Java Swing - Janelas, Caixas de Texto, Botões, Imagens

Janela - JFrame, JLabel, JTextField, JButton, JCheckBox, FlowLayout, GridLayout

Arquivos anexados: FrameTest.java (1,72 KB)

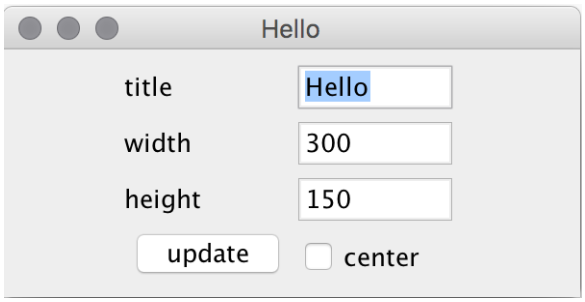
O objetivo deste exercício é tomar contacto com as classes do Swing mais elementares, para representar janelas (JFrame), painéis (JPanel), etiquetas (JLabel), campos de texto (JTextField), botões (JButton, JCheckBox), e formas de os organizar (FlowLayout, GridLayout).

1. Experimente executar a classe em anexo (FrameTest), alterando alguns dos valores utilizados no exemplo, tais como o texto utilizado nos elementos e os valores utilizados no GridLayout.

2. Com base na estrutura da classe dada, desenvolva uma outra classe cujo o aspeto da interface gráfica seja semelhante ao apresentado na figura em baixo. Deverá ser possível executar a classe configurando a janela com um texto para o título e abrindo-a dando uma dimensão inicial:

```
public static void main(String[] args) {
    MyFrame window = new MyFrame("Hello");
    window.open(300, 150);
}
```

Ao carregar no botão a janela deverá ser redimensionada de acordo com os valores nas caixas de texto, e caso a checkbox esteja selecionada, a localicação da janela deverá também ser alterada para o centro do ecrã.



Dica: A JFrame pode redimensionada utilizando o método Frame.setSize(...).

Dica: É possível obter a dimensão do ecrã da seguinte forma:

```
Dimension dimension = Toolkit.getDefaultToolkit
().getScreenSize();
```

Visualizador de imagens - BorderLayout, ImageIcon

Arquivos anexados: imagens.zip (105,959 KB)

Desenvolva uma aplicação que recebe um argumento de execução (parâmetro do main()) com o caminho para um diretório, de modo a permitir visualizar as imagens contidas nesse diretório (ver vídeo). O utilizador pode percorrer as imagens utilizando os botões. Quando o utilizador tentar ir para a frente da última imagem ou para trás da primeira deve ser mostrada uma mensagem na zona de exibição de imagens. No topo deverá ser mostrado o nome do ficheiro da imagem que está a ser mostrada. Por fim, deverá ser possível recarregar as imagens do diretório através do botão que se encontra na parte de baixo.

Para dispor os elementos gráficos como na forma apresentada podemos utilizar um BorderLayout no painel, que implica que ao adicionar um elemento deveremos indicar a sua localização no painel (este, oeste, norte, sul, centro).

```
frame.add(..., BorderLayout.EAST/WEST/NORTH/SOUTH/CENTER)
```

A zona de exibição de imagens deverá ser uma JLabel, onde é possível colocar uma imagem fazendo:

```
ImageIcon icon = new ImageIcon("ficheiro.png");
label.setIcon(icon);
```

Os ficheiros podem ser lidos de um diretório filtrando os ficheiros/diretórios nele contidos (de acordo com determinado critério) da seguinte forma:

```
String path = System.getProperty("user.dir"); // obtem o
diretorio de execucao (raiz do projeto Eclipse)

File[] files = new File(path).listFiles(new FileFilter() {
     public boolean accept(File f) {
          // se retornar verdadeiro, f será incluido
     }
});
```

