Tema 9. Gràfics. Java Swing

Introducció

Fins ara hem estat utilitzant la consola i el teclat a les nostres aplicacions per tal de comunicar-nos amb l'usuari. En aquest tema utilitzarem l'API de Java per al desenvolupament de components gràfics Swing.

Swing és un conjunt d'eines per a Java. És una API que proporciona interfície gràfica d'usuari (GUI) per a programes de Java. Entre d'altres Swing conté: frames, botons, textbox, menús a banda dels events associats als elements.

Swing va ser desenvolupat per a proporcionar un conjunt més complex de components GUI que l'anterior Abstract Window Toolkit (AWT). Swing proporciona un aspecte i aparença natiu que emula diverses plataformes, també dóna suport a extensions d'aspecte i aparença que permeten a les aplicacions tindre un aspecte i apareça que no guarda relació amb la plataforma subjacent.

Huí en dia la majoria de desenvolupadors Java utilitzen Swing o JavaFX per construir interfícies d'usuari

La llibreria Swing

Com tot en Java, swing no anava a ser menys, està dissenyat com una llibreria de classes, interfícies, recursos, etc.. per la construcció d'insterfícies gràfiques. Swing conté tres APIs una per a components 2D, una altra per al Drag & Drop i l'última per facilitar l'accés. Swing està basada en AWT però és independent de la plataforma, és a dir, sempre mostra la mateixa aparença independentment del sistema en el que s'execute l'aplicació a diferència d'AWT.

Amb Swing podem fer:

- Marcs: Finestres amb de títol, menú, botons maximitzar, minimitzar i tancar.
- Contenidors: poden agrupar diversos controls.
- Botons
- Etiqutes: text
- Camps i arees de text.
- Desplegables

Per tal de poder utilitzar les llibreries Swing de Java hauriem d'afegir al nostre codi el següent:

```
import javax.swing.*;
```

Components i contenidors

En general, els components Swing es deriven de la classe JComponent. Les úniques excepcions són els quatre contenidors de nivell superior. JComponent proporciona la funcionalitat que és comú a tots els components. Per exemple, JComponent admet la look & feel conectables. JComponent hereta les classes AWT Container i Component. Per tant, un component Swing està integrat i es compatible amb un component AWT.

Tots els components de Swing estan representats per classes definides dins del paquet javax.swing com hem explicat abans. La següent taula mostra els noms de classe per als components Swing inclosos els contenidors.

Components Java Swing

JApplet	JButton	JCheckBox	JCheckBoxMenuItem	JColorChooser	JComboBox

JComponent	JDesktopPane	JDialog	JEditorPane	JFileChooser	JFormattedTextField
JFrame	JInternalFrame	JLabel	JLayer	JLayeredPane	JList
JMenu	JMenuBar	JMenuItem	JOptionPane	JPanel	JPasswordField
JPopUpMenu	JProgressBar	JRadioButton	JRadioButtonMenuItem	JRootPane	JScrollBar
JScrollPane	JSeparator	JSlider	JSpineer	JSplitPane	JTabbedPane
JTable	JTextArea	JTextField	JTextPane	JTogglebutton	JToolBar
JToolTip	JTree	JViewport	JWindows		

La majora de noms solen ser prou significatius i resulta intuitiu a primer cop d'ull saber quina és la finalitat de cadascun dels components.

Contenidors Java Swing

Swing defineix dos tipus de contenidors. Els primers són contenidors de nivell superior: JFrame, JApplet, JWindow i JDialog. (JApplet, que admet applets basats en Swing, ha estat descartat per JDK 9.) Aquests contenidors no hereten JComponent. No obstant això, hereten les classes AWT Component i Container. A diferència d'altres components de Swing, que són lightweight, els contenidors de nivell superior són heavyweight. Això fa que els contenidors de nivell superior siguin un cas especial a la biblioteca de components Swing.

Com el seu nom indica, un contenidor de nivell superior ha d'estar a la part superior d'una jerarquia de contenció. Un contenidor de nivell superior no està contingut en cap altre contenidor. A més, cada jerarquia de contenció ha de començar amb un contenidor de nivell superior. El més comunament utilitzat per les aplicacions és JFrame.

