Programação Gráfica em Java.

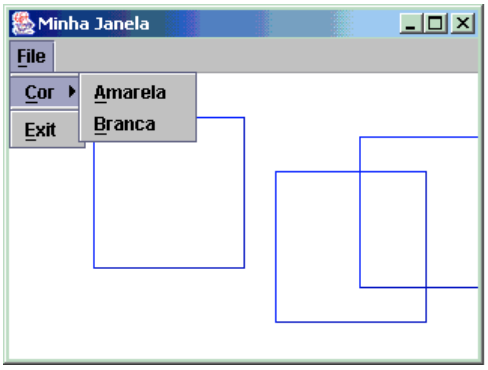
Objetivo:

Ao final da aula o aluno deverá ser capaz de desenvolver a aplicação gráfica ilustrada ao lado.

A interface gráfica será simples, composta por elementos de menus. Deverá também permitir que o usuário utilize o mouse para interagir com a aplicação.

Essa interatividade consiste em:

- Desenhar um retângulo na janela na posição definida pelo clique do mouse.
- O Menu deverá permitir a troca da cor de fundo da janela e encerrar a aplicação.



Sumário.

1. Janela

Como elaborar uma janela em Java ?

A definição de uma janela em Java é feita através da classe JFrame do pacote `javax.swing`.

Tipicamente, cria-se uma classe "filha" dessa classe e, em seguida, modifica-se os parâmetros dessa janela.

Por exemplo, o código abaixo, define uma classe MinhaPrimeiraJanela a partir da classe `JFrame`.

O construtor dessa classe define o seu tamanho para 320 por 240 pixels e o título da janela é definido através da instrução `setTitle(String)` nas linhas 10 e 09 do código abaixo..

```
////****** arquivo MinhaPrimeiraJanela.java
/*01*/ package interfaceGrafica;
/*02*/ import javax.swing.*;
/*03*/ import java.awt.event.*;
/*04*/ import java.awt.*;

/*05*/ public class MinhaPrimeiraJanela extends JFrame
/*06*/ {
/*07*/     public MinhaPrimeiraJanela(String titulo)
/*08*/     {
/*09*/       setTitle(titulo);
/*10*/       setSize(320,240);
/*11*/     }
/*12*/ }
```

Por sua vez, o arquivo abaixo permite executar e testar a classe definida anteriormente.

```
////***** arquivo testeMinhaPrimeiraJanela.java
/*01*/ package interfaceGrafica;
/*02*/ import javax.swing.*;
/*03*/ public class testeMinhaPrimeiraJanela
/*04*/ {  public static void main(String[] a)
/*05*/   { MinhaPrimeiraJanela tela = new
/*06*/          MinhaPrimeiraJanela("Minha Janela");
/*07*/     tela.setVisible(true);  }  }
```

A execução:

A Figura 1 ilustra o execução da classe testeMinhaPrimeiraJanela.

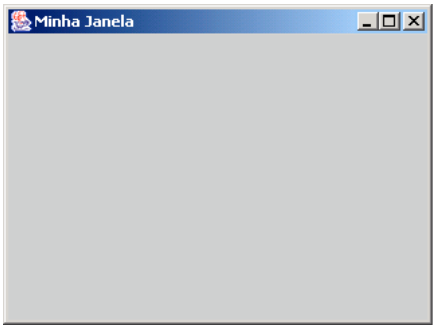


Figura 1-Primeira Janela

A posição da janela na tela do computador não foi definida pelo programador. Logo, a posição adotada é a coordenada (0,0), ou seja, no canto superior a esquerda.

Outro aspecto a ser observado é que ao clicar no ícone do canto superior a direita da janela na tentativa de fechá-la, a janelaapenas desaparece da tela do computador. A aplicação ainda continua "rodando" e a janela ainda existe.

Como resolver essas as questões de posicionamento e finalização da aplicação?

Como centralizar e fechar a janela?

O posicionamento é feito através da instrução `setLocation(int, int)`.

A finalização, por sua vez, não pode ser resolvida de maneira tão simples. Finalizar uma aplicação gráfica ou executar qualquer ação ou evento não é função dos objetos gráficos em Java. Há a necessidade de instruções específicas.

