ECM251 – Linguagens de Programação I

Aula 10 – L1/1 e L2/1

Engenharia da Computação – 3ª série

Interfaces Gráficas
(L1/1 – L2/1)

2024

Horário

Terça-feira: 2 x 2 aulas/semana

- L1/1 (07h40min-09h20min): *Prof. Calvetti*;
- L1/2 (09h30min-11h10min): *Prof. Calvetti*;
- L2/1 (07h40min-09h20min): *Prof. Igor Silveira*;
- L2/2 (11h20min-13h00min): *Prof. Calvetti*.

Tópico

- Interface Gráfica em Java

Definição

- Uma Interface Gráfica em Java, ou GUI - *Graphical User Interface*, é um conjunto de componentes visuais que permitem que os usuários interajam com o programa de uma forma mais amigável, visual e intuitiva;

- Esses componentes visuais incluem janelas, botões, menus, caixas de texto, barras de rolagem, tabelas, entre outros;

Definição

- Em Java, é possível criar GUI usando o pacote *Swing*, que fornece uma ampla variedade de componentes e recursos para a criação de interfaces de usuário;

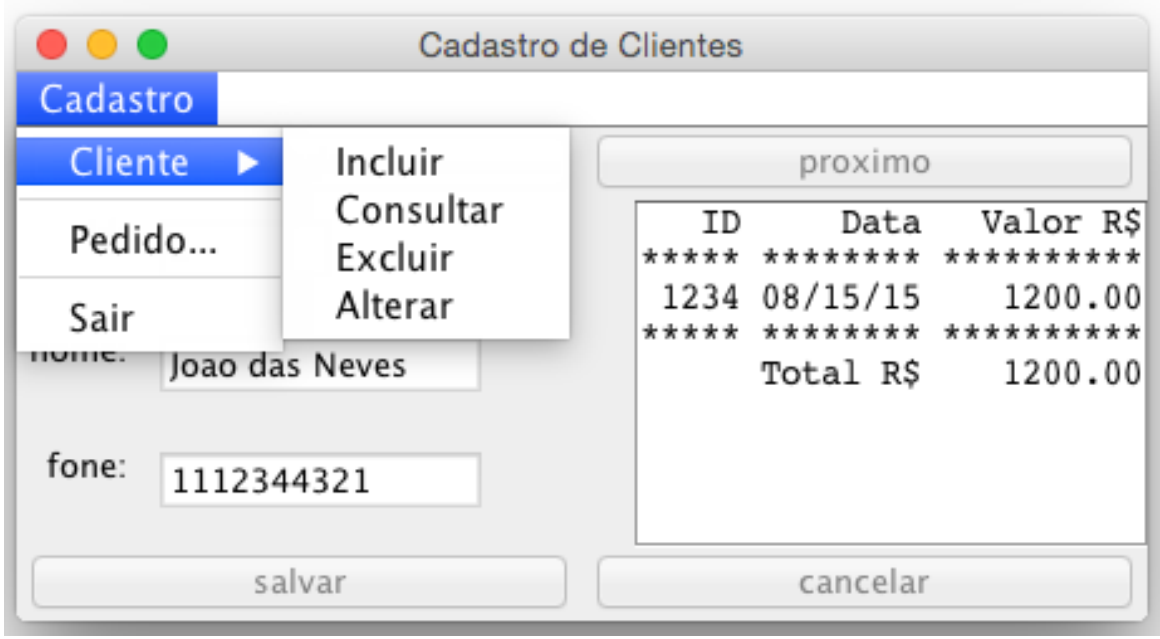
- Também é possível criar GUI usando outras bibliotecas, como a biblioteca AWT – *Abstract Window Toolkit*, ou a biblioteca *JavaFX*;

Definição

- Para criar uma GUI em Java, é necessário definir e organizar os componentes visuais em uma janela ou painel;
- Em seguida, é preciso escrever o código para manipular a interação do usuário com esses componentes, como capturar eventos de clique de botão ou preencher campos de texto;
- Isso pode ser feito por meio da implementação de classes de ouvintes de eventos e do uso de métodos para atualizar a interface gráfica com base nas ações do usuário.