El segon tipus de contenidor compatible amb Swing és el contenidor lightweight. Els contenidors lightweight hereten JComponent. Exemples de contenidors lightweight són JPanel, JScrollPane i JRootPane. Els contenidors lightweight sovint es fan servir per a organitzar i administrar col·lectivament grups de components relacionats perquè un contenidor lightweight es pot contenir dins d'un altre contenidor. Per tant, pot utilitzar contenidors lightweight per crear subgrups de controls relacionats que estan continguts dins d'un contenidor extern.

Swing vs AWT

És possible utilitzar Swing i AWT a la mateixa interfície, però podria donar problemes. És recomanable utilitzar només components Swing, ja que tot component AWT té el seu equivalent Swing.

La majoria de classes Swing comencen per J, per exemple: JButton, JFrame... encara que també existeixen Frame i Button que pertanyen a la llibreria AWT. Cal tindre molta cura en no oblidar de posar la 'J' davant dels components ja que si no podria portar a inconsistències a causa de mesclar components.

A l'actualitat quasi tota la programació gràfica en Java es fa amb Swing que implemente una interfície gràfica normalment va a tenir quatre tipus d'elements:

1. Un contenidor de nivell superior: un marc (JFrame), un applet (JApplet) o bé objectes JDialog. Aquests contenidors no estan dins d'una altra finestra, són les finestres principals.
2. Components de la interfície gràfica: botons, camps de text, etcètera, que se situen a la finestra principal o en contenidors.
3. Contenidors dissenyats per altres elements de la interfície; JPanel i JScrollPane són dos contenidors i, a el mateix temps, són components.

4. Elements per a la gestió d'esdeveniments.
   Els components sempre s'afegeixen a una làmina o panell; pot ser la de el marc, o bé un panell tipus JPanel.
   En general, sempre es creen classes derivades de les classes contenidors de nivell superior; tot marc serà una subclasse de JFrame, a l'igual que un applet és una subclasse de JApplet.

Creació d'un JFrame

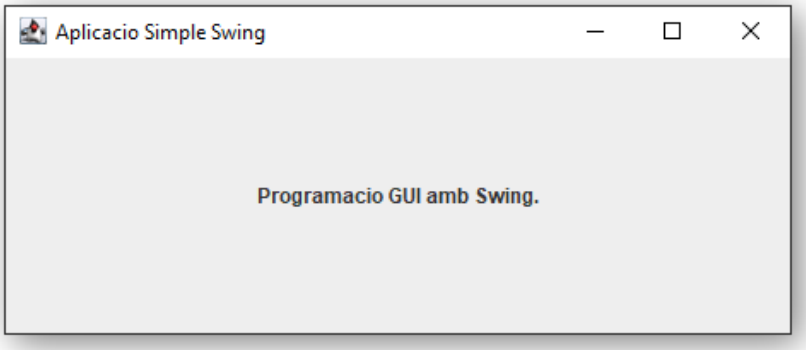
La diferència entre crear programes que es comuniquen amb l'usuari mitjançant el terminal i el teclat i fer programes amb Swing és considerable. Al crear una aplicació Swing s'han de tenir en compte conceptes nous com el **threading**. Però per entendre-ho tot millor, farem un programa Swing d'exemple.

Per al programa d'exemple crearem un **JFrame** al qual li posarem una **JLabel**

Exemple 1. Programa simple Swing

```
// Un simple programa Swing

import javax.swing.*;
public class SwingDemo {
    SwingDemo(){
        JFrame jfrm=new JFrame("Aplicació Simple Swing");        // Crea un nou
contenidor JFrame.
        jfrm.setSize(475,200);                                   // Establim el tamany
inicial
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);     // Acabe el programa
quan l'usuari tanque l'aplicació.
        JLabel jLabel= new JLabel("Programació GUI amb Swing."); // Creem una etiqueta
        jLabel.setHorizontalAlignment(SwingConstants.CENTER);    // Situem l'etiqueta
al centre del frame
        jfrm.add(jLabel);                                        // Afegim l'etiqueta
al frame
        jfrm.setVisible(true);                                   // Visualitzem el
marc.
    }
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run(){
                new SwingDemo();
            }
        });
    }
}
```

El resultat del codi anterior seria el següent:



JFrame és un contenidor de nivell superior que s'utilitza de forma comú a les aplicacions Swing, mentre que **JLabel** és una etiqueta que ens servirà per posar text.