Para a definição dessas ações, pode-se desenvolver uma classe específica para essa tarefa, por exemplo. Essa classe, tipicamente, será filha da classe `WindowAdapter` ou `MouseAdapter`. Essas classes possuem assinaturas de métodos que são executados quando um evento ocorre.

Por exemplo, o arquivo MinhaSegundaJanela.java poderia ser definida conforme o código a seguir. O método `WindowClosing(WindowEvent e)` da classe Eventos definido nas linhas 07 a 08 do código define o que deverá ser feito quando a janela for fechada. Nesse caso, encerrar a aplicação.

A instrução "`addWindowListener( new Eventos());`" adiciona uma instância da classe Eventos ao "responsável" pela janela. Em Java, essa "responsabilidade" é definida como "listener".

A posição da janela é definida pela instrução `setLocation(double,double)` do método `centraliza()` definido nas linhas 18 e 19. As coordenadas que permitem centralizar a janela são obtidas nas linhas 14, 15, 16 e 17. O método `getSize()` das linhas 18 e 19 retornam o tamanho da janela.

Código.

```
////**** arquivo MinhaSegundaJanela.java
/*1*/ package interfaceGrafica;
/*2*/ import javax.swing.*;
/*3*/ import java.awt.event.*;
/*4*/ import java.awt.*;

/*5*/ class Eventos extends WindowAdapter
/*6*/ {
/*7*/     public void windowClosing(WindowEvent e)
/*8*/                 {System.exit(0); }
/*9*/ }

/*10*/ public class MinhaSegundaJanela extends
MinhaPrimeiraJanela
/*12*/ {
/*13*/  private void centraliza()
/*14*/  {int x = ((Toolkit.getDefaultToolkit()).
/*15*/                  getScreenSize()).width;
/*16*/   int y = ((Toolkit.getDefaultToolkit()).
/*17*/                  getScreenSize()).height;
/*18*/   setLocation((x - getSize().width)/2
/*19*/              ,(y - getSize().height)/2);
/*20*/  }

/*21*/  public MinhaSegundaJanela(String titulo)
/*22*/  { super(titulo);
/*23*/    addWindowListener( new Eventos());
/*25*/    centraliza();
/*26*/  }}
```

Teste da classe MinhaSegundaJanela.

```
////***** arquivo testeMinhaSegundaJanela.java
/*01*/ package interfaceGrafica;
/*02*/ import javax.swing.*;
/*03*/ public class testeMinhaSegundaJanela
/*04*/ {  public static void main(String[] a)
/*05*/  {      MinhaSegundaJanela tela = new
/*06*/         MinhaSegundaJanela("Minha Segunda Janela");
/*07*/         tela.setVisible(true);  }       }
```

2. Painel.

Como inserir conteúdo em uma Janela?

Uma janela, conforme definido anteriormente, define apenas uma região. Para a definição de conteúdos dessa janela, há a necessidade da definição de um ou vários painés que irão organizar e armazenar informações.

Um painel é definido a partir da classe JPanel. A linha 12 do código abaixo define um painel de nome `conteudo`. Por motivos de ilustração, o painel tem a sua cor de fundo modificado para cor verde na linha 16.

A associação entre o painel e a janela é feita através do objeto container da classe JFrame, linhas 17 e 18. Ao ser executado, a janela agora terá um fundo verde.

Esse painel permitirá que adicionemos um pouco mais de interatividade conforme descrito na próxima seção.

Código

```
////**************** arquivo MinhaTerceiraJanela.java
/*01*/ package interfaceGrafica;
/*02*/ import javax.swing.*;
/*03*/ import java.awt.event.*;
/*04*/ import java.awt.*;

/*10*/ public class MinhaTerceiraJanela extends
MinhaSegundaJanela
/*11*/ {

/*12*/    JPanel conteudo = new JPanel();

/*13*/    public MinhaTerceiraJanela(String titulo)
/*14*/    {
/*15*/      super(titulo);
/*16*/      conteudo.setBackground(Color.GREEN);
/*17*/      Container fundo = getContentPane();
/*18*/      fundo.add(conteudo);
/*19*/     }
/*20*/ }
```