Exemplo

- Uma Interface Gráfica em Java pode ser vista na imagem abaixo:



Definição

- Para criar uma GUI em Java, em geral, importam-se para o aplicativo os 3 (três) pacotes, ou *packages*, a seguir:
  - ***javax.swing.****, que tem os componentes mais novos que sempre começam com jota;
  - ***java.awt.****, que tem os componentes mais antigos, sem o jota a frente;
  - ***java.awt.event.****, que contém as interfaces de eventos.

Exemplo

- Importação de alguns pacotes da Interface Gráfica em Java:

```
1 import javax.swing.JFrame;
2 import javax.swing.JButton;
3 import javax.swing.JTextField;
4 import javax.swing.JLabel;
5 import java.awt.Container;
6 import java.awt.FlowLayout;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
```

Tópico

- A Classe *JFrame*

Classe JFrame

Definição

- A classe *JFrame* fornece uma tela com moldura, que pode ser maximizada e minimizada e que é vista pela aplicação, gerando ícone na barra de tarefas;

- Para a tela ser um *JFrame* , basta utilizar a herança e estender a *JFrame*:

```
public class TelaCalculadora extends JFrame{

}
```

Tópico

- Controle de Eventos

Definição

- O mecanismo de controle de eventos do Java funciona assim:
  - Os componentes de tela geram eventos quando o usuário faz alguma coisa;
  - Quando clica-se com o mouse, digita-se em um campo, ou aciona-se um botão, um evento é gerado e passado para a JVM – *Java Virtual Machine* instalada;

Definição

- O mecanismo de controle de eventos do Java funciona assim:
  - Pode-se reconhecer esses eventos e solicitar ao sistema que se execute uma determinada ação quando eles acontecerem, inscrevendo-se a classe que executará essa ação como ouvinte, ou *listener*, do evento;

Definição

- O mecanismo de controle de eventos do Java funciona assim:

  - Por enquanto, é importante salientar que, para poder ser ouvinte de eventos gerados por um botão, a classe precisa implementar a interface ***ActionListener*** ;

  - Como consequência disso, é necessário inserir e definir o método definido por ela, que é:

public void actionPerformed(ActionListener)

Controle de Eventos

Exemplo

- Controlando eventos e executando ações:

```
10 public class TelaCalculadora extends JFrame implements ActionListener{
11
12     public void actionPerformed(ActionEvent e){
13
14     }
15 }
```

Tópico

- Elementos Gráficos

Definição

- Para criar as variáveis de instância, que, neste caso, serão os elementos gráficos da tela, ou *widgets*, insira os códigos em Java abaixo, na sequência, para botão, texto e etiqueta:

```
12      private JButton botao; // botao
13      private JTextField texto; //campo texto de uma linha
14      private JLabel etiqueta; //etiqueta de nome do campo
```

Tópico

- Montando a Tela

Montando a Tela

Definição

- Será no construtor que, efetivamente, irá se montar a tela;
- Primeiramente, deve-se invocar o método ***super()***, passando-se como parâmetro, o texto do título da tela, através de uma *String*, que irá ficar na barra superior da janela:

```
16      public TelaCalculadora(){
17      //chamar construtor da superclasse e configurar o titulo
18          super("Calculadora");
19
20      }
```

Tópico

- Instanciação dos Elementos de Tela

Instanciação dos Elementos de Tela

Definição

- Instanciam-se os botões, campos de texto e etiquetas previamente criados, lembrando que cada construtor pode ser de uma forma diferente do outro;
- O botão, definido por ***JButton***, recebe seu texto;
- A etiqueta, definida por ***JLabel***, recebe seu texto;
- O campo texto, definido por ***JTextField***, recebe o tamanho do seu campo;
- Para outras opções de construtor, consulte a documentação da ***API*** do Java.

Instanciação dos Elementos de Tela

Exemplo

- Para instanciar os elementos da tela, ou widgets, tem-se o código abaixo:

```
19          //instanciar os widgets
20          botao = new JButton("Soma");
21          texto = new JTextField("0", 10);
22          etiqueta = new JLabel("Valor: ");
23
```

Tópico

- Gerenciador de *Layout*

Definição

- A tela de um **JFrame**, além da moldura, tem um painel, que é onde são colocados os elementos de interface;
- Todo painel tem que ter o próprio gerenciador de *layout*;
- Se não tiver o gerenciador de *layout*, todo elemento de tela colocado neste painel irá assumir o tamanho do painel e, como consequência, somente o último elemento adicionado irá aparecer;
- No caso do **JFrame**, o painel é o **Container**, que se obtém invocando o método ***getContentPane()***.