Expliquem amb més detall el que fa la funció main:

- SwingUtilities.invokeLater()
  - static void invokeLater(Runnable doRun): Executa un doRun.run() per tal que s'execute de forma asincrona al AWT event dispatching thread. A aquets mètode se li ha de passar un objecte Runnable
- new Runnable() { }:
  - Objecte Runnable que s'ha de passar al mètode invokeLater
- public void run() { }: Mètode run que s'executa de l'objecte runnable que se li passa a InvokeLater
- new SwingDemo()
  - Codi del mètode run de l'objecte runnable.

Una aplicació que implemente una interfície gràfica normalment va a tindre quatre tipus d'elements.

1. Un contenidor de nivell superior: un marc (JFrame), un applet (JApplet) o bé diàlegs (JDialog) que faran el paper de finestra principal.
2. Components de la interfície gràfica: botons, camps de text, etc, que s'afegiran al contenidor de nivell superior anterior.
3. Contenidors dissenyats per altres elements de la interfície: JPanel i JScrollPane són dos contenidors i al mateix temps són components.
4. Elements per la gestió d'events.

Els components sempre s'afegeixen a una làmina o paper; pot ser la del marc o bé un panel tipus JPanel. En general, sempre es creen classes derivades de les classes contenidores de nivell superior. Tot marc serà una subclasse de JFrame, de la mateixa forma que un applet és una subclasse de JApplet

JFrame

La classe JFrame és un tipus de contenidor que hereta de la classe java.awt.Frame. JFrame funciona com la finestra principal on components com etiquetes, botons, camps de text etc, s'afegeixen per crear una interfície gràfica.

Mètodes

Mètode	**Descripció**
JFrame()	Constructor de la classe. Crea un marca sense títol
JFrame (String titol)	Crea un marc amb títol.
void setTitle (String titlo)	Estableix el títol del marc

Mètode	Descripció
void setIconImage (Image img)	Estableix la icona del marc
void setDefaultCloseOperation (int op)	programa el comportament del marca quan es tanca. Possibles valors: EXIT_ON_CLOSE, DO_NOTHING_ON_CLOSE, DISPOSE_ON_CLOSE, HIDE_ON_CLOSE
Container getContentPane ()	proporciona el layout de continguts del marc, és a dir, les vores
void setResizable (boolean r)	si r val true, es pot redimensionar el marc
void add (Component c)	Afegeix el component c al marc
void removeComponent (Component c)	elimina el component c del marc
Component add (Component c, int p)	Coloca el component c a la posició p del marc
void setLayout (LayoutManager mngr)	estableix la forma de distribuir el components al marc, els quals normalment es distribueixen en posicions relatives, segons el layout que tinga associat el marc
void setVisible (boolean b)	fa visible el component si b és true
void setBounds (int x, int y, int ample, int alt)	situa el component i canvia el seu tamany
void setLocation (int x, int y)	situa el component a les coordenades x i y
void setLocation (Point p)	situa el component al punt p
void setSize (int ample, int alt)	estableix les dimensions del component
void setSize (Dimension dim)	estableix la dimensió del component a dim

Layout Manager - Gestor de posicionament

A diferència d'altres aplicacions que es dediquen al disseny d'interfícies gràfiques, les posicions dels elements dintre del nostre contenidor no es fan en pixels ni valors absoluts, s'utilitzen els **Layout Manager** que ho podriem traduir com "gestors de disposició" o "gestor de plantilla" o "gestor de posicionament". Aquestos gestors de posicionament són elements que implementen la interfície "LayoutManager".

Cada cotenidor té associat un Layout Manager que pot ser: BorderLayout, FlowLayout, GridLayout, BoxLayout, GridBagLayout, CardLayout, SpringLayout...

Veiem un exemple:

Exemple 2. Marc de prova

```
```

```
JFrame marc = new JFrame("MarcProva");
marco.setLayout(new GridLayout(3,4));
```

El que hem fet al codi anterior és crear i instanciar un nou marc de tipus JFrame amb el títol "MarcProva" i després hem establert el seu gestor de posicionament de tipus GridLayout.

