Material desenvolvido por Thiago Cury

Aula 2 – LINGUAGEM DE PROGRAMAÇÃO VISUAL II

JavaDoc

Javadoc é uma ferramenta para elaboração de documentação para os pacotes, classes, atributos e métodos. Em suma, o javadoc é um gerador de documentação que a Sun Microsystems criou para documentar todo o código.

Para iniciar um JavaDoc utiliza-se **barra seguido de 2 asteriscos** e para fechar **1 asterico seguido de barra**, conforme as linhas logo abaixo:

/**
*/

Devemos iniciar a linha com /**
Encerrar o JavaDoc com */

Após iniciar o JavaDoc podemos utilizar diretrizes para informar o que deve ser incluído na documentação gerada por padrão em HTML.
Para inserir um diretriz utilizamos o caracter especial @ (arroba).

Algumas diretrizes que podemos utilizar no JAVADOC:

@author
Informa o nome de quem fez o código

@param
Explica o parâmetro, deve ser criado um @param para cada parâmetro recebido pelo método.

@return
Explica o que será retornado pelo método.

@since
Esse é para controle de versão, aqui você informa quando o método foi adicionado na Classe, ou quando a Classe foi adicionada ao pacote.

@version
Aqui você pode indicar a versão da Classe ou do método.

@category
Você pode criar categorias para seus códigos, como “utilidades” ou “manipulação” ou services”, etc.

@deprecated
Informa se esse método está depreciado, ou seja, irá deixar de existir na próxima versão do programa

@exception
Informa se o método retorna ou trata algum tipo de exceção.

@throws
Similar a @exception

@see
Cria um link de referência para outra Classe ou método.
{@link XY}
Cria um link para uma classe ou método, o nome da classe ou método deve ser colocado no lugar das letras "XY".

Exemplo de utilização de JavaDoc

```
/**
* @author Thiago Cury
* @version 1.0.1 Sagat
* @since 01/01/1900
*/
public class TesteJavaDoc{
	private double media;

	/**
	* @author Thiago Cury
	* @version 1.0.1
	* @param nota1, variável do tipo double que receberá a primeira nota do aluno
	* @param nota2, variável do tipo double que receberá a segunda nota do aluno
	*/
	public void calcularMedia(double nota1, double nota2){
		media = (nota1 + nota2)/2;
	}

	/**
	* @author Thiago Cury
	* @version 1.0.1
	* @return Retorna o atributo com a média do aluno.
	*/
	public double getMedia(){
		return media;
	}
}
```

Classe JOptionPane

Bom, chegou a hora de deixar definitivamente a Classe Scanner (API) de lado. Vamos começar a

utilizar as primeiras janelas gráficas. Para isto, vamos utilizar uma API do Java. O nome da Classe que utilizaremos é JOptionPane. Esta Classe deve ser importada para que possamos utilizá-la.

Exemplo de código utilizado para importação da API:
import javax.swing.JOptionPane;

obs: As janelas da Classe JOptionPane são do tipo Modal. Mas o que isso quer dizer? Quando aparecer uma dessas janelas na tela, o usuário não poderá realizar nenhuma outra operação antes de encerrar a janela.

Abaixo segue as explicações, códigos e figuras referente as janelas da Classe JoptionPane.

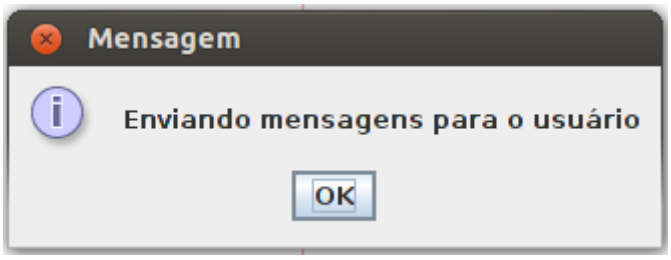
showMessageDialog

O método showMessageDialog constrói uma janela para enviar informações para o usuário. Nos parametros do método podemos passar as seguintes informações: janela pai, mensagem, título da janela e tipo de ícone.

Nesse primeiro código apenas foi enviada uma mensagem para o usuário.