Tópico

- O gerenciador **FlowLayout**

O gerenciador FlowLayout

Definição

- O **FlowLayout** será o primeiro gerenciador de *layout* a ser visto neste material, porém, há outros gerenciadores, que serão apresentados posteriormente;
- O comportamento do **FlowLayout** segue a seguinte lógica de trabalho:
  - O **FlowLayout** organiza os elementos da tela, que vão sendo a ela adicionados, da esquerda para direita, um após o outro;
  - Se acabar o espaço horizontal da tela, então os próximos elementos são posicionados a partir do início da próxima linha, sequencialmente, assim por diante.

O gerenciador FlowLayout

Exemplo

- Para configurar o gerenciador de *layout*, tem-se o código abaixo:

```
23        //pega o container (ou painel)
24        Container caixa = getContentPane();
25        //configura o gerenciador de layout
26        caixa.setLayout(new FlowLayout());
```

Tópico

- Adicionando os elementos de tela

Adicionando os elementos de tela

Definição

- Para adicionar os elementos de tela no painel, neste caso, no *container*, utiliza-se o comando ***add(****objeto****)***;

- O gerenciador de *layout* é quem irá organizá-los:

```
27        //adiciona na tela na ordem em que quer que apareca
28        caixa.add(etiqueta);
29        caixa.add(texto);
30        caixa.add(botao);
```

Tópico

- Registrando o *listener*

Registrando o listener

Definição

- O *listener* é quem irá "ouvir" os eventos e reagir a eles;
- A tela que está sendo elaborada nesta sequência, por implementar a interface **ActionListener**, pode ser registrada como ouvinte de eventos de botão.

```
31        //registra este objeto como listener
32        botao.addActionListener(this);
```

Tópico

- Ajustes finais da tela

Ajustes finais da tela

Definição

- Antes de finalizar um tela, deve-se configurar seu tamanho, ou seja, sua largura ***x*** e sua altura ***y*** (em pixels);

- A seguir, deve-se definir o que será feito quando o usuário clicar no botão X da janela, no seu canto superior direito, para solicitar seu fechamento, que neste caso, mas não sempre, irá também encerrar o programa;

- Por fim, deve-se tornar a tela programada visível ao usuário, o que deve, sempre, ocorrer por último.

Ajustes finais da tela

Exemplo

- Para configurar o tamanho, o fechamento e a visibilidade da tela, tem-se o código abaixo:

```
33        //configura ajustes finais
34        //configura o tamanho inicial da tela
35        setSize(200,100);
36        //encerra a aplicacao quando clica o xis
37        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
38        //torna a janela visivel
39        setVisible(true);
```

Tópico

- Implementando a reação do botão

Implementando a reação do botão

Definição

- O método que responde aos cliques do mouse sobre os botões da tela é o ***actionPerformed()***;
- No exemplo a seguir, verifica-se a origem do evento, invocando o método ***getSource()***;
- Neste caso, por exemplo, irá se somar 10 ao valor que está o predefinido no campo texto;
- O método ***getText()*** irá pegar a *String* com o texto que está digitado no respectivo **JTextField**;
- O método ***setText(****String****)*** coloca o texto da *String* no respectivo **JTextField**.

Implementando a reação do botão

Exemplo

- Para implementar uma reação do botão, tem-se o código abaixo:

```
42    public void actionPerformed(ActionEvent e){
43       if(e.getSource()==botao){
44          int valor = Integer.parseInt(texto.getText());
45          valor+=10;
46          texto.setText(""+valor);
47       }
48    }
```

Tópico

- Apresentação do código completo

Apresentação do código completo

Definição

- A seguir, tem-se a apresentação do código em Java completo, implementando uma tela e as respectivas reações ao ser acionado seu botão.