A continuació detallarem els més utilitzats

BorderLayout

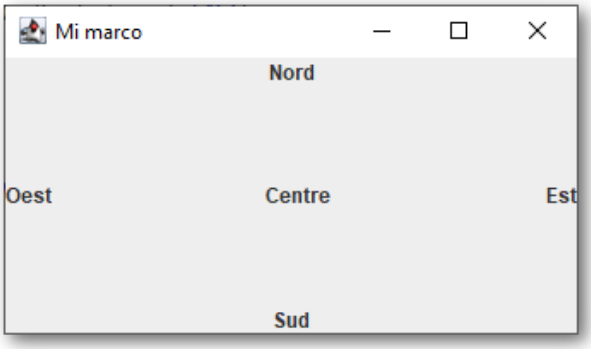
Gestor de posicionament per defecte dels marcs (JFrame) i diàlegs (JDialog). Divideix al contenidor en cinc zones: nord, sud, est, oest i centre, que es corresponen a: BorderLayout.NORTH, BorderLayout.SOUTH, BorderLayout.EAST, BorderLayout.WEST i BorderLayout.CENTER

Veiem un exemple:

Exemple 3. Marc amb vora

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

public class MarcBorder extends JFrame
{
    static int AMPLE =350;
    static int ALT = 200;
    public MarcBorder()
    {
        super("Títol Marc");
        add(new JLabel("Nord", SwingConstants.CENTER), BorderLayout.NORTH);
        add(new JLabel("Sud", SwingConstants.CENTER), BorderLayout.SOUTH);
        add(new JLabel("Centre", SwingConstants.CENTER), BorderLayout.CENTER);
        add(new JLabel("Oest", SwingConstants.CENTER), BorderLayout.WEST);
        add(new JLabel("Est", SwingConstants.CENTER), BorderLayout.EAST);
        setSize(AMPLE,ALT);
        setVisible(true);
    }
    public static void main(String args[])
    {
        MarcBorder marc = new MarcBorder();
        marc.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

El resultat del codi anterior és:



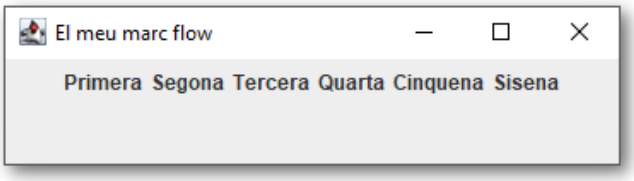
FlowLayout

Amb aquest gestor podem colocar els elements d'esquerra cap a dreta i de dalt cap avall. La classe FlowLayout disposa de diversos constructors, un d'ells ens permet establir l'alineació dels components **FlowLayout(int align)** on align pot ser: FlowLayout.RIGHT, FlowLayout.CENTER, FlowLayout.LEFT.

Exemple 4. Flowlayout

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

public class MarcFlow extends JFrame
{
    static int AMPLE =175;
    static int ALT = 100;
    public MarcFlow()
    {
        super("El meu marc");
        setLayout(new FlowLayout());
        add(   new JLabel("Primera"));
        add(new JLabel("Segona"));
        add(new JLabel("Tercera"));
        add(new JLabel("Quarta"));
        add(new JLabel("Cinquena"));
        add(new JLabel("Sisena"));
        setSize(AMPLE,ALT);
        setVisible(true);
    }
    public static void main(String args[])
    {
        MarcFlow marc = new MarcFlow();
        marc.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

I aquest és el resultat:



GridLayout

Aquest gestor de posicionament distribueix els components del container associat en forma de graella de cel·les iguals en forma de quadricula de dalt cap avall i de esquerra a dreta, és a dir, en files i columnes.

