

Criando GUI com Swing

Lab 14

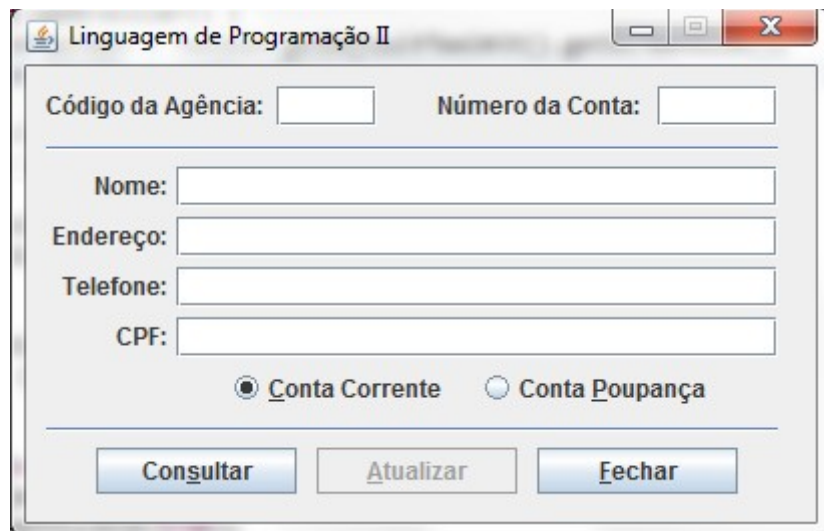
Neste laboratório iremos desenvolver uma interface gráfica com eventos que será construída ao longo deste laboratório.

Antes faça a instalação do Window Builder no eclipse para trabalhar interfaces GUI Swing. Siga o tutorial [Tutorial Window Builder](#)

Duração prevista: 40 minutos

Exercícios

Exercício 1: Desenvolver a janela abaixo.



Exercício 2: Manipulação de eventos com Java Swing

Desenvolver janela swing.

1 - Crie uma classe chamada Janela que herde as características da superclasse javax.swing.JFrame. Essa janela deve ter as seguintes características:

```
public final class Janela extends JFrame {
```

- a. O tamanho da Janela deve ser de 400 x 255 pixels
- b. O título da Janela deve ser "Linguagem de Programação II"

```
public Janela() {  
    setSize(400, 255);  
    setTitle("Linguagem de Programação II");  
}
```

- c. Implemente o método `centralizar()`, para centralizar a janela na área de trabalho do usuário. Esse método deve ser chamado dentro do construtor da classe `Janela`

```
private void centralizar() {
    Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension janela = getSize();

    if (janela.height > screen.height) {
        setSize(janela.width, screen.height);
    }
    if (janela.width > screen.width) {
        setSize(screen.width, janela.height);
    }

    setLocation((screen.width - janela.width) / 2,
                (screen.height - janela.height) / 2);
}
```

- d. Para evitar que o usuário redimensione a janela, adicione o código abaixo no construtor da classe `Janela`:

```
setResizable(false);
```

- e. Para poder ajustar os componentes livremente na Janela, você deve definir o gerenciador de Layouts do container `JFrame` como nulo. Para isso, adicione o código abaixo dentro do construtor da classe `Janela`:

```
getContentPane().setLayout(null);
```

- f. Para evitar que a aplicação continue executando após o usuário clicar no botão fechar da janela, adicione o código abaixo dentro do construtor da classe `Janela`:

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

2. Para executar a classe `Janela`, adicione um método `main()` com o seguinte código:

```
public static void main(String args[]) {
    Janela janela = new Janela();
    janela.setVisible(true);
}
```