Implementando a reação do botão

Exemplo

- Código completo em Java:

```
1  import javax.swing.JFrame;
2  import javax.swing.JButton;
3  import javax.swing.JTextField;
4  import javax.swing.JLabel;
5  import java.awt.Container;
6  import java.awt.FlowLayout;
7  import java.awt.event.ActionEvent;
8  import java.awt.event.ActionListener;
9
10 public class TelaCalculadora extends JFrame implements ActionListener{
11
12    private JButton botao; // botao
13    private JTextField texto; //campo texto de uma linha
14    private JLabel etiqueta; //etiqueta de nome do campo
15
16    public TelaCalculadora(){
17    //chamar construtor da superclasse e configurar o titulo
18       super("Calculadora");
19       //instanciar os widgets
20       botao = new JButton("Soma");
21       texto = new JTextField("0", 10);
22       etiqueta = new JLabel("Valor: ");
```

Implementando a reação do botão

Exemplo

- Código completo em Java:

```
23          //pega o container (ou painel)
24          Container caixa = getContentPane();
25          //configura o gerenciador de layout
26          caixa.setLayout(new FlowLayout());
27          //adiciona na tela na ordem em que quer que apareca
28          caixa.add(etiqueta);
29          caixa.add(texto);
30          caixa.add(botao);
31          //registra este objeto como listener
32          botao.addActionListener(this);
33          //configura ajustes finais
34          //configura o tamanho inicial da tela
35          setSize(200,100);
36          //encerra a aplicacao quando clica o xis
37          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
38          //torna a janela visivel
39          setVisible(true);
40      }
41
42      public void actionPerformed(ActionEvent e){
43          if(e.getSource()==botao){
44              int valor = Integer.parseInt(texto.getText());
45              valor+=10;
46              texto.setText(""+valor);
47          }
48      }
49  }
```

Implementando a reação do botão

Exemplo

- Código completo em Java:

```
23          //pega o container (ou painel)
24          Container caixa = getContentPane();
25          //configura o gerenciador de layout
26          caixa.setLayout(new FlowLayout());
27          //adiciona na tela na ordem em que quer que apareca
28          caixa.add(etiqueta);
29          caixa.add(texto);
30          caixa.add(botao);
31          //registra este objeto como listener
32          botao.addActionListener(this);
33          //configura ajustes finais
34          //configura o tamanho inicial da tela
35          setSize(200,100);
36          //encerra a aplicacao quando clica o xis
37          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
38          //torna a janela visivel
39          setVisible(true);
40      }
41
42      public void actionPerformed(ActionEvent e){
43          if(e.getSource()==botao){
44              int valor = Integer.parseInt(texto.getText());
45              valor+=10;
46              texto.setText(""+valor);
47          }
48      }
49  }
```

Implementando a reação do botão

Exemplo

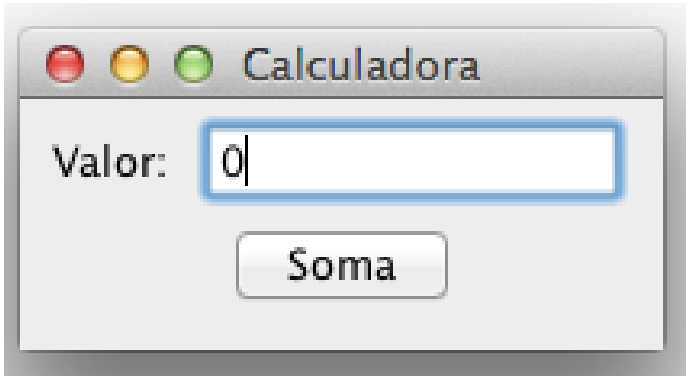
- Código completo em Java:

```
1 public class Calculadora{
2
3     public static void main(String[] args){
4         //instancia o JFrame
5         TelaCalculadora tela = new TelaCalculadora();
6     }
7 }
```

Implementando a reação do botão

Exemplo

- Resultado:



Tópico

- O gerenciador **BorderLayout**

O gerenciador BorderLayout

Definição

- O **BorderLayout** divide a tela em 5 áreas:

centro, norte(topo), sul(rodapé), leste(direita) e oeste(esquerda)

- Para adicionar objetos no **BorderLayout** deve ser invocado o mesmo método *add()*, só que, desta vez, com um parâmetro a mais, o parâmetro da posição na qual se deseja inserir o objeto:

container.add(objeto, BorderLayout.SOUTH);