Arquivo para teste

```
////***** arquivo testeMinhaTerceiraJanela.java
/*01*/ package interfaceGrafica;
```

```
/*02*/ import javax.swing.*;
/*03*/ public class testeMinhaTerceiraJanela
/*04*/ {  public static void main(String[] a)
/*05*/  {      MinhaTerceiraJanela tela = new
/*06*/         MinhaTerceiraJanela("Minha Terceira Janela");
/*07*/    tela.setVisible(true);  } }
```

3. O listener do Mouse

Introdução:

De maneira similar ao exemplo de finalização de aplicação anteriormente apresentado, o mouse possui o seu próprio listener. Como exemplo, o código a seguir define uma classe GerenteMouse filha de MouseAdapter.

Código alterado. As alterações necessárias são representadas em negrito.

```
////******* arquivo MinhaQuartaJanela.java
/*01*/ package interfaceGrafica;
/*02*/ import javax.swing.*;
/*03*/ import java.awt.event.*;
/*04*/ import java.awt.*;

/*05*/ class GerenteMouse extends MouseAdapter
/*06*/ {
/*07*/  int cliqueX, cliqueY;
/*08*/  JPanel j;
/*09*/
/*10*/  public GerenteMouse(JPanel janela)
/*11*/  {j = janela;}
/*12*/  public void mouseReleased(MouseEvent evt)
/*13*/  {
/*14*/   if (evt.getButton() != MouseEvent.BUTTON1)
return;
/*15*/   cliqueX = evt.getX();  cliqueY = evt.getY();

/*16*/   System.out.println(cliqueX+ ","
/*17*/                              + cliqueY);
/*18*/    paintComponent(j.getGraphics());
/*19*/  }

/*20*/  public void paintComponent(Graphics g)
/*21*/  { g.setColor(Color.BLUE);
/*22*/    g.draw3DRect(cliqueX, cliqueY, 100
/*23*/                               , 100,true);
/*24*/  }
/*25*/ }

/*26*/ public class MinhaQuartaJanela extends
MinhaTerceiraJanela
/*27*/ {

/*28*/  public MinhaQuartaJanela(String titulo)
/*29*/  {
/*30*/   super(titulo);
/*31*/   super.conteudo.addMouseListener(new
/*32*/       GerenteMouse(super.conteudo));
/*33*/   }
/*34*/ }
```

Teste da classe MinhaQuartaJanela

```
////***** arquivo testeMinhaQuartaJanela.java
/*01*/ package interfaceGrafica;
/*02*/ import javax.swing.*;
/*03*/ public class testeMinhaQuartaJanela
/*04*/ {  public static void main(String[] a)
/*05*/  {      MinhaQuartaJanela tela = new
/*06*/         MinhaQuartaJanela("Minha Quarta Janela");
/*07*/    tela.setVisible(true);  }        }
```

Outros Métodos do MouseAdapter :

A classe MouseAdapter disponibiliza ainda os seguinte métodos:

- void mouseClicked(MouseEvent e)
  Invoked when the mouse has been clicked on a component.
- void mouseEntered(MouseEvent e)
  Invoked when the mouse enters a component.
- void mouseExited(MouseEvent e)
  Invoked when the mouse exits a component.
- void mousePressed(MouseEvent e)
  Invoked when a mouse button has been pressed on a component.
- void mouseReleased(MouseEvent e)
  Invoked when a mouse button has been released on a component.

Comentários:

Nesse exemplo, a cada clique que o usuário fornecer, um retângulo azul é desenhado no JPanel.

4. Menus

A barra de Menus

Os menus de uma aplicação gráfica devem ser inseridos em uma barra de menus. Uma barra de menus denominada por "menu" é definida através da instrução, por exemplo:

```
JMenuBar menu = new JMenuBar();
```

Como inserir uma barra de menus em um Frame?

Para inserir uma barra de menus em um frame, a seguinte função deve ser chamada:

```
setJMenuBar(menu);
```

Como definir um menu?

Um elemento de Menu que estará visível na barra de menus é um elemento JMenu. Como exemplo, apresenta-se o seguinte código que define um elemento de de menu.

```
JMenu mFile  = new JMenu("File");
     mFile.setMnemonic('F');
```

Nesse exemplo, além de definir um elemento de menu denominado por File, é definido como tecla de acesso a letra 'F'.