Código:
```
JOptionPane.showMessageDialog(
                null,
                "Enviando mensagens para o usuário");
```

Visualização código:



O ícone de **"informação"** é utilizado por padrão, mas pode ser trocado por 4 possíveis ícones, são eles:

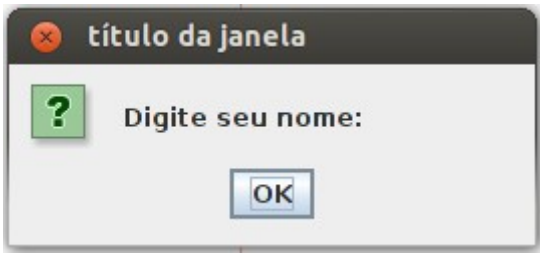
ERROR_MESSAGE
INFORMATION_MESSAGE
WARNING_MESSAGE
QUESTION_MESSAGE

Para **não** mostrar nenhum ícones utilizamos o código:

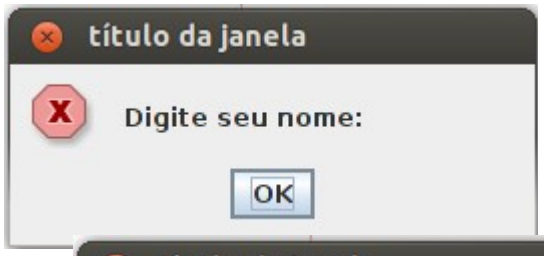
PLAIN_MESSAGE

Exemplos de JOptionPane.showMessageDialog() com os ícones:
Obs.: Para colocar ícones somos obrigados a colocar o título da janela no código conforme os códigos abaixo:

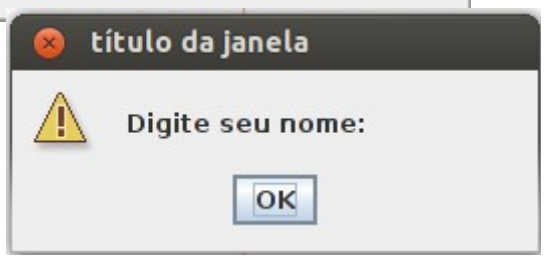
```
JOptionPane.showMessageDialog(
        null,
        “Digite seu nome:”,
        “título da janela”,
        JOptionPane.QUESTION_MESSAGE);
```



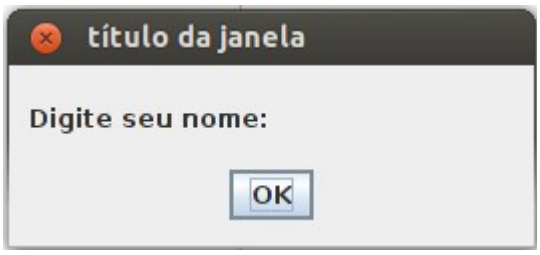

```
JOptionPane.showMessageDialog(
        null,
        “Digite seu nome:”,
        “título da janela”,
        JOptionPane.ERROR_MESSAGE);
```



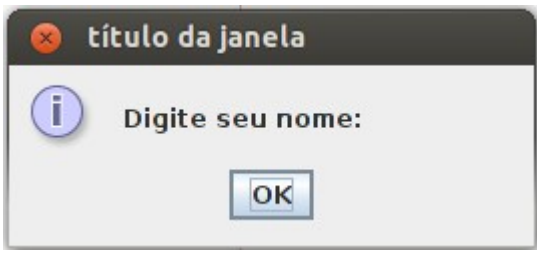

```
JOptionPane.showMessageDialog(
        null,
        “Digite seu nome:”,
        “título da janela”,
        JOptionPane.WARNING_MESSAGE);
```




```
JOptionPane.showMessageDialog(
        null,
        “Digite seu nome:”,
        “título da janela”,
        JOptionPane.PLAIN_MESSAGE);
```




```
JOptionPane.showMessageDialog(
        null,
        “Digite seu nome:”,
        “título da janela”,
        JOptionPane.INFORMATION_MESSAGE);
```



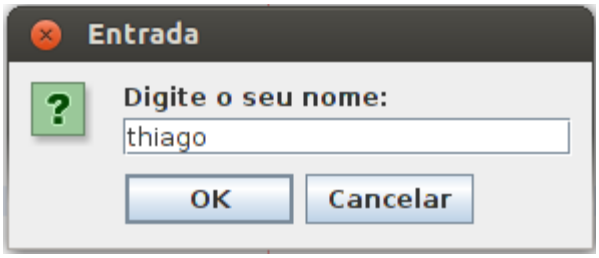
showInputDialog

O método showInputDialog envia uma caixa de diálogo para o usuário, porém permite que o mesmo digite algum valor para entrar no sistema. O método showInputDialog **retorna um objeto String como padrão.** Se desejarmos outro tipo de dado precisamos utilizar os métodos parse.