Exemple 5. GridLayout

```
import java.awt.*;
import javax.swing.*;
public class MarcGrid extends JFrame
    {
    static int AMPLE =175;
    static int ALT = 100;
    public MarcGrid()
    {
        super("El meu marc");
        setLayout(new GridLayout(3,2,15,15));
        add(new JLabel("Primera"));
        add(new JLabel("Segona"));
        add(new JLabel("Tercera"));
        add(new JLabel("Quarta"));
        add(new JLabel("Cinquena"));
        add(new JLabel("Sisena"));
        setSize(AMPLE,ALT);
        setVisible(true);
    }

    public static void main(String args[])
    {
        MarcGrid marc = new MarcGrid();
        marc.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Resultat del codi anterior:

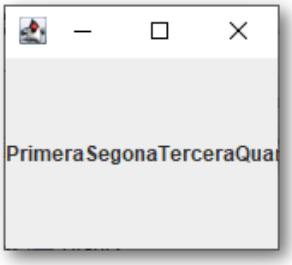
BoxLayout

Aquest gestor coloca els components en una única fila o una única columna, s'assembla a una caixa amb orientació horitzontal o vertical; el constructor necessita un argument amb el contenidor que es va a utilitzar i l'orientació que pot ser: BoxLayout.X_AXIS o BoxLayout.Y_AXIS.

El format del constructor és: BoxLayout (Container desti, int orientacio)

Exemple 6. BoxLayout

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

public class MarcBox extends JFrame
{
    static int AMPLE =175;
    static int ALT = 150;

    public MarcBox()
    {
        super("El meu marcBox");
        JPanel panel = new JPanel() ;
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.add(new JLabel("Primera"));
        panel.add(new JLabel("Segona"));
        panel.add(new JLabel("Tercera"));
        panel.add(new JLabel("Quarta"));
        panel.add(new JLabel("Cinquena"));
        panel.add(new JLabel("Sisena"));
        add(panel);
        setSize(AMPLE,ALT);
        setVisible(true);
    }

    public static void main(String args[])
    {
        MarcBox miMarcBox = new MarcBox();
        miMarcBox.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



BoxLayout - Box

Box o caixa, és un contenidor que té com gestor de posicionament (LayoutManager) predeterminat un BoxLayout; utilitzant aquest contenidor no cal crear un panel perquè directament es crea un objecte Box i a continuació s'afegeixen els components.

La classe Box disposa de dos mètodes static (mètodes factoria) que creen l'objecte, i són:

```
Box.createHorizontalBox();
Box.createVerticalBox();
```

Aleshores, per crear un Box no s'utilitza el constructor sino que es crida a un d'aquestos dos mètodes; per un objecte Box amb orientació horitzontal, per exemple:

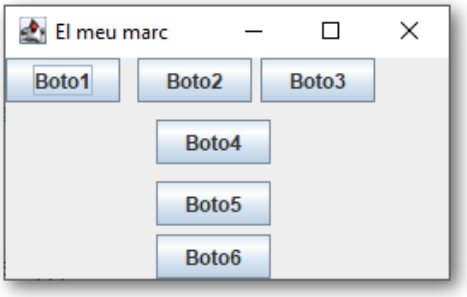
```
Box caixaHoriz = Box.createHorizontalBox();
```

Al contenidor se li afegeix el component caixaHoriz.add(element) i després el contenidor al marc.

Exemple 7. BoxLayout - Box

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

public class Marc2Box extends JFrame
{
    static int AMPLE =275;
    static int ALT = 175;

    public Marc2Box()
    {
        super("El meu marc");
        JButton b1 = new JButton("Boto1");
        JButton b2 = new JButton("Boto2");
        JButton b3 = new JButton("Boto3");
        JButton b4 = new JButton("Boto4");
        JButton b5 = new JButton("Boto5");
        JButton b6 = new JButton("Boto6");
        Box caixaH = Box.createHorizontalBox(); // método factoría
        caixaH.add(b1);
        // separación horizontal de 10 pixeles
        caixaH.add(Box.createHorizontalStrut(10));
        caixaH.add(b2);
        // zona rígida, separación horizontal
        caixaH.add(Box.createRigidArea(new Dimension(5,5)));
        caixaH.add(b3);
        add(caixaH,BorderLayout.NORTH);

        Box caixaV = Box.createVerticalBox();
        caixaV.add(Box.createHorizontalStrut(70));
        caixaV.add(b4);
        // separación horizontal 10 pixeles
        caixaV.add(Box.createVerticalStrut(10));
        caixaV.add(b5);
        caixaV.add(Box.createRigidArea(new Dimension(5,5)));
        caixaV.add(b6);
        add(caixaV,BorderLayout.CENTER);
        setSize(AMPLE,ALT);
        setVisible(true);
    }
```

```
    public static void main(String args[])
    {
        Marc2Box marcBox2 = new Marc2Box();
        marcBox2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



Combinar Layouts

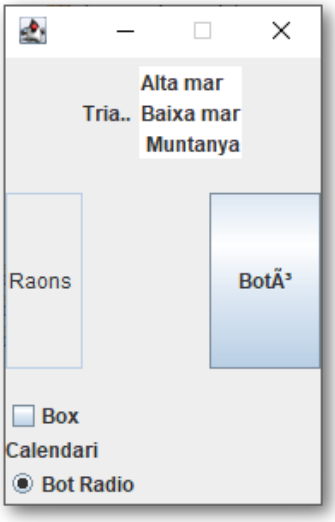
JFrame, JDialog i JPanel només poden tindre un gestor de posicionament però si agafem un contenidor de primer nivell com un marc, podríem combinar diversos layouts niats (nested). Al següent exemple podem veure una combinació de tres gestors de posicionament en un mateix contenidor.

Exemple 8. Combinació de disposicions (layouts)

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

public class MarcGestors extends JFrame
{
    public MarcGestors()
    {
        JPanel pa1 = new JPanel(new FlowLayout());
        JPanel pa2 = new JPanel(new BorderLayout());
        JPanel pa3 = new JPanel();
        pa3.setLayout(new BoxLayout(pa3, BoxLayout.Y_AXIS));

        // componentes del panel 1
        String [] opc = {"Alta mar", "Baixa mar", " Muntanya"};
        pa1.add(new JLabel ("Tria..", JLabel.CENTER));
        pa1.add(new JList(opc));
        pa1.add(new JButton("Prèmer"));

        // componentes del panel 2
        JTextField j = new JTextField("Raons ");
        j.setEditable(false);
        pa2.add(j,BorderLayout.WEST);
        pa2.add(new JButton("Botó"),BorderLayout.EAST);

        // componentes del panel 3
        pa3.add(new JCheckBox("Box ",false));
        pa3.add(new JLabel ("Calendari", JLabel.CENTER));
        pa3.add(new JRadioButton("Bot Radio", true));

        // asigna layout al marco y se ponen los paneles
```

```
            setLayout( new BorderLayout(10, 15));
            add(pa1,BorderLayout.NORTH);
            add(pa2,BorderLayout.CENTER);
            add(pa3,BorderLayout.SOUTH);
        }

        public static void main(String[] args) {
            MarcGestors m;
            m = new MarcGestors();
            m.setSize(200,300);
            m.setLocation(20,200);
            m.setResizable(false);
            m.setVisible(true);
            m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
    }
```