Um menu é adicionado na barra de menus, denominada por "menu" por exemplo, através da instrução:

```
menu.add(mFile);
```

Como definir um ítem de menu?

Um ítem de menu pode ser um JMenuItem, como por exemplo:

```
JMenuItem mExit  = new JMenuItem("Exit",'E');
```

Esse ítem pode ser adicionado a um elemento menu, denominado por mFile, por exemplo através da instrução:

```
mFile.add(mExit);
```

Como definir um sub-item de menu?

Quando um ítem de menu for formado por sub-itens, esse ítem de menu não pode ser definido como um JMenuItem e sim como um JMenu.

Por exemplo, o código a seguir irá colocar um ítem de menu chamado "Cor" que será o início de um novo menu, ou sub-menu.

```
JMenu mSelCor  = new JMenu("Cor");
mSelCor.setMnemonic('C');
mFile.add(mSelCor);
```

Um novo elemento "Cor" é definido como um JMenuItem. Esse elemento poderia conter os seguintes sub-itens:

```
JMenuItem sm_corBranca = new JMenuItem("Branca",'B');
JMenuItem sm_corAmarela = new JMenuItem("Amarela",'A');
mSelCor.add(sm_corBranca);
mSelCor.add(sm_corAmarela);
```

Exemplo de Menus.

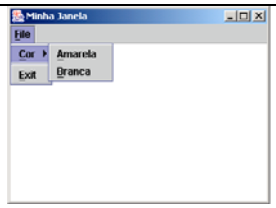


Figura 2-Janela com Menu

A Figura 2 ilustra um exemplo de menu. Nesse exemplo, menus e sub-menus foram definidos com os fragmentos de código Java apresentados anteriormente. O código completo que permite apresentar a janela da Figura 2 é apresentado a seguir.

Nesse código, optou-se por criar a função criarMenus() para o desenvolvimento do menu. Essa função está definida entre as linhas 07 a 25. A linha 28 faz uma chamada a essa função recebendo como resultado dessa chamada, uma barra de menus. Em seguida, o menu é associado ao Jframe por meio da instrução setJMenuBar( ) na linha 29.

Código Java para definição de Menus.

```
  ////**************** arquivo MinhaQuintaJanela.java
  /*01*/ package interfaceGrafica;
  /*02*/ import javax.swing.*;
  /*03*/ import java.awt.event.*;
  /*04*/ import java.awt.*;
  /*05*/ public class MinhaQuintaJanela extends
MinhaQuartaJanela
  /*06*/ {
  /*07*/   private JMenuBar criarMenus()
  /*08*/   {
  /*09*/   JMenuBar menu = new JMenuBar();
  /*10*/   JMenu mFile  = new JMenu("File");
  /*11*/   mFile.setMnemonic('F');
  /*12*/   JMenu mSelCor  = new JMenu("Cor");
  /*13*/   mSelCor.setMnemonic('C');
  /*14*/   sm_corAmarela = new JMenuItem("Amarela",'A');
  /*16*/   sm_corBranca = new  JMenuItem("Branca",'B');
  /*18*/   mSelCor.add(sm_corAmarela);
  /*19*/   mSelCor.add(sm_corBranca);
  /*20*/   mExit  = new JMenuItem("Exit",'E');
  /*21*/   mFile.add(mSelCor);
  /*22*/   mFile.addSeparator();
  /*23*/   mFile.add(mExit);
  /*24*/   menu.add(mFile);
  /*25*/   return menu;}
  /*26*/ public MinhaQuintaJanela(String titulo)
  /*27*/ { super(titulo);
  /*28*/   BarraDeMenus = criarMenus();
  /*29*/   setJMenuBar(BarraDeMenus);
  /*30*/ }
  /*31*/ JMenuBar BarraDeMenus; JMenuItem sm_corAmarela;
  /*33*/   JMenuItem sm_corBranca;    JMenuItem mExit;}
```

Teste da Classe MinhaQuintaJanela

```
////***** arquivo testeMinhaQuintaJanela.java
/*01*/ package interfaceGrafica;
/*02*/ import javax.swing.*;
/*03*/ public class testeMinhaQuintaJanela
/*04*/ {  public static void main(String[] a)
/*05*/  {      MinhaQuintaJanela tela = new
/*06*/         MinhaQuintaJanela("Minha Quinta Janela");
/*07*/    tela.setVisible(true);  }     }
```

Como adicionar ações aos elementos do menu?