3. Crie os seguintes atributos na classe `Janela`:

- `jIAgencia` (`javax.swing.JLabel`): Rótulo do campo agência
- `jtfAgencia` (`javax.swing.JTextField`): Campo de texto para digitar o número da agência
- `jIConta` (`javax.swing.JLabel`): Rótulo do campo conta
- `jtfConta` (`javax.swing.JTextField`): Campo de texto para digitar o número da conta
- `jSeparator01` (`javax.swing.JSeparator`): Separador que vamos utilizar para separar as informações bancárias das informações do cliente
- `jINome` (`javax.swing.JLabel`): Rótulo do campo nome
- `jtfNome` (`javax.swing.JTextField`): Campo de texto para digitar o nome do cliente
- `jIEndereco` (`javax.swing.JLabel`): Rótulo do campo endereço
- `jtfEndereco` (`javax.swing.JTextField`): Campo de texto para digitar o endereço do cliente
- `jITelefone` (`javax.swing.JLabel`): Rótulo do campo telefone
- `jtfTelefone` (`javax.swing.JTextField`): Campo de texto para digitar o telefone do cliente
- `jICpf` (`javax.swing.JLabel`): Rótulo do campo CPF
- `jtfCpf` (`javax.swing.JTextField`): Campo de texto para digitar o CPF do cliente
- `jrbCorrente` (`javax.swing.JRadioButton`): Botão de rádio para selecionar contas do tipo

“Conta Corrente”

- o. `jrbPoupanca` (`javax.swing.JRadioButton`): Botão de rádio para selecionar contas do tipo “Conta Poupança”
- p. `bgContas` (`javax.swing.ButtonGroup`): Contêiner para agrupar os componentes do tipo `JRadioButton`
- q. `jSeparator02` (`javax.swing.JSeparator`): Separador que vamos utilizar para separar as informações do cliente dos botões da janela
- r. `jbConsultar` (`javax.swing.JButton`): Botão utilizado para realizar uma consulta nas contas da agência bancária
- s. `jbAtualizar` (`javax.swing.JButton`): Botão utilizado para atualizar as informações da conta bancária
- t. `jbFechar` (`javax.swing.JButton`): Botão utilizado para fechar a aplicação

```
private JLabel jlAgencia;  
private JTextField jtfAgencia;  
private JLabel jlConta;  
private JTextField jtfConta;  
private JSeparator jSeparator01;  
private JLabel jlNome;  
private JTextField jtfNome;  
private JLabel jlEndereco;  
private JTextField jtfEndereco;  
private JLabel jlTelefone;  
private JTextField jtfTelefone;  
private JLabel jlCpf;  
private JTextField jtfCpf;  
private JRadioButton jrbCorrente;  
private JRadioButton jrbPoupanca;  
private ButtonGroup bgContas;  
private JSeparator jSeparator02;  
private JButton jbConsultar;  
private JButton jbAtualizar;  
private JButton jbFechar;
```

- 4. Crie a instância do componente `jlAgencia` e configure as seguintes opções:
 - a. O texto do componente `jlAgencia` deve ser “Código da Agência:”
 - b. O tamanho do componente `jlAgencia` deve ser de 110 x 18 pixels.
 - c. O componente `jlAgencia` deve ser posicionado no pixel 10 x 10 da janela
 - d. Adicione o componente `jlAgencia` no container da Janela

```
jlAgencia = new JLabel("Código da Agência:");  
jlAgencia.setBounds(10, 10, 110, 18);  
add(jlAgencia);
```

- 5. Crie a instância do componente `jtfAgencia` e configure as seguintes opções:
 - a. O tamanho do componente `jtfAgencia` deve ser de 50 x 20 pixels
 - b. O componente `jtfAgencia` deve ser posicionado no pixel 125 x 10 da janela
 - c. Adicione o componente `jtfAgencia` no container da Janela

```
jtfAgencia = new JTextField();  
jtfAgencia.setBounds(125, 10, 50, 20);  
add(jtfAgencia);
```

- 6. Crie a instância do componente `jlConta` e configure as seguintes opções:
 - a. O texto do componente `jlConta` deve ser “Número da Conta:”

