

Eclipse IDE

“Dá-se importância aos antepassados quando já não temos nenhum.”

Francois Chateaubriand

O Eclipse

O Eclipse (www.eclipse.org) é uma IDE (integrated development environment). Diferente de uma RAD, onde o objetivo é desenvolver o mais rápido possível através de arrastar e soltar do mouse, onde montanhas de código são geradas em background, uma IDE te auxilia no desenvolvimento, evitando se intrometer a fazer muita mágica.

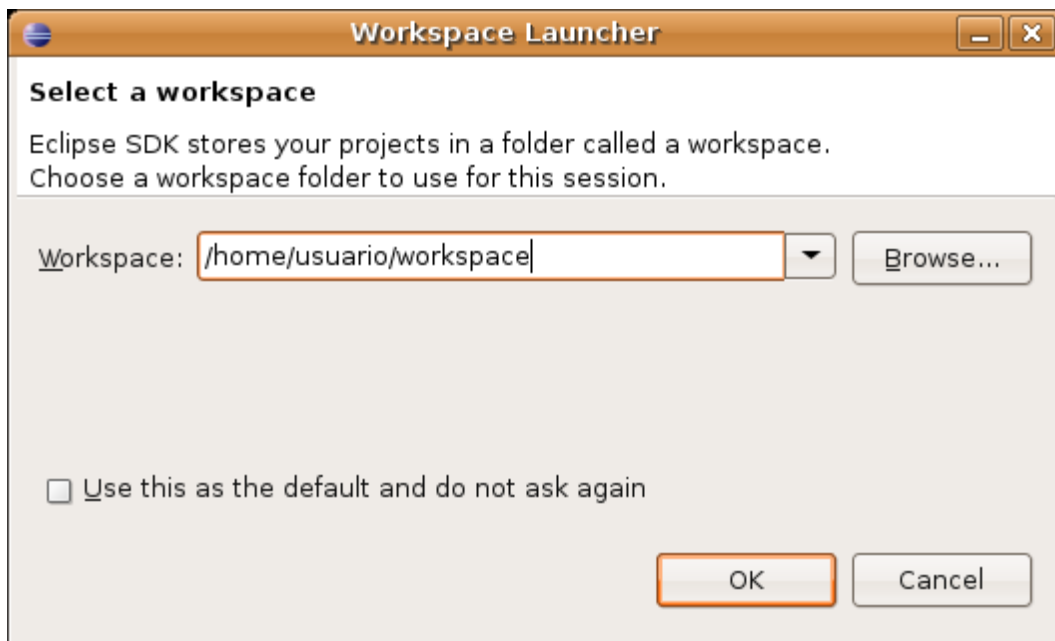
O Eclipse é a IDE líder de mercado. Formada por um consórcio liderado pela IBM, possui seu código livre. A última versão em desenvolvimento é a 3.2. Precisamos do Eclipse 3.1 ou posterior, pois a partir dessa versão é que a plataforma dá suporte ao Java 5.0. Você precisa ter apenas a Java RE instalada.

Veremos aqui os principais recursos do Eclipse. Você irá perceber que ele evita ao máximo te atrapalhar, e apenas gera trechos de códigos óbvios, sempre ao seu comando.

Existem também centenas de plugins gratuitos para gerar diagramas UML, suporte a servidores de aplicação, visualizadores de banco de dados e muitos outros.

Uma outra IDE open source famosa é o Netbeans, da Sun. (www.netbeans.org).

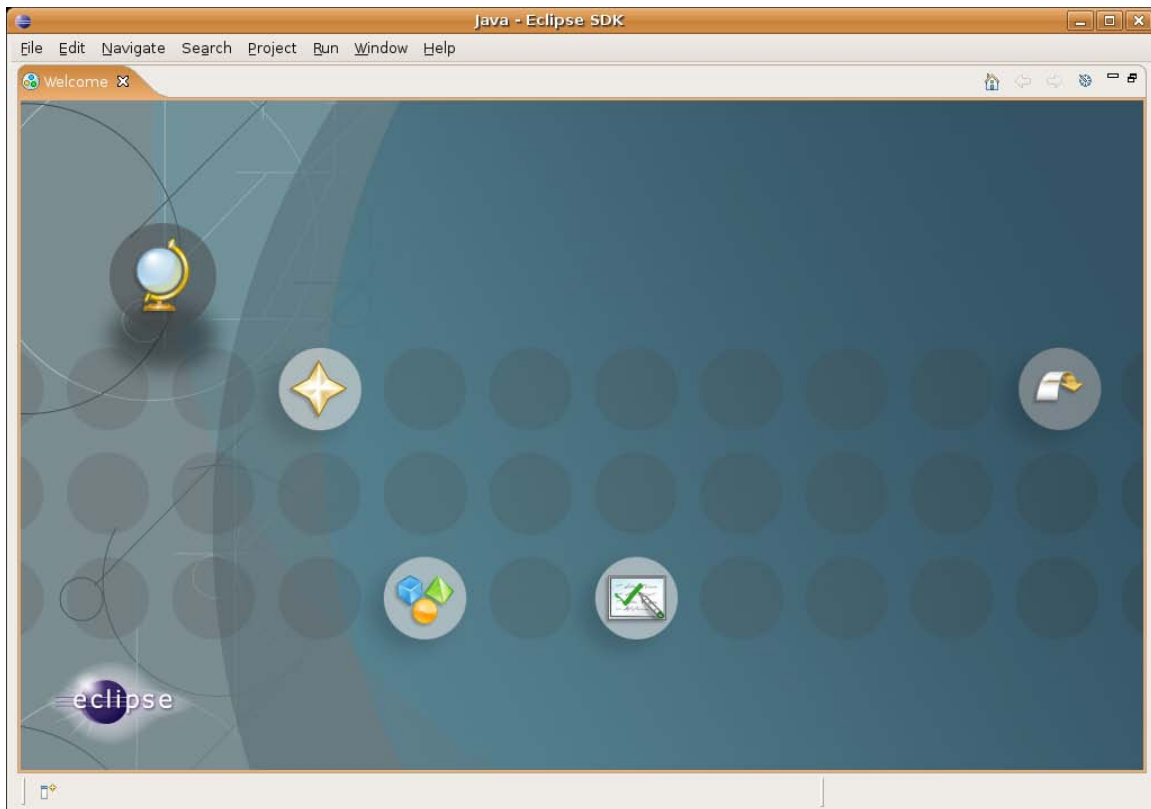
A primeira pergunta que ele te faz é que workspace você vai usar. Workspace define o diretório em que as suas configurações pessoais e seus projetos serão gravados.



Você pode deixar o diretório que ele já definiu.

