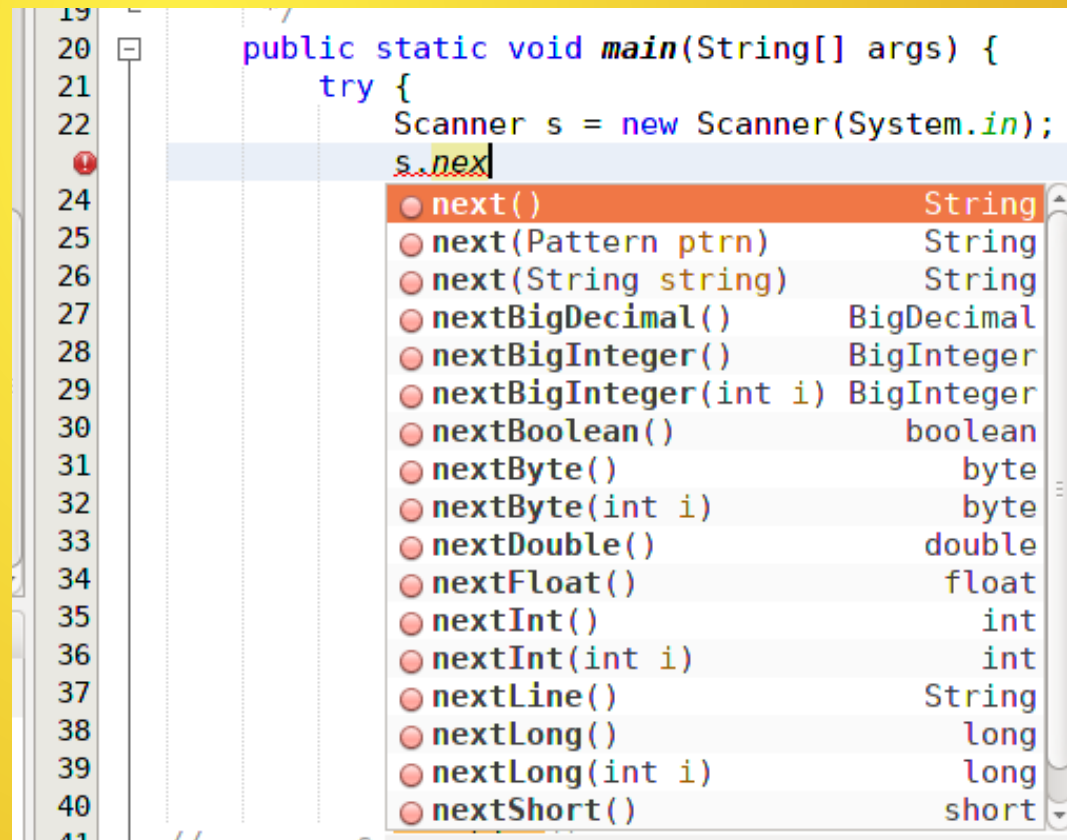
Classe Scanner

(A03ex01.jar)

```
import java.util.Scanner; // importando a classe Scanner
public static void main(String[] args){
    System.out.println("Digite a nota: ");
    // instancia o objeto 's' do tipo Scanner
    Scanner s = new Scanner(System.in);
    // aguarda a digitação, converte o que foi digitado para
    float e armazena em 'nota'
    float nota = s.nextFloat();
    System.out.println("nota digitada: "+nota);
}
```

Métodos disponíveis

- Veja que podemos receber dados em diversos formatos pela classe Scanner, utilizando o método correspondente:

A screenshot of an IDE showing a Java code snippet. The code is inside a `main` method. A `Scanner` object named `s` is created with `System.in` as the source. The cursor is positioned on `s.next()`, and a dropdown menu is open, listing various methods of the `Scanner` class. The methods are listed with their return types. The first method, `next()`, is highlighted in orange. The list includes methods for parsing strings, integers, longs, floats, doubles, booleans, bytes, and shorts.

```
19  
20 public static void main(String[] args) {  
21     try {  
22         Scanner s = new Scanner(System.in);  
23         s.next()  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41
```

<code>next()</code>	<code>String</code>
<code>next(Pattern ptrn)</code>	<code>String</code>
<code>next(String string)</code>	<code>String</code>
<code>nextBigDecimal()</code>	<code>BigDecimal</code>
<code>nextBigInteger()</code>	<code>BigInteger</code>
<code>nextBigInteger(int i)</code>	<code>BigInteger</code>
<code>nextBoolean()</code>	<code>boolean</code>
<code>nextByte()</code>	<code>byte</code>
<code>nextByte(int i)</code>	<code>byte</code>
<code>nextDouble()</code>	<code>double</code>
<code>nextFloat()</code>	<code>float</code>
<code>nextInt()</code>	<code>int</code>
<code>nextInt(int i)</code>	<code>int</code>
<code>nextLine()</code>	<code>String</code>
<code>nextLong()</code>	<code>long</code>
<code>nextLong(int i)</code>	<code>long</code>
<code>nextShort()</code>	<code>short</code>

Exceções

- Em java, podemos tratar exceções com a cláusula **try/catch**;
- Ela direciona a execução para o código de **catch** toda vez que o código dentro de **try** retornar uma exceção;

```
try{  
    // tenta executar este código  
}  
catch(Exception e){  
    /* executa isto caso aconteça um  
    problema no código anterior */  
}
```

Classe Scanner com try/catch

(A03ex02.jar)

```
import java.util.Scanner;
public static void main(String[] args){
    try{
        System.out.println("Digite a nota: ");
        Scanner s = new Scanner(System.in);
        float nota = s.nextFloat();
        System.out.println("nota digitada: "+nota);
    }
    catch (Exception e){
        System.out.println("Houve um erro. Digite apenas números!");
        //e.printStackTrace();// imprime a mensagem de erro padrão
    }
}
```