A adição de interatividade pode ser feita com duas estratégias:

1. Criar uma classe específica para fazer a função de "listener" das ações.
2. Ou definir códigos dentro da classe filha de JFrame.

Fica registrado aqui que não existe uma forma mais correta que a outra. Ambas as formas possuem aspectos positivos e negativos que acabam influenciando o programador experiente.

Neste material de apoio, as duas formas serão analisadas.

Criação de uma classe específica para fazer a função de "listener" das ações.

Essa estratégia é simples e pode ser resumida a:

1. definir uma classe
2. criar uma instância dessa classe e
3. associar essa instância ao elemento ou ítem de Menu.

Exemplo:

```
class Acoes implements ActionListener
{
    public void actionPerformed(ActionEvent evt)
    {
      if (evt.getSource() instanceof JMenuItem )
      {JMenuItem quem =
                        (JMenuItem)evt.getSource();
       if (quem.getActionCommand() == "Exit")
           System.exit(0);
      }
} }
```

Exemplo de criação de instância dessa classe e associação com um elemento de menu:

```
JMenuItem mExit  = new JMenuItem("Exit",'E');
mExit.addActionListener(new Acoes());
```

Quando há a necessidade de manipular atributos de algum objeto, pode-se definir um atributo para fazer referência a esse objeto. Por exemplo:

```
class Acoes2 implements ActionListener
{    JPanel conteudo;
     Acoes2(JPanel c) {conteudo = c;}
     public void actionPerformed(ActionEvent evt)
   {  if (evt.getSource()instanceof JMenuItem )
      {JmenuItem quem = (JMenuItem)evt.getSource();
       if (quem.getActionCommand() == "Amarela")
           conteudo.setBackground(Color.YELLOW);
} } }
```

Como definir código dentro da classe filha de JFrame?

A outra estratégia é definir o código de maneira isolada, como por exemplo:

```
JPanel conteudo = new JPanel();
JMenuItem sm_corAmarela = new
           JMenuItem("Amarela",'A');
sm_corAmarela.addActionListener(
  new AbstractAction()
    {public void actionPerformed(ActionEvent evt)
      {conteudo.setBackground(Color.YELLOW);} }
);
```

O JMenuItem sm_corAmarela após instanciado tem o método addActionListener definido com um código específico para ele.

A grande vantagem dessa estratégia em relação à outra apresentada anteriormente é a possibilidade de fazer referência ou chamar instruções de objetos da classe filha de JFrame. Por exemplo, se o usuário selecionar esse elemento de menu, o painel denominado "conteudo" tem a sua cor de fundo modificada. Como a definição do listener está dentro da classe filha de JFrame, o método pode fazer referência a qualquer elemento do frame.