Logo em seguida uma tela de Welcome será aberta, onde você tem diversos links para tutoriais e ajuda.

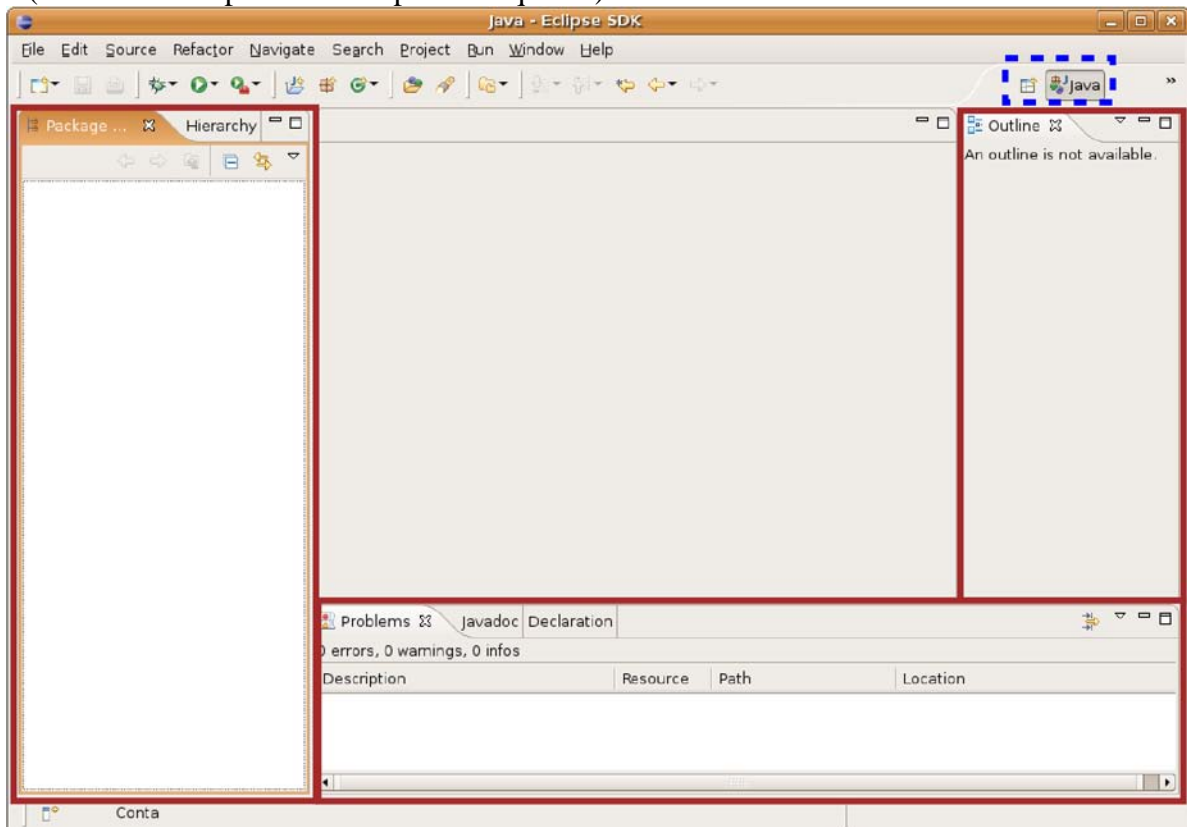
Clique em Workbench. A tela de Welcome do Eclipse 3.2 (que está na figura abaixo) é um pouco diferente da do 3.1.



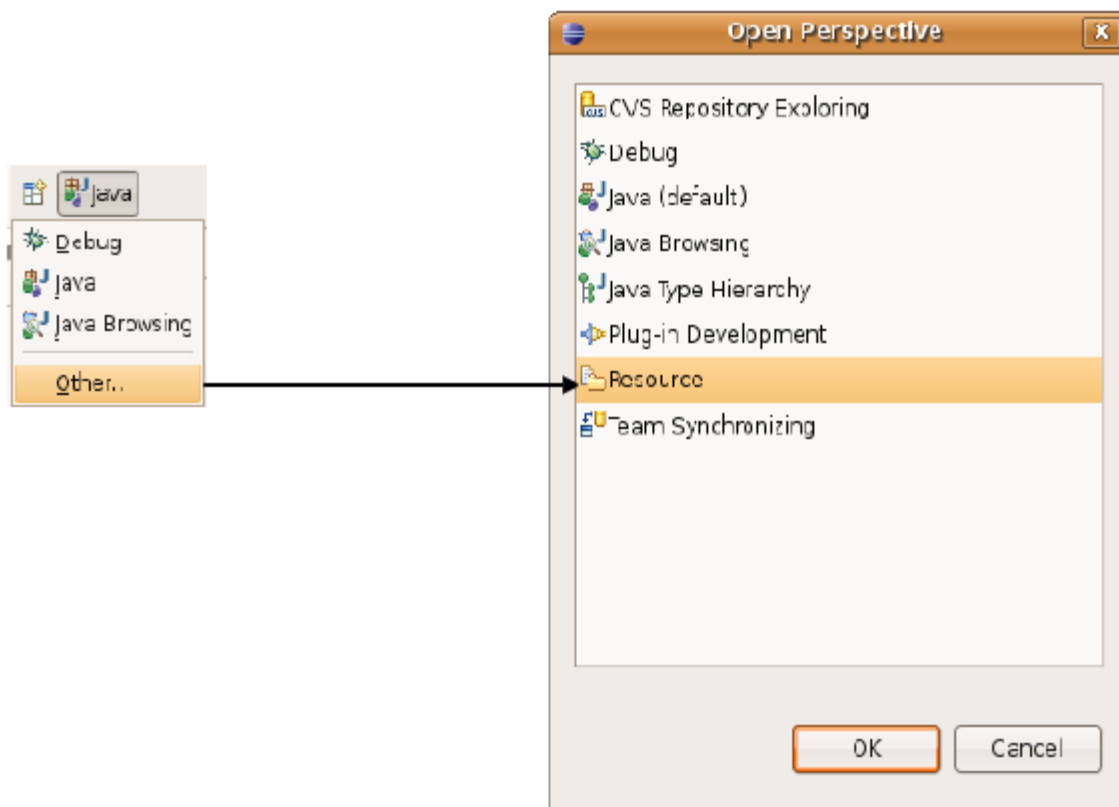
Feche a tela de Welcome e você verá a tela abaixo. Nesta tela, destacamos as Views (em linha contínua) e as Perspectives (em linha pontilhada) do Eclipse.