- b. O tamanho do componente `jlConta` deve ser de 105 x 18 pixels.
 - c. O componente `jlConta` deve ser posicionado no pixel 205 x 10 da janela
 - d. Adicione o componente `jlConta` no container da Janela
7. Crie a instância do componente `jtfConta` e configure as seguintes opções:
 - a. O tamanho do componente `jtfConta` deve ser de 60 x 20 pixels
 - b. O componente `jtfConta` deve ser posicionado no pixel 315 x 10 da janela
 - c. Adicione o componente `jtfConta` no container da Janela
8. Crie a instância do componente `jSeparator01` e configure as seguintes opções:
 - a. O tamanho do componente `jSeparator01` deve ser de 365 x 10 pixels
 - b. O componente `jSeparator01` deve ser posicionado no pixel 10 x 40 da janela
 - c. Adicione o componente `jSeparator01` no container da Janela

```
jSeparator01 = new JSeparator();  
jSeparator01.setBounds(10, 40, 365, 10);  
add(jSeparator01);
```
9. Crie a instância do componente `jlNome` e configure as seguintes opções:
 - a. O texto do componente `jlNome` deve ser "Nome:"
 - b. O tamanho do componente `jlNome` deve ser de 60 x 18 pixels.
 - c. O componente `jlNome` deve ser posicionado no pixel 10 x 50 da janela
 - d. Alinhe o texto do componente à direita.

```
jlNome.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
```

 - e. Adicione o componente `jlNome` no container da Janela
10. Crie a instância do componente `jtfNome` e configure as seguintes opções:
 - a. O tamanho do componente `jtfNome` deve ser de 300 x 20 pixels
 - b. O componente `jtfNome` deve ser posicionado no pixel 75 x 50 da janela
 - c. Adicione o componente `jtfNome` no container da Janela
11. Crie a instância do componente `jlEndereco` e configure as seguintes opções:
 - a. O texto do componente `jlEndereco` deve ser "Endereço:"
 - b. O tamanho do componente `jlEndereco` deve ser de 60 x 18 pixels.
 - c. O componente `jlEndereco` deve ser posicionado no pixel 10 x 75 da janela
 - d. Alinhe o texto do componente `jlEndereco` à direita.
 - e. Adicione o componente `jlEndereco` no container da Janela
12. Crie a instância do componente `jtfEndereco` e configure as seguintes opções:
 - a. O tamanho do componente `jtfEndereco` deve ser de 300 x 20 pixels
 - b. O componente `jtfEndereco` deve ser posicionado no pixel 75 x 75 da janela
 - c. Adicione o componente `jtfEndereco` no container da Janela

13. Crie a instância do componente `jlTelefone` e configure as seguintes opções:
 - a. O texto do componente `jlTelefone` deve ser "Telefone:"
 - b. O tamanho do componente `jlTelefone` deve ser de 60 x 18 pixels.
 - c. O componente `jlTelefone` deve ser posicionado no pixel 10 x 100 da janela
 - d. Alinhe o texto do componente `jlTelefone` à direita.
 - e. Adicione o componente `jlTelefone` no container da Janela

14. Crie a instância do componente `jtfTelefone` e configure as seguintes opções:
 - a. O tamanho do componente `jtfTelefone` deve ser de 300 x 20 pixels
 - b. O componente `jtfTelefone` deve ser posicionado no pixel 75 x 100 da janela
 - c. Adicione o componente `jtfTelefone` no container da Janela

15. Crie a instância do componente `jlCpf` e configure as seguintes opções:
 - a. O texto do componente `jlCpf` deve ser "CPF:"
 - b. O tamanho do componente `jlCpf` deve ser de 60 x 18 pixels.
 - c. O componente `jlCpf` deve ser posicionado no pixel 10 x 125 da janela
 - d. Alinhe o texto do componente `jlCpf` à direita.
 - e. Adicione o componente `jlCpf` no container da Janela