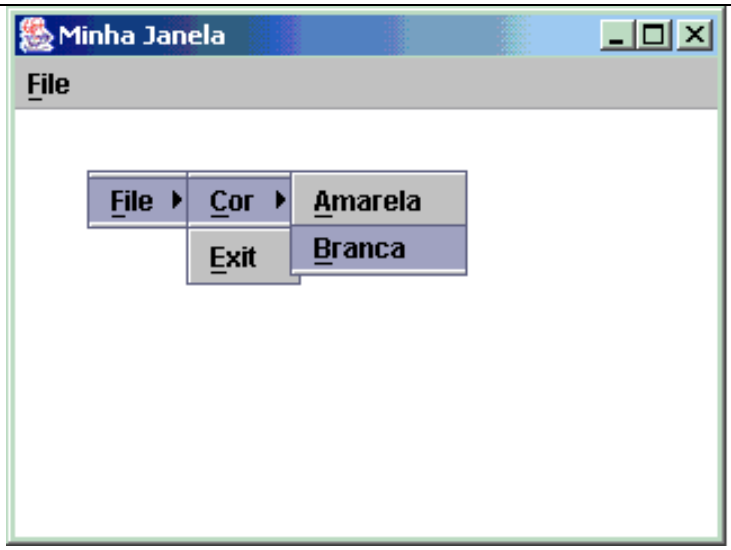
Código alterado.

```
////**************** arquivo MinhaSextaJanela.java
/*01*/ package interfaceGrafica;
/*02*/ import javax.swing.*;
/*03*/ import java.awt.event.*;
/*04*/ import java.awt.*;

/*05*/ class Acoes implements ActionListener
/*06*/ {
/*07*/    public void actionPerformed(ActionEvent evt)
/*08*/    {
/*09*/      if (evt.getSource()   instanceof JMenuItem )
/*10*/      {JMenuItem quem =
/*11*/           (JMenuItem)evt.getSource();
/*12*/       if (quem.getActionCommand() == "Exit")
/*13*/           System.exit(0);
/*14*/      }
/*15*/    } }

/*16*/ class Acoes2 implements ActionListener
/*17*/ {
/*18*/    JPanel conteudo;
/*19*/    Acoes2(JPanel c) {conteudo = c;}
/*20*/     public void actionPerformed(ActionEvent evt)
/*21*/     {
/*22*/      if (evt.getSource()   instanceof JMenuItem )
/*23*/      {JMenuItem quem =
/*24*/           (JMenuItem)evt.getSource();
/*25*/       if (quem.getActionCommand() == "Amarela")
conteudo.setBackground(Color.YELLOW);
/*26*/       if (quem.getActionCommand() == "Branca")
conteudo.setBackground(Color.WHITE);
/*27*/       } } }

/*28*/ public class MinhaSextaJanela extends
MinhaQuintaJanela
/*29*/ {
/*30*/   JPanel conteudo = super.conteudo;
/*31*/   public MinhaSextaJanela(String titulo)
/*32*/   {
```

```
  /*33*/          super(titulo);
  /*34*/          super.mExit.addActionListener(new Acoes());
  /*35*/          super.sm_corBranca.addActionListener
  /*36*/                   (
  /*37*/                    new AbstractAction()
  /*38*/                    { public void
actionPerformed(ActionEvent evt)
  /*39*/
{conteudo.setBackground(Color.WHITE);}
  /*40*/                              }
  /*41*/                     );
  /*42*/          super.sm_corAmarela.addActionListener(new
Acoes2(super.conteudo));
  /*43*/     }}
```

Exemplo de aplicação para testar o MinhaSextaJanela

```
  ////***** arquivo testeMinhaSextaJanela.java
  /*01*/ package interfaceGrafica;
  /*02*/ import javax.swing.*;
  /*03*/ public class testeMinhaSextaJanela
  /*04*/ {  public static void main(String[] a)
  /*05*/  {      MinhaSextaJanela tela = new
  /*06*/         MinhaSextaJanela("Minha Sexta
Janela");
  /*07*/    tela.setVisible(true);  } }
```