Mude para a perspectiva Resource **clcando no ícone ao lado da perspectiva Java** selecionando Other e depois **Resource**. Neste momento, trabalharemos com esta perspectiva antes da de Java, pois ela possui um conjunto de Views mais simples.

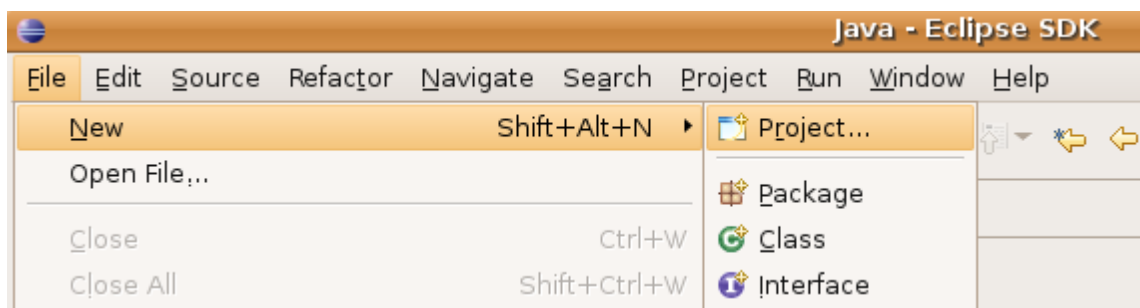
A View Navigator mostra a estrutura de diretório assim como esta no sistema de arquivos. A View Outline mostra um resumo das classes, interfaces e enumerações declaradas no arquivo java atualmente editado (serve também para outros tipos de arquivos).

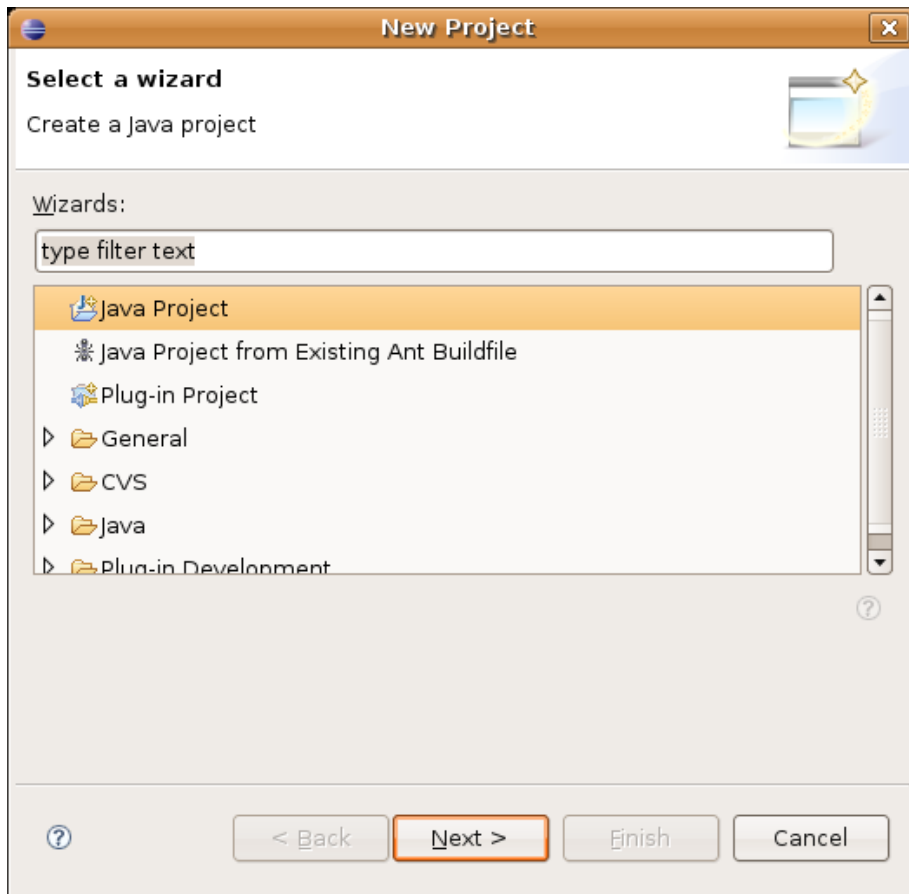


Mude para a perspectiva Resource clicando no ícone ao lado da perspectiva Java selecionando Other e depois Resource. Neste momento, trabalharemos com esta perspectiva antes da de Java, pois ela possui um conjunto de Views mais simples.