Criando uma Exceção

- Podemos criar nossa própria exceção usando o comando throw (lançar):

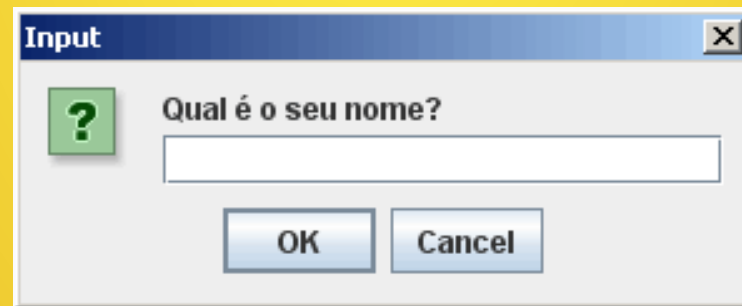
```
if (x > 100) {  
    throw new Exception("Numero muito grande");  
}  
} catch (Exception ex) {  
    System.out.println(ex.getMessage());  
}
```


A03ex03

- A partir do `printStackTrace` é possível efetuar testes e gerar diversas exceções;
- Crie um programa que receba dois valores **por argumentos** e valide três tipos diferentes de exceções;
- Dica: utilize o recurso 'auto completar' (CTRL+espaço) para descobrir as classes de exceções disponíveis em Java.

Janelas gráficas

- Vamos criar nossa primeira aplicação gráfica com Java;
- Utilizaremos a classe JOptionPane;
- Esta classe permite criar caixas do tipo sim/não, ok/cancelar entre outras.



Classe JOptionPane

(A03ex04.jar)

```
import javax.swing.JOptionPane;
public static void main(String[] args){
    try{
        String valor = JOptionPane.showInputDialog("Digite a nota:");
        float nota = Float.parseFloat(valor);
        JOptionPane.showMessageDialog(null, "Nota digitada: "+nota);
    }
    catch (Exception e){
        JOptionPane.showMessageDialog(null, "Houve um erro. Digite apenas números!");
        //e.printStackTrace();// imprime a mensagem de erro padrão
    }
}
```

Sobrecarga

- showMessageDialog(Component parentComponent, Object message):
 - Recebe dois parâmetros:
 - parentComponent é a janela “pai”, que será travada até que a mensagem seja fechada;
 - message é a mensagem que será exibida.

Sobrecarga (cont.)

- `showMessageDialog(Component parentComponent, Object message, String title, int messageType):`
 - Recebe quatro parâmetros:
 - `parentComponent` é a janela “pai”, que será travada até que a mensagem seja fechada;
 - `message` é a mensagem que será exibida;
 - `title` é o título da janela;
 - `messageType` é um número inteiro que representa o ícone que será exibido na janela:
 - `JOptionPane.ERROR_MESSAGE; // 0`
 - `JOptionPane.INFORMATION_MESSAGE; //1`
 - `JOptionPane.WARNING_MESSAGE; //2`
 - `JOptionPane.QUESTION_MESSAGE; //3`

Operadores lógicos

- Igual:

$x == y$

- Diferente:

$x != y$

- Maior que:

$x > y$

- Maior ou igual a:

$x >= y$

- Menor que:

$x < y$

- Menor ou igual a:

$x <= y$

Estruturas condicionais

- Permitem o desvio da execução do programa de acordo com as condições:
 - if / else if / else;
 - switch / case;

if / else if / else

(A03ex05.jar)

```
public static void main(String[] args){  
    int nota = 10;  
    if (nota == 10){  
        System.out.println("Parabéns!");  
    }  
    else if (nota >= 5){  
        System.out.println("Regular...");  
    }  
    else{  
        System.out.println("Estude mais!");  
    }  
}
```


Método “equals”

- Não é possível fazer comparação de Strings utilizando os operadores lógicos;
- Para comparação de Strings utilizamos o método **equals**:
 - já que uma variável do tipo String é, na verdade, um objeto da classe String;

equals

(A03ex06.jar)

```
import javax.swing.*;
public static void main(String[] args){
    String escolha = JOptionPane.showInputDialog("Digite V
    para verdadeiro e F para falso.");
    if (escolha.equals("V")||escolha.equals("F")){
        JOptionPane.showMessageDialog(null, "Escolha:
        "+escolha);
    }
    else
        JOptionPane.showMessageDialog(null, "Erro");
}
```

Switch / case

- Utilizada também para desvio de execução baseado em condição;
- Mais utilizado para casos onde há muitas opções e o uso do if/else resultaria em mais código.

Exemplo de switch/case

```
(A03ex07.jar)
public static void main(String[] args){
    if (args.length > 0){ // verifica se ha argumentos
        String info="";
        switch (args[0]){
            case "1" : info="Domingo"; break;
            case "2" : info="Segunda-feira"; break;
            case "3" : info="Terça-feira"; break;
            case "4" : info="Quarta-feira"; break;
            case "5" : info="Quinta-feira"; break;
            case "6" : info="Sexta-feira"; break;
            case "7" : info="Sábado"; break;
            default: info="Argumento inválido!");
        }
        System.out.println(info);
    }
    else{
        System.out.println("Argumento numérico obrigatório!");
    }
}
```

O que aprendemos?

- Modelos de códigos, autocompletar e atalhos;
- Entrada de dados;
- Exceções;
- Janelas gráficas;
- Sobrecarga de métodos;
- Estruturas de decisão.

Na próxima aula...

- Nesta aula vamos falar das estruturas de repetição;
- Vamos conhecer as classes Math e DecimalFormat;
- Também iremos conhecer novos métodos da classe String e aprender a trabalhar com expressões regulares.