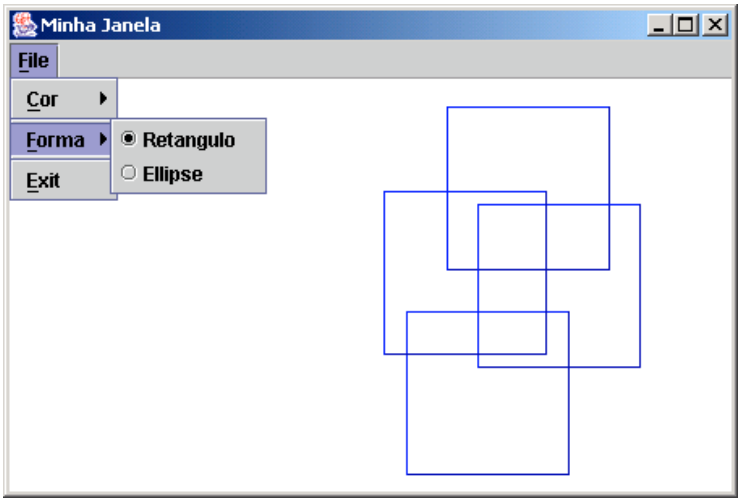
Como definir menus Flutantes?	Um menu flutuante é criado de maneira similar a um menu comum. Um menu flutuante JPopupMenu é instanciado e itens são adicionados a ele. O exemplo abaixo, finaliza a aplicação. As modificações em negrito adicionam um menu flutante com as mesmas funcionalidades do menu definido anteriormente. Um menu flutuante é exibido através a instrução show(Component, int, int). A linha 17 do código abaixo, define a visualização do menu. Um destaque deve ser dado à linha 16. Nessa linha é verificado se o usuário clicou chamando o menu pop-up (no Windows isso significa clicar com o botão direito do mouse) ou se o clique foi com o botão esquerdo chamando então a função de desenho.
Exemplo de execução:	
Código Completo da aplicação:	``` ////**************** arquivo MinhaSetimaJanela.java /*01*/ package interfaceGrafica; /*02*/ import javax.swing.*; /*03*/ import java.awt.event.*; /*04*/ import java.awt.*; /*05*/ class GerenteMouse2 extends GerenteMouse /*06*/ { /*07*/ private JPopupMenu menu; /*08*/ public GerenteMouse2(JPanel janela /*09*/ , JPopupMenu _menu) /*10*/ {super(janela); menu = _menu;} /*11*/ public void mouseReleased(MouseEvent evt) /*12*/ { /*13*/ super.mouseReleased(evt); /*14*/ cliqueX = evt.getX(); /*15*/ cliqueY = evt.getY(); /*16*/ if (evt.isPopupTrigger()) /*17*/ {menu.show(evt.getComponent() /*18*/ ,cliqueX, cliqueY);} /*19*/ } /*20*/ } ```

```
  /*21*/ public class MinhaSetimaJanela extends
MinhaSextaJanela
  /*22*/ {
  /*23*/   JPanel conteudo = super.conteudo;

  /*24*/   public JPopupMenu criarMenuPopUp()
  /*25*/   {
  /*26*/   JPopupMenu saida = new JPopupMenu();

  /*27*/   JMenu mFile  = new JMenu("File");
  /*28*/   mFile.setMnemonic('F');
  /*29*/   JMenu mSelCor  = new JMenu("Cor");
  /*30*/   mSelCor.setMnemonic('C');
  /*31*/   sm_corAmarela = new JMenuItem("Amarela",'A');
  /*32*/   sm_corBranca = new  JMenuItem("Branca",'B');
  /*33*/    sm_corAmarela.addActionListener(new
Acoes2(super.conteudo));
  /*34*/   sm_corBranca.addActionListener(new
Acoes2(super.conteudo));

  /*35*/   mSelCor.add(sm_corAmarela);
  /*36*/   mSelCor.add(sm_corBranca);
  /*37*/   mExit  = new JMenuItem("Exit",'E');
           mExit.addActionListener(new Acoes());
  /*38*/   mFile.add(mSelCor);
  /*39*/   mFile.addSeparator();
  /*40*/   mFile.add(mExit);

  /*41*/   saida.add(mFile);
  /*42*/   saida.add(mExit);
  /*43*/   return saida;
  /*44*/  }
  /*45*/  private JPopupMenu menuFlutuante;

  /*46*/ public MinhaSetimaJanela(String titulo)
  /*47*/ {
  /*48*/   super(titulo);
  /*49*/   menuFlutuante = criarMenuPopUp();
  /*50*/   super.conteudo.addMouseListener(new
GerenteMouse2(conteudo,menuFlutuante));
  /*51*/ }
  /*52*/}
```

Desafio: Ao apresentar o programa anterior ao usuário, ele solicitou uma pequena modificação. Adicionar uma opção no programa que permita selecionar a forma geométrica a ser desenhada na tela. O usuário deverá poder selecionar uma das seguintes formas:

- Retângulo (opção padrão).
- Círculo ou Elipse. O código java para desenhar essa figura é ilustrado a seguir: `g.drawOval(cliqueX, cliqueY, 100, 100);`

O usuário ainda sugeriu que o menu tivesse a seguinte aparência:



Tarefa de Casa: Implementar o pedido do usuário!