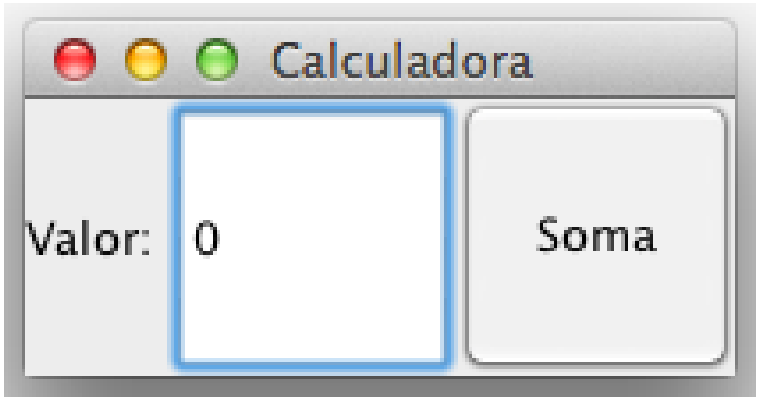
container.add(outroObjeto, BorderLayout.EAST);

O gerenciador BorderLayout

Exemplo

- Código e resultado:

```
25        //configura o gerenciador de layout
26        caixa.setLayout(new BorderLayout());
27        caixa.add(etiqueta, BorderLayout.WEST);
28        caixa.add(texto, BorderLayout.CENTER);
29        caixa.add(botao, BorderLayout.EAST);
```



Tópico

- O gerenciador **GridLayout**

O gerenciador GridLayout

Definição

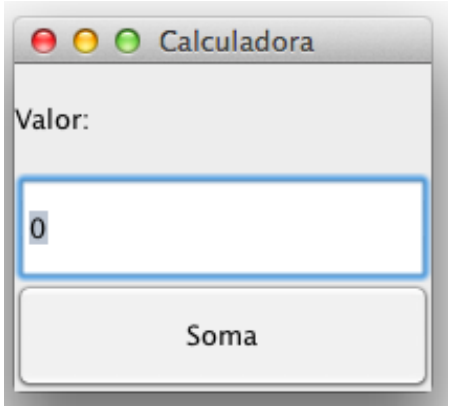
- O **GridLayout** divide a tela em uma matriz, com linhas e colunas;
- Por exemplo, ao invocar-se o construtor ***new GridLayout(3,1)*** está se criando uma matriz com 3 linhas e 1 coluna;
- Usa-se, também, o método ***add()*** para se colocar os elementos na tela;
- A cada ***add()***, o gerenciador irá adicionar o elemento de tela em uma célula da matriz, começando do topo, à esquerda, e preenchendo a linha até seu final e, na sequência, prosseguindo da mesma forma para as linhas subsequentes.

O gerenciador GridLayout

Exemplo

- Código e resultado:

```
26          caixa.setLayout(new GridLayout(3,1));
27          //adiciona na tela na ordem em que quer que apareca
28          caixa.add(etiqueta);
29          caixa.add(texto);
30          caixa.add(botao);
```



Tópico

- Combinando os *layouts*

Combinando os layouts

Definição

- Para usar mas de um gerenciador em um mesmo **JFrame** e construir *layouts* mais elaborados, deve-se fazer o seguinte:

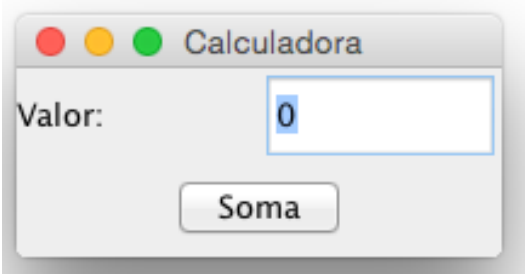
1. Cria-se um **JPane** e se atribui a ele o gerenciador de *layout* desejado;
2. Adiciona-se os elementos gráficos a este **JPane**;
3. Adiciona-se o **JPane** ao *container* do **JFrame**.