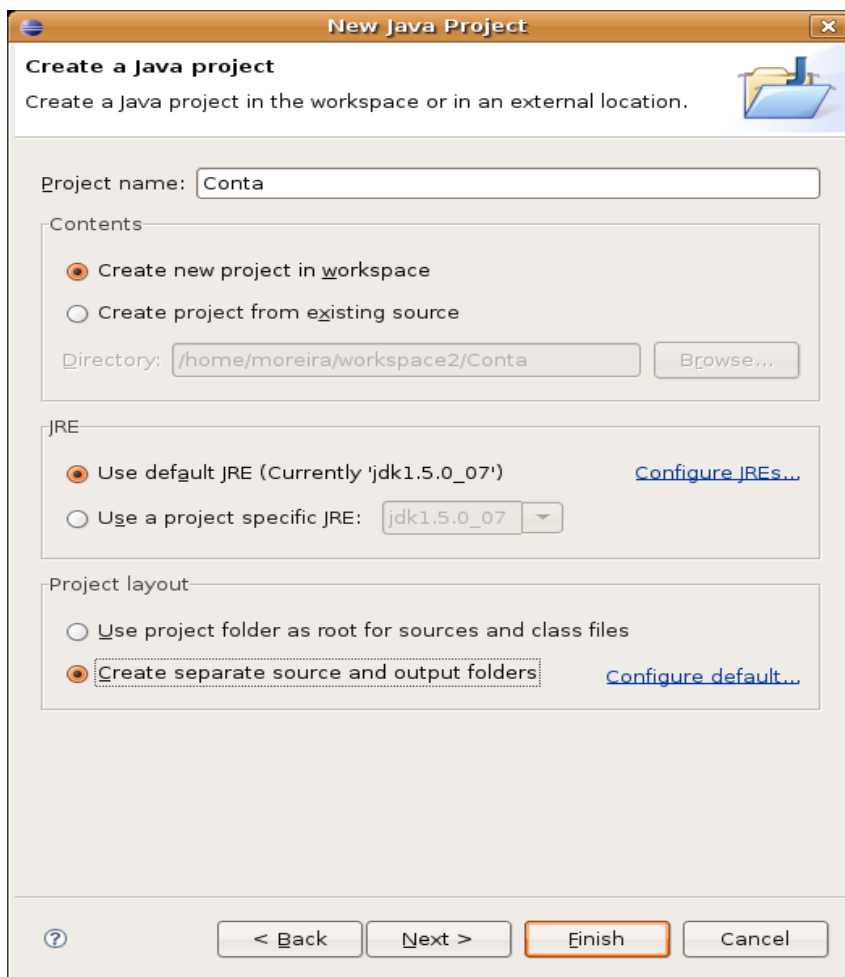


Vá em File > New >Project. Selecciona Java Project e clique em Next

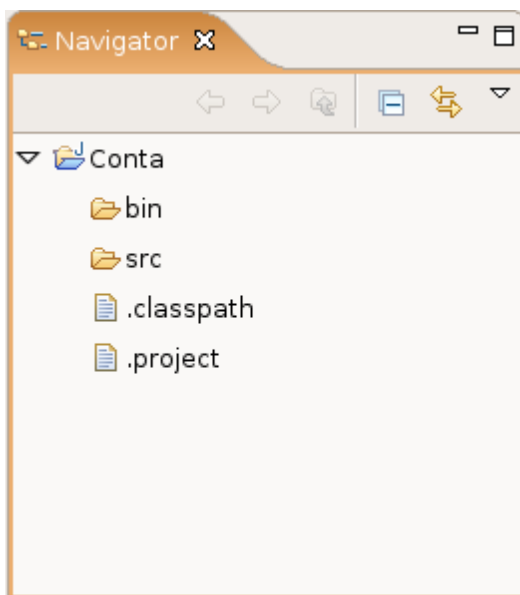




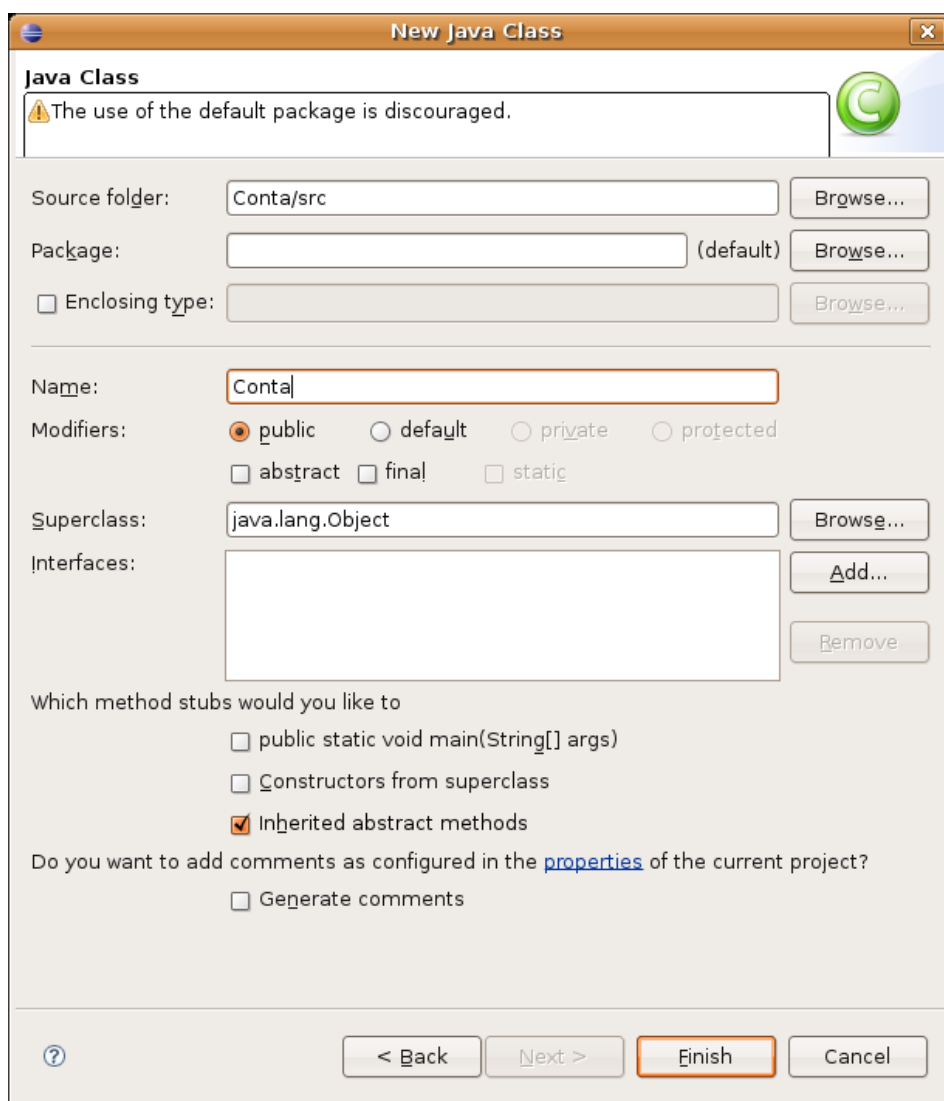
Nesta tela, configure seu projeto como na tela abaixo:



Isto e, marque “create separate source and output folders”, desta maneira seus arquivos java e arquivos class estarão em diretórios diferentes, para você trabalhar de uma maneira mais organizada. Clique em Finish. O Eclipse pedira para trocar a perspectiva para Java; escolha “No” para permanecer em Resource. Agora, na View Navigator, você verá o novo projeto e suas pastas e arquivos:



Vamos iniciar nosso projeto criando a classe Janela. Para isso, vá em File >New >Other>Class. Clique em Next e crie a classe seguindo a tela abaixo:



Clique em Finish. O Eclipse possui diversos wizards, mas usaremos o mínimo deles.

Código para criação de uma janela.

