Aula 21

UFRPE

Fontes, Cores, Interface Gráfica e Eventos
(Parte 3)

Richarlyson A. D'Emery

site: https://sites.google.com/site/profricodemery/mpoo

grupo: http://groups.google.com/group/**mpoo_uast**

email grupo: **mpoo_uast**@googlegroups.com

contato: rico_demery@yahoo.com.br

Sumário

- ItemListener e ListSelectionListener
- JTextField
- JCheckBox e JRadioButton
- JList

A classe Font

UFRPE

- A maior parte dos métodos de fontes pertencem a classe **`Font`**
- Método Construtor
  - **`public Font`** `(String name,` **`int`** `style,` **`int`** `size)`
    - Cria um objeto Font com características dos argumentos
- Constantes e Descrições
  - **`public final static int`** `PLAIN // classe Font`
    - Constante que representa um estilo de fonte simples
  - **`public final static int`** `BOLD // classe Font`
    - Constante que representa um estilo de fonte negrito
  - **`public final static int`** `ITALIC // classe Font`
    - Constante que representa um estilo de fonte itálico

A classe Font

UFRPE

- Outros métodos
  - `public int getStyle()`
    - Retorna um `int` que indica o estilo da fonte
  - `public int getSize()`
    - Retorna um `int` que indica o tamanho da fonte atual
  - `public int getName()`
    - Retorna o nome da fonte atual como um string
  - `public boolean isPlain()`
    - Se fonte é simples retorna `true`
  - `public boolean isBold()`
    - Se fonte é negrito retorna `true`
  - `public boolean isItalic()`
    - Se fonte é itálica retorna `true`

A Classe Font

UFRPE

- A Classe Graphic
  - `public void setFont (Font f)`
    - Configura a fonte atual coma fonte, o estilo e o tamanho especificados para Font f
- As fontes Default
  - Monospace, SansSerif e Serif
- O estilo da fonte é
  - `Font.PLAIN`, `Font.BOLD` ou `Font.ITALIC`
- O estilo da fonte pode ser combiando
  - `Font.ITALIC + Font.BOLD`
- Ex.: ver Fonts.java

Exemplo classe Font


```
import java.awt.*;
import javax.swing.*;

public class Fontes extends JFrame {
   public Fontes(){
      super( "Using fonts" );
      setSize( 420, 125 );
      setVisible(true);
   }
    public void paint(Graphics g){
      super.paint( g ); // call superclass's paint method
      // seta a fonte como Serif (Times), bold e 12pt
       g.setFont( new Font( "Serif", Font.BOLD, 12));
      g.drawString("Serif 12 point negrito.", 20, 50 );
      // seta a fonte como Monospaced(Courier), italic e 24pt
       g.setFont( new Font( "Monospaced", Font.ITALIC, 24));
      g.drawString("Monospace 24 points italic", 20, 70);
      // seta a fonte como SansSerif(Helvetica), plain e 14pt
      g.setFont( new Font( "SansSerif", Font.PLAIN, 14));
      g.drawString("SansSerif 14 points normal", 20, 90);
      // seta a fonte como Serif (times), bold/italic e 18pt
      g.setColor(Color.red);
      g.setFont(new Font( "Serif", Font.BOLD, 18));
      g.drawString(g.getFont().getName() + " " +
g.getFont().getSize() + " point bold italic.", 20, 110);
    }
```

```
// aplicacao
   public static void main(String args[]){
      Fontes application = new Fontes();
      application.setDefaultCloseOperation(
         JFrame.EXIT_ON_CLOSE );
   }
} //fim da classe Fontws
```

A classe Color

- Podemos utilizar o método `setColor` para alterar a cor de strings (drawString)
- Tais métodos pertencem a `Graphics`
  - Ex.: ver **ShowColors.java**