16. Crie a instância do componente `jtfCpf` e configure as seguintes opções:
 - a. O tamanho do componente `jtfCpf` deve ser de 300 x 20 pixels
 - b. O componente `jtfCpf` deve ser posicionado no pixel 75 x 125 da janela
 - c. Adicione o componente `jtfCpf` no container da Janela

17. Crie a instância do componente `jrbCorrente` e configure as seguintes opções:
 - a. O tamanho do componente `jrbCorrente` deve ser de 111 x 20 pixels
 - b. O componente `jrbCorrente` deve ser posicionado no pixel 100 x 150 da janela
 - c. Configure o atalho (alt + c) para o componente `jrbCorrente`
`jrbCorrente.setMnemonic('C');`
 - d. Por padrão, o radio da Conta Corrente estará selecionado quando o usuário abrir a janela. Para isso, adicione o código abaixo:
`jrbCorrente.setSelected(true);`
 - e. Adicione o componente `jrbCorrente` no container da Janela

18. Crie a instância do componente `jrbPoupanca` e configure as seguintes opções:
 - a. O tamanho do componente `jrbPoupanca` deve ser de 118 x 20 pixels
 - b. O componente `jrbPoupanca` deve ser posicionado no pixel 225 x 150 da janela
 - c. Configure o atalho (alt + p) para o componente `jrbPoupanca`
`jrbPoupanca.setMnemonic('P');`

- d. Adicione o componente `jrbPoupanca` no container da Janela

19. Para garantir que apenas um botão radio seja selecionado pelo usuário, temos que agrupar os componentes `jrbCorrente` e `jrbPoupanca` em um container do tipo `ButtonGroup`. Crie a instância do container `bgContas` e adicione os componentes `jrbCorrente` e `jrbPoupanca` nele.

```
bgContas = new ButtonGroup();  
bgContas.add(jrbCorrente);  
bgContas.add(jrbPoupanca);
```

20. Crie a instância do componente `jSeparator02` e configure as seguintes opções:

- a. O tamanho do componente `jSeparator02` deve ser de 365 x 10 pixels
- b. O componente `jSeparator02` deve ser posicionado no pixel 10 x 180 da janela
- c. Adicione o componente `jSeparator02` no container da Janela

21. Crie a instância do componente `jbConsultar` e configure as seguintes opções:

- a. O tamanho do componente `jbConsultar` deve ser de 100 x 23 pixels
- b. O componente `jbConsultar` deve ser posicionado no pixel 35 x 190 da janela
- c. Configure o atalho (`alt + s`) para o componente `jbConsultar`
- d. Adicione o componente `jbConsultar` no container da Janela

```
jbConsultar = new JButton("Consultar");  
jbConsultar.setBounds(35, 190, 100, 23);  
jbConsultar.setMnemonic('S');  
add(jbConsultar);
```

22. Crie a instância do componente `jbAtualizar` e configure as seguintes opções:

- a. O tamanho do componente `jbAtualizar` deve ser de 100 x 23 pixels
- b. O componente `jbAtualizar` deve ser posicionado no pixel 145 x 190 da janela
- c. Configure o atalho (`alt + a`) para o componente `jbAtualizar`
- d. Por padrão, o componente `jbAtualizar` deve ficar desabilitado. Para isso, adicione o código abaixo:

```
jbAtualizar.setEnabled(false);
```

- e. Adicione o componente `jbAtualizar` no container da Janela

23. Crie a instância do componente `jbFechar` e configure as seguintes opções:

- a. O tamanho do componente `jbFechar` deve ser de 100 x 23 pixels
- b. O componente `jbFechar` deve ser posicionado no pixel 225 x 190 da janela
- c. Configure o atalho (`alt + f`) para o componente `jbFechar`
- d. Adicione o componente `jbFechar` no container da Janela

Manipulação de eventos com Java Swing.