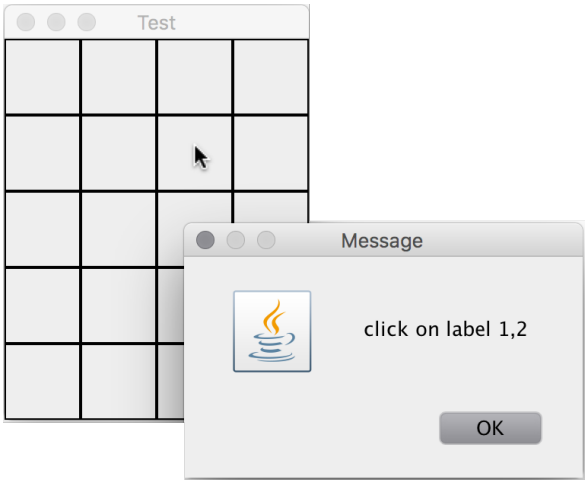
Componente reutilizável para grelhas - Border, MouseListener, JLabel estilizada

Desenvolva uma classe para representar grelhas com posições quadradas, oferecendo a possibilidade de definir o número de linhas, o número de colunas, e a

dimensão das posições em pixéis (lado de cada quadrado). Deverá ser possível executar a classe da seguinte forma:

```
public static void main(String[] args) {
    Grid grid = new Grid("Test", 5, 4, 50);
    grid.open();
}
```

de modo a ter o título “Test” e uma grelha de 5 linhas por 4 colunas, tendo cada posição 50 pixéis de largura.



1. Defina a disposição das posições da grelha. O Layout mais adequado é o GridLayout, o qual pode ser inicializado com o número exacto de linhas/colunas. Cada posição pode consistir numa JLabel configurada com uma linha de contorno (Border) com uma cor e uma espessura de linha, o que pode ser feito da seguinte forma:

```
Border border = BorderFactory.createLineBorder(Color.black,
2);
label.setBorder(border);
```

De modo a que a grelha tenha a dimensão pretendida podemos utilizar o método JPanel.setPreferredSize(Dimension) indicando que o painel tem uma dimensão pretendida, o que será levado em conta no momento em que JFrame.pack() executa.

Dica: a Frame pode ser configurada para não permitir redimensionamento manual utilizado JFrame.setResizable(false)

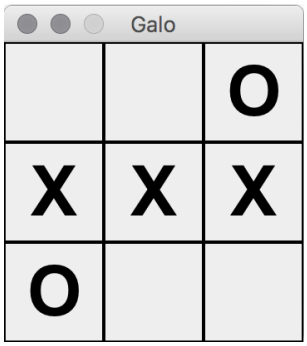
2. Defina o comportamento quando são pressionadas as posições da grelha. Isto pode ser conseguido registando uma sentinela para reagir a eventos do rato (interface MouseListener) em cada JLabel.

```
label.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        //...
    }
});
```

MouseListener é uma interface com diversas operações relacionadas com eventos do rato. Para conveniência, existe uma classe abstrata MouseAdapter que implementa a interface com todos os métodos sem comportamento, e desta forma pode ser estendida sobrepondo a operação de interesse (como no exemplo em cima, onde é definido o método associado aos cliques do rato).

3. Altere a classe desenvolvida para que a mesma possa ser estendida com vista a alterar o comportamento da aplicação quando é pressionada uma posição da grelha. Com uma solução baseada em herança, isto pode ser conseguido tendo o comportamento das sentinelas definido num método da classe que possa ser sobreposto em subclasses. Permita também que as JLabel das posições sejam estilizadas, isto é, que as subclasses possam configurá-las no momento em que as mesmas são instanciadas.

4. Desenvolva um subclasse de Grid para representar a grelha do Jogo do Galo (ver figura), incluindo as marcações de jogadas (com as letras “X” e “O”). As marcas dos jogadores deverão ser inseridas em alternância, sendo que um clique numa posição já preenchida não terá efeito.



Dica: o tipo de letra da JLabel pode ser redefinido, bem como o alinhamento horizontal:

```
label.setFont(new Font("Arial", Font.BOLD, 42));
label.setHorizontalAlignment(SwingConstants.CENTER);
```

5. Desenvolva uma subclasse de Grid para representar um tabuleiro de Xadrez, não atendendo a quaisquer regras de jogo. Porém, deverá ser possível colocar e mover peças no tabuleiro (ver animação para uma possível forma de interagir com o tabuleiro).

Dica: a cor de fundo de uma JLabel pode ser alterada utilizando o método JLabel.setBackground(Color) (por exemplo: `label.setBackground(Color.red)`). Contudo, para que a cor seja mostrada é necessário configurar a JLabel como opaca (`label.setOpaque(true)`).