Exemplo ShowColor.java


```
import java.awt.*;
import javax.swing.*;

public class ShowColors extends JFrame {
     public ShowColors(){
        super( "Using colors" );
        setSize( 400, 130 );
        setVisible(true);
     }

     // desenha retângulos e Strings em diferentes cores
     public void paint( Graphics g ){
        super.paint( g ); // call superclass's paint method
        // set new drawing color using integers
        g.setColor( new Color( 255, 0, 0 ) );
        g.fillRect( 25, 25, 100, 20 );
        g.drawString( "Current RGB: " + g.getColor(), 130, 40 );
        // set new drawing color using floats
        g.setColor( new Color( 0.0f, 1.0f, 0.0f ) );
        g.fillRect( 25, 50, 100, 20 );
        g.drawString( "Current RGB: " + g.getColor(), 130, 65 );
        // set new drawing color using static Color objects
        g.setColor( Color.blue );
        g.fillRect( 25, 75, 100, 20 );
        g.drawString("Current RGB: " + g.getColor(), 130, 90 );
        // display individual RGB values
        Color color = Color.magenta;
        g.setColor( color );
        g.fillRect( 25, 100, 100, 20 );
        g.drawString("RGB values: " +
color.getRed() + ", " + color.getGreen() + ", "
+ color.getBlue(), 130, 115 );
        }

// execute application
     public static void main( String args[] ){
         ShowColors application = new
ShowColors();
         application.setDefaultCloseOperation(
JFrame.EXIT_ON_CLOSE );
     }
} // fim da classe ShowColors
```

Fonte do JTextField

UFRPE

- Após criado o objeto `JTextField` podemos alterar a fonte através do método:
  - **`setFont`**, onde:

```
setFont(new Font ("TIPO DE FONTE ", Font.ESTILO, TAMANHO));
```

Ex.:

```
field = new JTextField("Watch the font style change", 20 );
field.setFont(new Font( "Serif", Font.PLAIN, 14 ) );
```

Alterando Fonte de JTextField com JCheckBox

- Ex.: **Ver CheckBoxFontsJTextFieldcomEventos.java**
  - Neste exemplo o tratamento do evento é executado por uma instância da classe interna `CheckBoxHandler` **(linhas 59 a 83)**
  - As linhas 43 a 45 criam uma instância da classe `CheckBoxHandler` e a registram com o método `addItemListner` com o `ItemListener` para as duas JCheckBoxes, ou seja, negrito **(linha 44)** e itálico **(linha 45)** .
  - O método `itemStateChanged` **(linhas 65 a 82)** é chamado quando o usuário clica na JCheckBox *bold* ou na *italic.* O método utiliza `event.getSource()` **(linhas 67 e 78)** para determinar em qual JCheckBox ocorreu o clique.

Alterando Fonte de JTextField com JCheckBox

UFRPE

- Se foi a `JCheckBox bold`, a condição (if/else) **(linhas 66 a 69)** utiliza getStateChange de ItemEvent para determina o estado do botão
  - ItemEvent.SELECT ou ItemEvent.DESELECT
- Se o estado for selecionado
  - valBold=Font.Bold
- Caso Contrário
  - valBold=Font.Plain
- Segue o mesmo para o JCheckBox italic
- **(linha 80)** atualiza o novo estilo de fonte em JtextField

Color x JList x JFrame

- Podemos mudar o background de um JFrame através do método setBackground.
- Ex.: **Ver ColorJListJFrame.java**

Color x Font x JCheckBox x JList x JFrame

- Ex.: **Ver ColorFontJCheckBoxJListJFrame.java**

Exercícios

1) Altere CheckBoxFontsJTextFieldcomEventos.java para RadioButtonFontsJTextFieldcomEventos.java,onde:
   - Ao invés de 2 JCheckBox são utilizados 3 JRadioButton:
     - bold
     - italic
     - bold+italic.
   - A fonte do JTextField obedecerá a seleção do JRadioButton

Exercícios

UFRPE

2) Implemente
   - Uma interface que permite alterar as fontes de uma frase num JTextField, onde possui:
     - um CheckBox para negrito e outro para itálico;
     - um ListBox para alterar a fontes para Verdana, Arial ou Times New Roman.
     - um grupo de JRadioButton para escolher o tamanho da fonte (8, 10, 12, 14)
     - um botão Restore para voltar as formatações iniciais
     - um botão Sublinhado que permite o texto do JTextField ficar sublinhado.

UFRPE

FIM

Richarlyson D'Emery
rico_demery@yahoo.com.br