1. Utilizando a janela implementada no exercício anterior crie os seguintes eventos.

- 1.1. Utilize o método `windowOpened()` da interface `WindowListener`, para exibir a caixa de

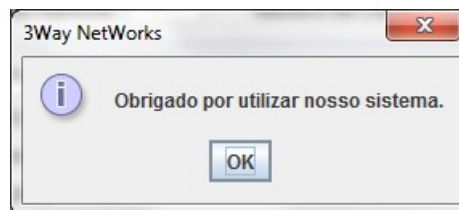
diálogo abaixo no momento em que o usuário abrir a aplicação.

- a. Dentro do construtor da classe Janela, faça uma chamada ao método `addWindowListener()`. Esse método adiciona um “ouvinte” a sua Janela.
- b. Dentro do parâmetro do método `addWindowListener()` crie uma classe do tipo `java.awt.event.WindowAdapter`
- c. Dentro da classe `WindowAdapter`, sobrescreva o método `windowOpened()` da interface `WindowListener`
- d. Dentro desse método, faça uma chamada para qualquer método da classe `Janela`.
- e. Dentro desse método, você deve criar uma caixa de diálogo (`JOptionPane`) para exibir a mensagem na tela. Veja o exemplo abaixo:

```
addWindowListener(new WindowAdapter() {  
    @Override  
    public void windowOpened(WindowEvent evt) {  
        JOptionPane.showMessageDialog(null, "Programação Java OO", "3Way NetWorks",  
        JOptionPane.INFORMATION_MESSAGE);  
    }  
});
```

- 1.2. Utilize o método `windowClosing()` da interface `WindowListener`, para exibir a caixa de diálogo abaixo no momento em que o usuário fechar a aplicação.

- a. Dentro da classe `WindowAdapter` (criada anteriormente), sobrescreva o método `windowClosing()` da interface `WindowListener`

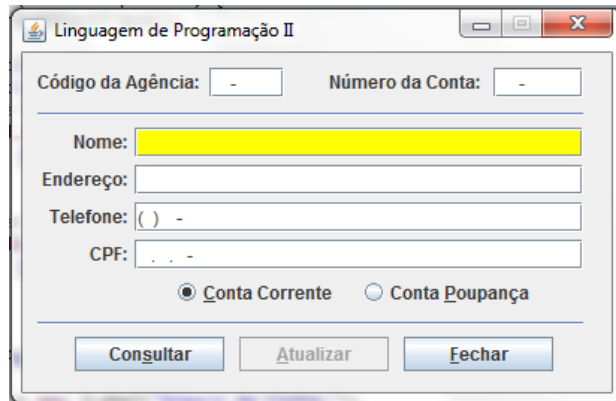


- b. Dentro desse método, faça uma chamada para qualquer método da classe `Janela` onde será exibida a caixa de diálogo acima.
- 1.3. A janela contém seis caixas de textos (`jtfAgencia`, `jtfConta`, `jtfNome`, `jtfEndereco`, `jtfTelefone` e `jtfCpf`). No momento em que o usuário ativar algum desses componentes, a cor do componente selecionado será alterado para amarelo. No momento em que o usuário sair do componente, a cor deve voltar a ser branca. Para isso, implemente a lógica abaixo em todos os componentes citados anteriormente.

- a. Dentro do construtor da classe `Janela`, após definir o tamanho e a posição de cada `JTextField`, faça uma chamada ao método `addFocusListener()` a partir do componente criado. Esse método adiciona um “ouvinte” que gerencia os focus do seu componente.
- b. Dentro do parâmetro do método `addFocusListener()` crie uma classe do tipo `java.awt.event.FocusAdapter`
- c. Dentro da classe `FocusAdapter`, sobrescreva o método `focusGained()` da interface `FocusListener`
- d. Dentro desse método, faça uma chamada para qualquer método da classe `Janela`. Veja o exemplo abaixo:
- e. Dentro desse método, você deve alterar a cor do componente selecionado pelo usuário:
- f. Essa mesma lógica deve ser utilizada para gerenciar quando o componente perde o focus. Dentro da classe `FocusAdapter` (criada anteriormente), sobrescreva o método `focusLost()` da interface `FocusListener`
- g. Dentro desse método, faça uma chamada para qualquer método da classe `Janela` onde será alterada a cor do componente.
- h. O resultado dessa implementação deve ser parecida com o exemplo abaixo:


```
jtfAgencia.addFocusListener(new FocusAdapter() {
    @Override
    public void focusGained(FocusEvent evt) {
        jtfAgencia.setBackground(Color.YELLOW);
    }

    @Override
    public void focusLost(FocusEvent evt) {
        jtfAgencia.setBackground(Color.WHITE);
    }
});
```