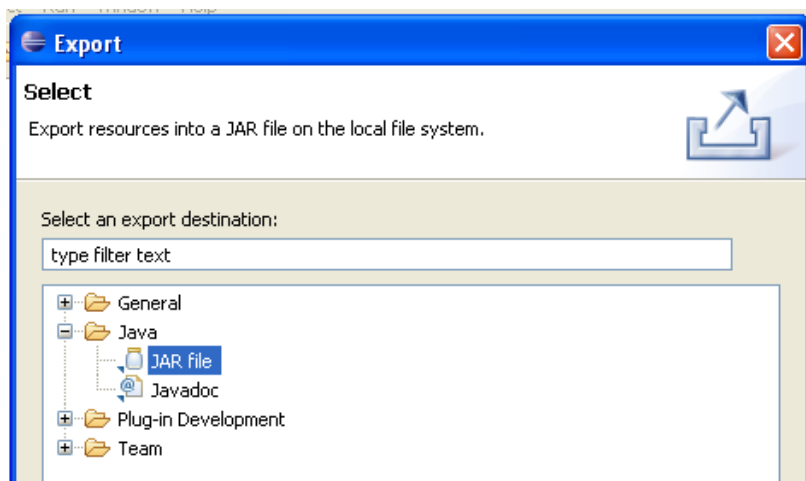
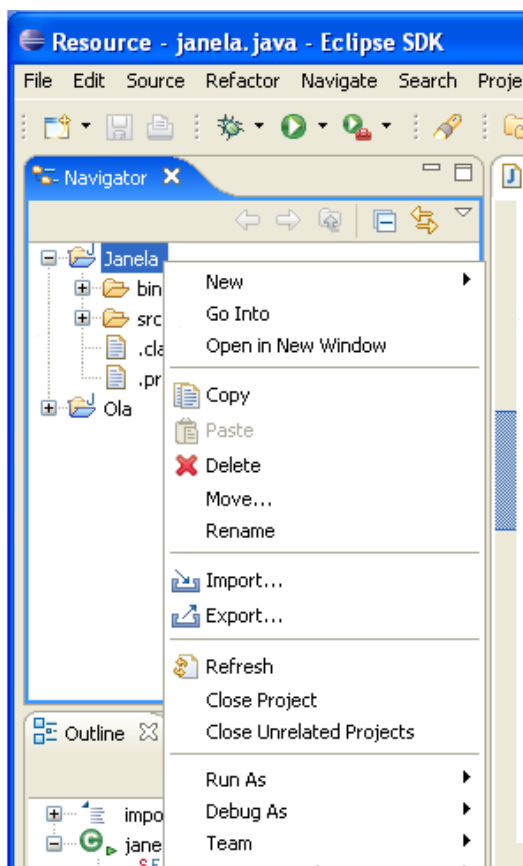
```
import javax.swing.*;
public class janela extends JFrame{
    public janela(){
        super("Primeira Janela");
        setSize(500,250);
        setVisible(true);}
    public static void main (String args[])
    {
        janela app = new janela();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Vá ao menu File >Save para gravar. Control + S tem o mesmo efeito.

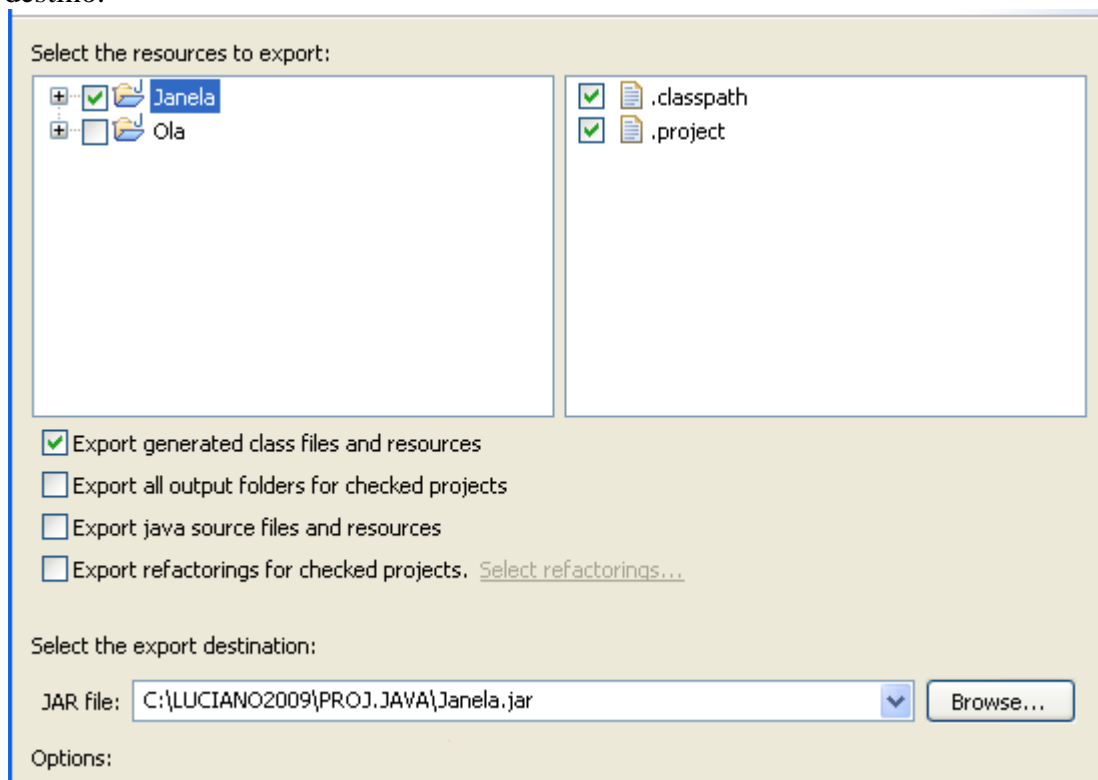
Vamos rodar o método main dessa nossa classe. No Eclipse, clique com o botão direito no arquivo Principal.java e vá em Run as... Java Application.

Criando um jar

- 1 - Clique com o botão direito do mouse no projeto que você queira criar um jar.
- 2- clique em export, surgira a janela export, clique em Java, escolha a opção Java.file



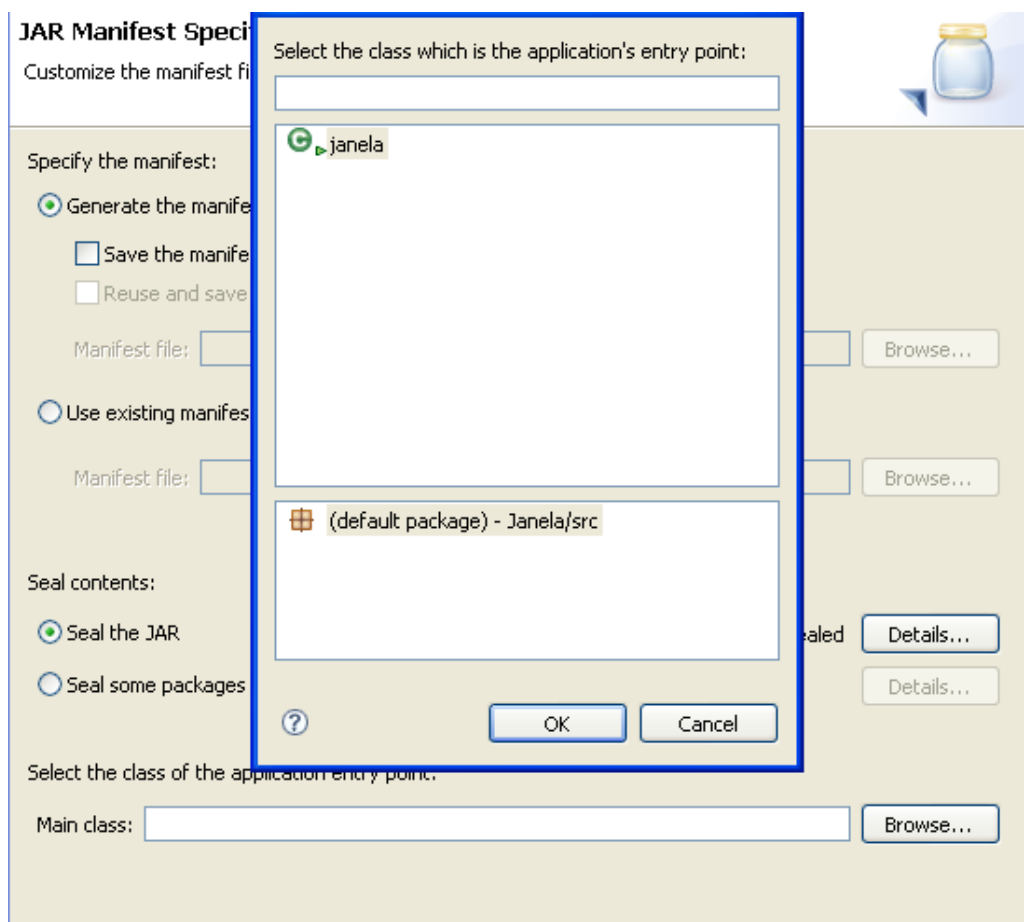
3- clique em next, na próxima janela escolha o projeto a ser tornar um jar, e escolha um diretório de destino.



4 - clique duas vezes em next .

5 – escolha a opção seal the jar.

6 – próxima janela vc tem escolher um arquivo main, clicando no botão Browse e selecionado janela.

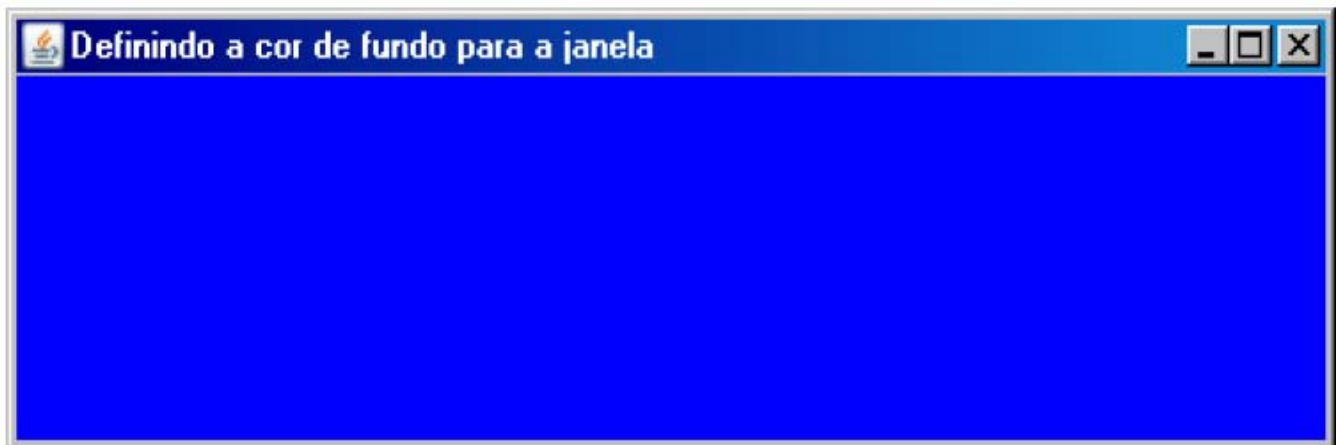


Clique em OK e depois em finish

Trocando a cor do fundo da janela

Embora as propriedades mais importantes da janela do aplicativo já tenham sido vista a possibilidade de alterar a cor de fundo da janela só pôde ser apresentada agora. O aplicativo seguinte exibe uma janela com a cor azul definida como cor de fundo. Observado que a cor de fundo para painel não afetará os demais controles da janela. Veja o aplicativo abaixo:

```
import javax.swing.*;
import java.awt.*;
public class CorDeFundo extends JFrame{
public CorDeFundo(){
super("Definindo a cor de fundo para a janela");
Container tela = getContentPane();
tela.setBackground(Color.blue);
setSize(500, 100);
setVisible(true);
}
public static void main(String args[]){
CorDeFundo app = new CorDeFundo();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```



Você pode usar: red, white, green, silver

Adicionando componentes JLabel ou rótulos na sua janela

Freqüentemente chamado de rótulo, esse componente raramente tem seu conteúdo alterado e, quando usado corretamente, possibilita manipulações bem interessantes, como veremos a seguir.

Esse aplicativo que adiciona JLabel, posiciona na janela, altera a cor e altera a fonte desse componente:

```
import javax.swing.*;
import java.awt.*;
public class ExemploLabel extends JFrame{
JLabel rotulo1,rotulo2,rotulo3,rotulo4;
public ExemploLabel(){
super("Exemplo com Label");
Container tela = getContentPane();
setLayout(null);
rotulo1 = new JLabel ("Nome");
rotulo2 = new JLabel ("Idade");
rotulo3 = new JLabel ("Telefone");
rotulo4 = new JLabel ("Celular");
```

```

rotulo1.setBounds(50,20,80,20);
rotulo2.setBounds(50,60,80,20);
rotulo3.setBounds(50,100,80,20);
rotulo4.setBounds(50,140,80,20);

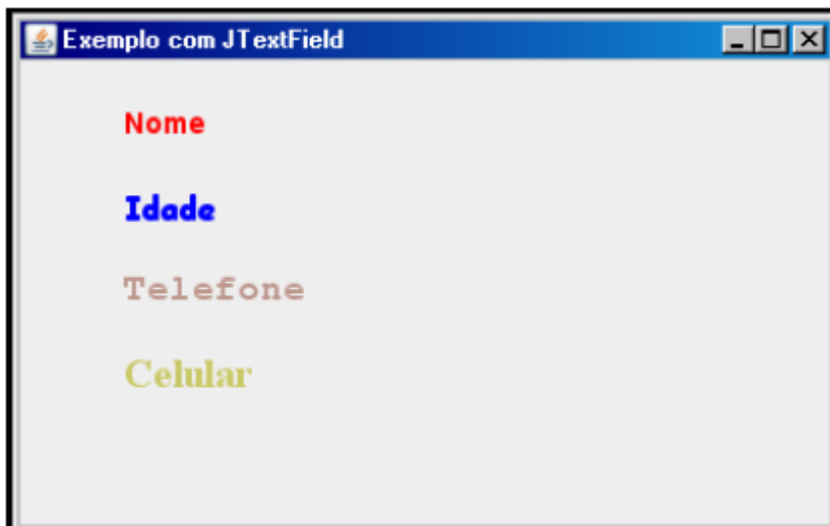
rotulo1.setForeground(Color.red);
rotulo2.setForeground(Color.blue);
rotulo3.setForeground(new Color(190,152,142));
rotulo4.setForeground(new Color(201,200,100));

rotulo1.setFont(new Font("Arial",Font.BOLD,14));
rotulo2.setFont(new Font("Comic Sans MS",Font.BOLD,16));
rotulo3.setFont(new Font("Courier New",Font.BOLD,18));
rotulo4.setFont(new Font("Times New Roman",Font.BOLD,20));

tela.add(rotulo1);
tela.add(rotulo2);
tela.add(rotulo3);
tela.add(rotulo4);

setSize(400, 250);
setVisible(true);
setLocationRelativeTo(null);
}
public static void main(String args[]){
ExemploLabel app = new ExemploLabel();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}