Código:

```
JOptionPane.showInputDialog(
                    null,
                    "Digite o seu nome: ");
```

Visualização do código:



obs: No exemplo estamos digitando uma String de 6 caracteres chamada “thiago”. Quando for clicado no botão de OK essa String será recebida em alguma **variável ou objeto** dentro do Java. Mas vamos imaginar que precisamos receber um número para realizar algum cálculo.
O JOptionPane não converte automaticamente para o tipo de dado numérico seja ele fracionário ou inteiro. Para realizar a conversão utilizaremos as Classes numéricas que desejamos converter, e dentro de cada classe temos um método que vai realizar essa conversão, o nome deste é “**parse**TipoDeDado”.

Exemplo de código com conversão de String para inteiro:

```
int numero = Integer.parseInt(
                        JOptionPane.showInputDialog(
                                    null,
                                    "Digite um valor numérico inteiro: "));
```

Exemplo de código com conversão de String para double:

```
double numero = Double.parseDouble(
                        JOptionPane.showInputDialog(
                                    null,
                                    "Digite um valor numérico com vírgula: "));
```

Exemplos de parses:

```
Integer.parseInt()
Double.parseDouble();
Boolean.parseBoolean();
Byte.parseByte();
Short.parseShort();
```

Obs.: Dentro dos parênteses colocamos o JOptionPane que retornará a String que será convertida

pelos métodos de “parse”.

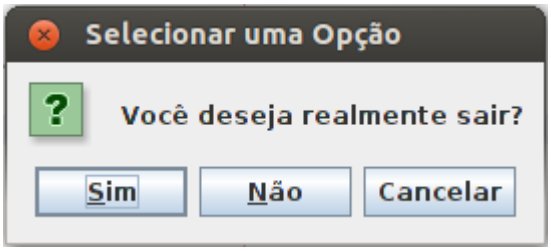
showConfirmDialog

O método showConfirmDialog é utilizado para enviar uma pergunta para o usuário. Quando o usuário clica no botão de reposta o método retorna um número “inteiro” relacionado ao botão clicado. Podemos personalizar quais botões desejamos que o usuário visualize.

Código:

```
JOptionPane.showConfirmDialog(
                null,
                "Você deseja realmente sair?");
```

Visualização do código:



obs: No caso da janela de Confirmação precisaremos saber qual botão o usuário clicou. O retorno do botão clicado é através do número do índice do botão, ou seja, o botão retorna um índice inteiro.

Por exemplo:

o botão Yes retornaria o índice 0.
o botão No retornaria o índice 1.
o botão Cancel retornaria o índice 2.

Podemos escolher entre outros tipos de opções de botões:

DEFAULT_OPTION
YES_OPTION
YES_NO_OPTION
YES_NO_CANCEL_OPTION
OK_CANCEL_OPTION
OK_OPTION

o código ficaria da seguinte forma:

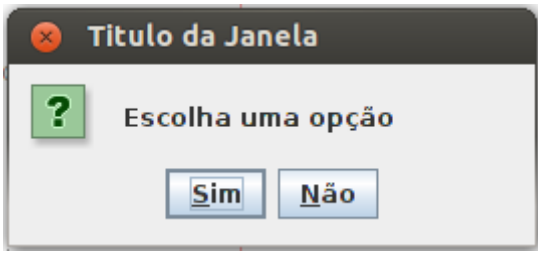
default:

```
JOptionPane.showConfirmDialog(null, args, null, optionType);
```

alterado:

```
JOptionPane.showConfirmDialog(
                null,
                "Escolha uma opção",
                "Titulo da Janela",
                JOptionPane.YES_NO_OPTION);
```

Resultado:



Podemos ainda escolher o ícone que aparecerá para o usuário. Os ícones possíveis são:

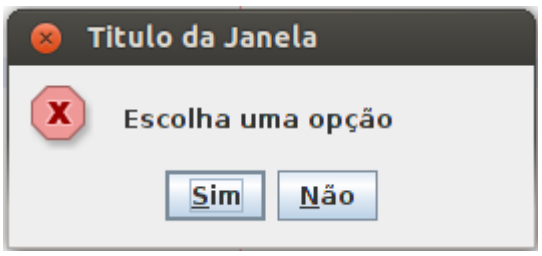
ERROR_MESSAGE
INFORMATION_MESSAGE
WARNING_MESSAGE
QUESTION_MESSAGE
PLAIN_MESSAGE

Abaixo segue a figura (seguida do código) de uma showConfirmDialog com ícone de ERROR_MESSAGE:

Código:

```
JOptionPane.showConfirmDialog(
                null,
                "Escolha uma opção",
                "Titulo da Janela",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.ERROR_MESSAGE);
```

Visualização do código:



Material desenvolvido por Thiago Cury

Referências Bibliográficas do material

http://codeerror.wordpress.com/2011/03/04/sem-medo-do-javadoc/
http://pt.wikipedia.org/wiki/Javadoc