Combinando os layouts

Exemplo

- Código e resultado:

```
29        caixa.setLayout(new BorderLayout());
30        //cria os paineis secundarios
31        JPanel painelSul = new JPanel(new FlowLayout());
32        JPanel painelCentro = new JPanel(new GridLayout(1,2));
33        //adiciona os widgets nos paineis secundarios
34        painelSul.add(botao);
35        painelCentro.add(etiqueta);
36        painelCentro.add(texto);
37        //adiciona os paineis secundarios no principal
38        caixa.add(painelSul, BorderLayout.SOUTH);
39        caixa.add(painelCentro, BorderLayout.CENTER);
```



Exercício Resolvido

- Fazer uma tela para a hierarquia de classes **Ponto**, **Circulo** e **Cilindro**. A tela deve decidir qual é a forma pelos parâmetros de entrada, instanciar esta forma e mostrar seus dados por meio do ***toString()*** e um ***JOptionPane()***. Após isto, calcular a área e o perímetro, se for um Círculo, ou a área, o perímetro, o volume e a área de superfície, se for um Cilindro. A tela deve ter um botão **Limpar**, que limpa todos os campos, e um botão **Sair**, para encerrar o programa.

Exercício Resolvido

- Fazer uma tela para a hierarquia de classes **Ponto**, **Circulo** e **Cilindro** da aula anterior. A tela deve decidir qual é a forma pelos parâmetros de entrada, instanciar esta forma e mostrar seus dados por meio do ***toString()*** e um ***JOptionPane()***. Após isto, calcular a área e o perímetro, se for um Círculo, ou a área, o perímetro, o volume e a área de superfície, se for um Cilindro. A tela deve ter um botão **Limpar**, que limpa todos os campos, e um botão **Sair**, para encerrar o programa: vide arquivo anexo "**ExercicioResolvidoAula10.rar**"

Exercícios Extras

- Propostos pelo professor em aula, utilizando os conceitos abordados neste material...

Bibliografia Básica

- MILETTO, Evandro M.; BERTAGNOLLI, Silvia de Castro. Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, javascript e PHP (Tekne). Porto Alegre: Bookman, 2014. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788582601969

- WINDER, Russel; GRAHAM, Roberts. Desenvolvendo Software em Java, 3ª edição. Rio de Janeiro: LTC, 2009. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/978-85-216-1994-9

- DEITEL, Paul; DEITEL, Harvey. Java: how to program early objects. Hoboken, N. J: Pearson, c2018. 1234 p. ISBN 9780134743356.

Continua...

Bibliografia Básica (*continuação*)

- HORSTMANN, Cay S; CORNELL, Gary. Core Java. SCHAFRANSKI, Carlos (Trad.), FURMANKIEWICZ, Edson (Trad.). 8. ed. São Paulo: Pearson, 2010. v. 1. 383 p. ISBN 9788576053576.
- LIANG, Y. Daniel. Introduction to Java: programming and data structures comprehensive version. 11. ed. New York: Pearson, c2015. 1210 p. ISBN 9780134670942.
- TURINI, Rodrigo. Desbravando Java e orientação a objetos: um guia para o iniciante da linguagem. São Paulo: Casa do Código, [2017]. 222 p. (Caelum).

Bibliografia Complementar

- HORSTMANN, Cay. Conceitos de Computação com Java. Porto Alegre: Bookman, 2009. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788577804078
- MACHADO, Rodrigo P.; FRANCO, Márcia H. I.; BERTAGNOLLI, Silvia de Castro. Desenvolvimento de software III: programação de sistemas web orientada a objetos em java (Tekne). Porto Alegre: Bookman, 2016. E-book. Referência Minha Biblioteca: https://integrada.minhabiblioteca.com.br/#/books/9788582603710
- BARRY, Paul. Use a cabeça! Python. Rio de Janeiro: Alta Books, 2012. 458 p.
  ISBN 9788576087434.

Continua...

Bibliografia Complementar (continuação)

- LECHETA, Ricardo R. Web Services RESTful: aprenda a criar Web Services RESTfulem Java na nuvem do Google. São Paulo: Novatec, c2015. 431 p.
  ISBN 9788575224540.
- SILVA, Maurício Samy. JQuery: a biblioteca do programador. 3. ed. rev. e ampl. São Paulo: Novatec, 2014. 544 p.
  ISBN 9788575223871.
- SUMMERFIELD, Mark. Programação em Python 3: uma introdução completa à linguagem Phython. Rio de Janeiro: Alta Books, 2012. 506 p.
  ISBN 9788576083849.