```



Para entender melhor como funciona o programa destacamos alguns trechos do programa, sempre lembrando que temos que no começo do programa importar esses pacotes que adicionam conteúdos a janela:

```

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

```

Apos fazermos as importações vamos criar os objetos do tipo JLabel:
 JLabel rotulo1,rotulo2,rotulo3,rotulo4;

Em seguida atribuir o conteúdo ao JLabel:

```
rotulo1 = new JLabel ("Nome");  
rotulo2 = new JLabel ("Idade");  
rotulo3 = new JLabel ("Telefone");  
rotulo4 = new JLabel ("Celular");
```

Próximo passo definir o largura e altura do JLabel e a coluna e a linha que ele irá ocupar na janela.

```
rotulo1.setBounds(50,20,80,20);  
rotulo2.setBounds(50,60,80,20);  
rotulo3.setBounds(50,100,80,20);  
rotulo4.setBounds(50,140,80,20);
```

50 – Coluna, 20 – linha, 80 – largura, 20 – comprimento

Próximo passo definir a cor da letra dos componentes JLabel, lembrando que se esses objetos não forem adicionados ao aplicativo, a cor default é preto.

```
rotulo1.setForeground(Color.red);  
rotulo2.setForeground(Color.blue);
```

Esses dois trechos abaixo especifica como criar cores personalizadas para o componente JLabel.

```
rotulo3.setForeground(new Color(190,152,142));  
rotulo4.setForeground(new Color(201,200,100));
```

E ainda podemos definir a fonte, o estilo e o tamanho da letra do componente JLabel.

```
rotulo1.setFont(new Font("Arial",Font.BOLD,14));  
rotulo2.setFont(new Font("Comic Sans MS",Font.BOLD,16));  
rotulo3.setFont(new Font("Courier New",Font.BOLD,18));  
rotulo4.setFont(new Font("Times New Roman",Font.BOLD,20));
```

E finalmente o método tela que representa a janela deverá ser chamado, onde vai exibir o rótulo o conteúdo do JLabel na janela.

```
tela.add(rotulo1);  
tela.add(rotulo2);  
tela.add(rotulo3);  
tela.add(rotulo4);
```

Adicionando imagem ao componente JLabel

É Possível exibir imagem em rótulos como instâncias da classe JLabel. Tais rótulos podem conter apenas imagens ou imagens e texto. O aplicativo seguinte mostra apenas uma imagem adicionada com um JLabel na janela.

Exemplo:

```
import javax.swing.*;  
import java.awt.*;  
public class LabellImagem extends JFrame{  
    JLabel imagem;  
    public LabellImagem(){  
        super("Uso da classe JLabel com Imagem");
```

```

Container tela = getContentPane();
Imagem icone = new Imagem("sapo.jpeg");
imagem = new JLabel(icone);
tela.add(imagem);
setSize(500, 460);
setVisible(true);
}
public static void main(String args[]){
LabelImagem app = new LabelImagem();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

```



Adicionando componentes JTextField ou caixa de texto na sua janela

A classe JTextField possibilita a criação de caixas de texto de uma única linha. Os usos mais frequentes desse controle são para receber e validar dados informados pelo usuário do aplicativo. Essa classe herda a maioria de seus atributos, eventose métodos da classe JTextComponent.

```

import javax.swing.*;
import java.awt.*;
public class ExemploJTextField extends JFrame{
JLabel rotulo1,rotulo2,rotulo3,rotulo4;
JTextField texto1,texto2,texto3,texto4;
public ExemploJTextField (){
super("Exemplo com JTextField");
Container tela = getContentPane();
setLayout(null);
rotulo1 = new JLabel ("Nome");
rotulo2 = new JLabel ("Idade");
rotulo3 = new JLabel ("Telefone");
rotulo4 = new JLabel ("Celular");
texto1 = new JTextField(50);

```

```

texto2 = new JTextField(3);
texto3 = new JTextField(10);
texto4 = new JTextField(10);
rotulo1.setBounds(50,20,80,20);
rotulo2.setBounds(50,60,80,20);
rotulo3.setBounds(50,100,80,20);
rotulo4.setBounds(50,140,80,20);
texto1.setBounds(110,20,200,20);
texto2.setBounds(110,60,20,20);
texto3.setBounds(110,100,80,20);
texto4.setBounds(110,140,80,20);
tela.add(rotulo1);
tela.add(rotulo2);
tela.add(rotulo3);
tela.add(rotulo4);
tela.add(texto1);
tela.add(texto2);
tela.add(texto3);
tela.add(texto4);
setSize(400, 250);
setVisible(true);
setLocationRelativeTo(null);
}
public static void main(String args[]){
ExemploJTextField app = new ExemploJTextField();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}

```

Declarar os objetos: TextField texto1,texto2,texto3,texto4;
Estipular a quantidade de caracteres para as caixas de texto:

```

texto1 = new JTextField(50); texto2 = new JTextField(3);
texto3 = new JTextField(10); texto4 = new JTextField(10);