1.4. As caixas de textos `jtfAgencia`, `jtfConta`, `jtfTelefone` e `jtfCpf` são todas numéricas. Desse modo, adicione aos componentes uma máscara para permitir que somente caracteres numéricos sejam digitados pelo usuário. Já que estamos trabalhando com máscaras, vamos formatar o texto digitado pelo usuário.

- A classe que controla as máscaras das caixas de texto é `javax.swing.text.MaskFormatter`. A classe `MaskFormatter` contém um construtor alternativo que recebe uma `String` com a máscara formatada. Por exemplo, para criar uma máscara para o CPF devemos criar o objeto com a `String`: `"###.###.###-##"`
- A máscara não pode ser aplicada diretamente em objetos do tipo `TextField`. Mantenha a declaração das caixas de texto como `TextField`, porém as instâncias criadas para esses objetos serão do tipo `JFormattedTextField` (polimorfismo).
- O componente `jtfAgencia` deve ter a seguinte máscara: `"####-#"`
- O componente `jtfConta` deve ter a seguinte máscara: `"#####-#"`
- O componente `jtfTelefone` deve ter a seguinte máscara: `"(0xx##) ####-####"`
- O componente `jtfCpf` deve ter a seguinte máscara: `"###.###.###-##"`
- O caractere `#` serve como um coringa, e só pode ser substituído por um caractere numérico
- Para utilizar a classe `MaskFormatter` é necessário fazer o tratamento de exceção do tipo `ParseException`. Veja um exemplo abaixo:

```
try {
    jtfAgencia = new JFormattedTextField(new MaskFormatter("####-#"));
} catch (ParseException e) {
    e.printStackTrace();
}
```

1.5. Finalizando essa lista de exercício, vamos adicionar os eventos paracomponentes do tipo `JButton` ao nosso projeto. Para isso:

- Utilize o método `actionPerformed()` da interface `ActionListener`, para controlar os eventos nos botões `jbConsultar`, `jbAtualizar` e `jbFechar`

- b. Dentro do construtor da classe Janela, após definir o tamanho, a posição e o atalho de cada JButton, faça uma chamada ao método `addActionListener()` a partir do componente criado. Esse método adiciona um “ouvinte” que gerencia todos os eventos que o usuário pode provocar nos botões a partir do teclado ou do mouse
- c. Dentro do parâmetro do método `addActionListener()` crie uma classe do tipo `java.awt.event.ActionListener`
- d. Dentro da classe `ActionListener`, implemente o método `actionPerformed()`
- e. Dentro desse método, faça uma chamada para qualquer método da classe `Janela`.
- f. Adicione um evento para o componente `jbConsultar`, que exiba a caixa de diálogo abaixo caso o usuário não preencha ao menos um dos campos `jtfAgencia` e `jtfConta`. Veja o exemplo abaixo:

```
jbConsultar.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        if(jtfAgencia.getText().equals("  ") || jtfConta.getText().equals("  ")){  
            JOptionPane.showMessageDialog(null, "É necessário informar a agência e a conta",  
"3Way NetWorks", JOptionPane.INFORMATION_MESSAGE);  
        }  
    }  
});
```

- g. Adicione um evento para o componente `jbFechar`, que feche a aplicação caso o usuário clique no botão Fechar. Para fechar a aplicação, utilize o comando abaixo:

```
jbFechar.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        System.exit(0);  
    }  
});
```