Continua...

Bibliografia Complementar (continuação)

- YING, Bai. Practical database programming with Java. New Jersey: John Wiley & Sons, c2011. 918 p.
- ZAKAS, Nicholas C. The principles of object-oriented JavaScript. San Francisco, CA: No Starch Press, c2014. 97 p. ISBN 9781593275402.

FIM

Engenharia da Computação – 3ª série

Interfaces Gráficas
(L1/2 – L2/2)

2024

Horário

Terça-feira: 2 x 2 aulas/semana

- L1/1 (07h40min-09h20min): *Prof. Calvetti*;
- L1/2 (09h30min-11h10min): *Prof. Calvetti*;
- L2/1 (07h40min-09h20min): *Prof. Igor Silveira*;
- L2/2 (11h20min-13h00min): *Prof. Calvetti*.

Exercícios

1. Crie uma tela com um campo texto, contendo a etiqueta **Texto**, um botão **Mostrar**, um botão **Limpar** e um botão **Sair**. O botão **Mostrar**, ao ser clicado, mostra o conteúdo do campo **Texto** em um ***JOptionPane()***. O botão **Limpar** limpa o campo **Texto** e o botão **Sair** sai do programa.

Exercícios

2. Use as classes **Relogio** e **Mostrador** das aulas anteriores. Crie uma tela que apresenta o mostrador do relógio (*hora* e *minuto*) em um **JLabel** e que tenha três botões: **TicTac**, que aumenta um minuto a cada clique, **Hora**, que acerta a hora (0 a 23) e **Minuto**, que acerta o minuto (0 a 59). Use sempre o método ***mostra()*** da classe **Relogio** para atualizar o mostrador da tela. Consulte a documentação do Java para aprender a aumentar o tamanho e a cor da fonte.

Exercícios

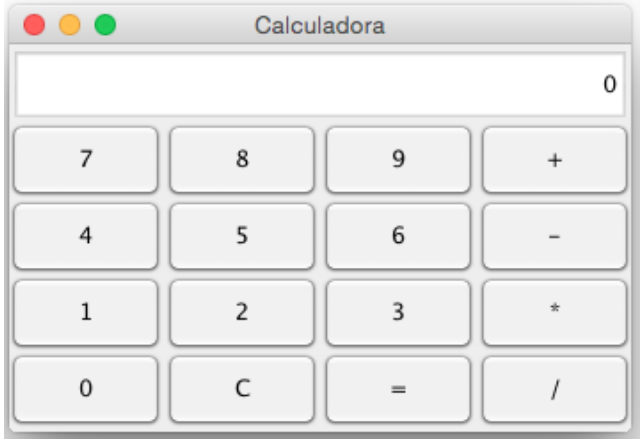
3. De modo análogo ao Exemplo Resolvido, crie uma tela para a hierarquia de classes **Empregado**, **Mensalista**, **Comissionado**, **Horista** e **Tarefeiro**.

Exercícios

4. Crie uma tela para a hierarquia de classes **PessoaFisica**, **Desempregado**, **Empregado**, **Mensalista**, **Comissionado**, **Horista** e **Tarefeiro**.

Exercícios

5. Faça uma calculadora com as 4 operações básicas, com botões de 0 a 9 para a entrada dos dígitos e um único mostrador que mostra os números digitados e o resultado das operações. Aceite apenas inteiros e faça as operações com inteiros. Use a imagem abaixo como exemplo de tela:



Bibliografia (apoio)

- LOPES, ANITA. GARCIA, GUTO. Introdução à Programação: 500 algoritmos resolvidos. Rio de Janeiro: Elsevier, 2002.
- DEITEL, P. DEITEL, H. Java: como programar. 8 Ed. São Paulo: Prentice-Hall (Pearson), 2010;
- BARNES, David J.; KÖLLING, Michael. Programação orientada a objetos com Java: uma introdução prática usando o BlueJ . 4. ed. São Paulo: Pearson Prentice Hall, 2009.

FIM