```

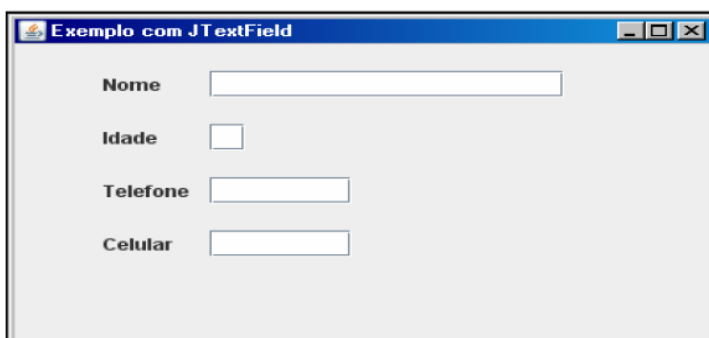
Especificar posicionamento das caixas:

```

texto1.setBounds(110,20,200,20);
texto2.setBounds(110,60,20,20);
texto3.setBounds(110,100,80,20);
texto4.setBounds(110,140,80,20);

```

E chamar o método tela para exibir as caixas na janela:
tela.add(texto1); tela.add(texto2); tela.add(texto3); tela.add(texto4);



Adicionando componentes JButton ou botões a sua janela

A classe JButton herda de AbstractButton, uma classe que herda de JComponent e define o comportamento básico para os botões e itens de menu. Como ocorre com instâncias da classe JLabel, objetos da classe JButton podem conter texto, texto e imagem ou apenas imagens. Seguindo os mesmos métodos dos outros aplicativos de como adicionar componentes na janela esse também segue o mesmo modelo.

Botão somente com texto:

```
import javax.swing.*;
import java.awt.*;
public class ExemploBotao extends JFrame{
    JButton botão;
    public ExemploBotao(){
        super("Exemplo com JButton");
        Container tela = getContentPane();
        setLayout(null);
        botao = new JButton ("Procurar");
        botao.setBounds(50,20,100,20);
        tela.add(botao);
        setSize(400, 250);
        setVisible(true);
    }
    public static void main(String args[]){
        ExemploBotao app = new ExemploBotao();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



Adicionando vários botões na janela

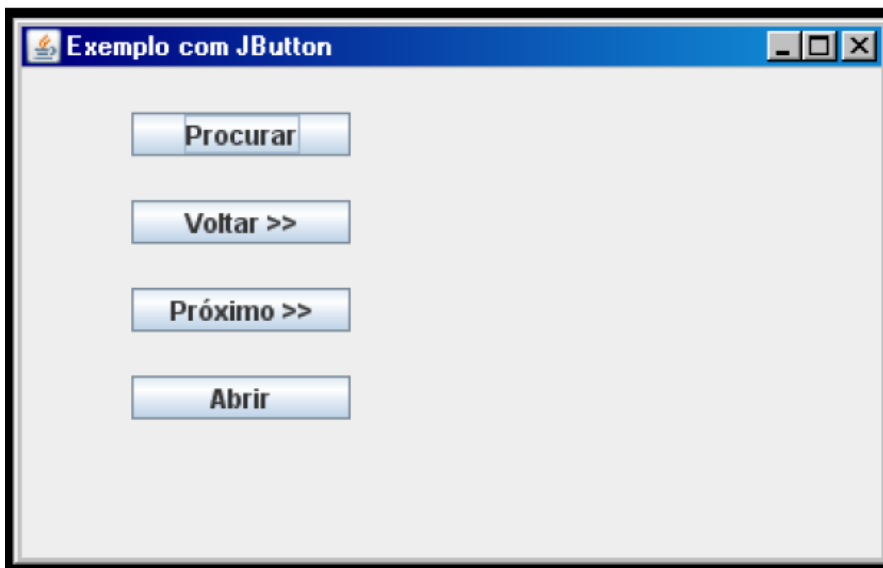
Veja o exemplo:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ExemploBotao extends JFrame{
    JButton botao1,botao2,botao3,botao4;
    public ExemploBotao(){
        super("Exemplo com JButton");
        Container tela = getContentPane();
        setLayout(null);
        botao1 = new JButton ("Procurar");
        botao2 = new JButton ("Voltar >>");
        botao3 = new JButton ("Próximo >>");
        botao4 = new JButton ("Abrir");
        botao1.setBounds(50,20,100,20);
        botao2.setBounds(50,60,100,20);
```

```

botao3.setBounds(50,100,100,20);
botao4.setBounds(50,140,100,20);
tela.add(botao1);
tela.add(botao2);
tela.add(botao3);
tela.add(botao4);
setSize(400, 250);
setVisible(true);
}
public static void main(String args[]){
ExemploBotao app = new ExemploBotao();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}

```



Fazendo um JButton executar uma ação - Clicando no botão para fechar uma janela

Exemplo:

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class ExemploBotaoSair extends JFrame{
JButton botaosair;
public ExemploBotaoSair(){
super("Exemplo com JButton");
Container tela = getContentPane();
setLayout(null);
botaosair = new JButton ("Sair");
botaosair.setBounds(100,50,100,20);
botaosair.addActionListener(
new ActionListener(){
public void actionPerformed(ActionEvent e){
System.exit(0);
}
}
);
tela.add(botaosair);
setSize(300, 150);
setVisible(true);
}
}

```

```

}
public static void main(String args[]){
ExemploBotaoSair app = new ExemploBotaoSair();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}

```



Clicando no botão somar e será mostrada a soma dos números

Clicando no botão somar e será mostrada a soma dos números Já nesse aplicativo alguns métodos novos foram aplicados, esse pede um pouquinho de nossa atenção, alguns componentes foram colocados ao lado para não ficar muito extenso e ser mais fácil de compreender.

Veja o exemplo:

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Soma extends JFrame{
JLabel rotulo1, rotulo2,exibir;
JTextField texto1,texto2;
JButton somar;
public Soma(){
super("Exemplo de soma");
Container tela = getContentPane();
setLayout(null);
rotulo1 = new JLabel("1º Número: ");
rotulo2 = new JLabel("2º Número: ");
texto1 = new JTextField(5);
texto2 = new JTextField(5);
exibir = new JLabel("");
somar = new JButton("Somar");
rotulo1.setBounds(50,20,100,20); rotulo2.setBounds(50,60,100,20);
texto1.setBounds(120,20,200,20); texto2.setBounds(120,60,200,20);
exibir.setBounds(50,120,200,20); somar.setBounds(150,100,80,20);
somar.addActionListener(
new ActionListener(){
public void actionPerformed(ActionEvent e){
int numero1,numero2,soma;
soma=0;
numero1 = Integer.parseInt(texto1.getText());
numero2 = Integer.parseInt(texto2.getText());

```

```
soma = numero1 + numero2;
exibir.setVisible(true);
exibir.setText("A soma é: "+soma);
}
}
);
exibir.setVisible(false);
tela.add(rotulo1); tela.add(rotulo2);
tela.add(texto1); tela.add(texto2);
tela.add(exibir); tela.add(somar);
setSize(400, 250);
setVisible(true);
}
public static void main(String args[]){
Soma app = new Soma();
app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```