El resultat seria:



Desactivar gestor de posicionament

Per defecte un contenidor sempre té associat un gestor de posicionament (layout) però de vegades ens pot resultar interessant desactivar aquest gestor de posicionament. Utilitzarem el mètode setLayout i li passarem per parametre null.

Etiquetes

Una etiqueta és un component només lectura al qual li podem modificar el text que conté i que normalment s'utilitza per posar algun missatge significatiu de la nostra interfície a l'usuari final. En altres paraules, són components de text fixe que només mostren text però que no reben cap event. Per gestionar les etiquetes a Java utilitzem la classe **JLabel**

Mètodes

- **JLabel ():** Crea una etiqueta sense text associat.
- **JLabel (String msg):** Crea una etiqueta amb el text msg.
- **JLabel (String msg, Icon icona):** Crea una etiqueta amb el text msg i la icona icon.

- **JLabel (String msg, int align):** crea una etiqueta amb el text msg alineat segons el segon argument que pot ser SwingConstants.CENTER, SwingConstants.LEFT o SwingConstants.RIGHT.
- **public String getText ():** torna el text de l'etiqueta
- **public void setText (String msg):** estableix el text de l'etiqueta.

Botons

A Swing es defineixe diversos tipus de botons, la classe base dels quals és **Abstract Button**, és una classe abstracta que encapsula propietats i mètodes comuns als diversos tipus de botons

Mètodes

- **void setText (String text)**: Estableix el text del botó.
- **String getText ()**: Obté el text del botó.
- **boolean isSelected()**: true si s'ha seleccionat el botó.
- **void setSelectes (boolean b)**: selecciona el botó.
- **void doClick (int temps)**: tria el botó durant temps milisegons
- **void setIcon (Icon icona)**: estableix la icona del botó.
- **void setMnemonic (int mnemonic)**: relaciona una tecla amb el botó.
- **void addActionListener(ActionListener al)**: assigna un listener per controlar events.

JButton

La classe JButton representa el botó comú; es crea especificament una cadena, una icona, ambdos, o un element encara sense especificar; els constructors de la classe són:

- **JButton()**: Constructor de la classe
- **JButton(String text)**: Constructor amb text del botó.
- **JButton(String text, Icon icona)**: Constructor amb text i icona.

La classe deriva d'AbstractButton per tant tots els seus mètodes estaran disponibles també.

```
JButton b1, b2, b3;
b1 = new JButton();
b2 = new JButton("GROC");
b3 = new JButton(new LibroIcon());
```

Botons amb dos estats

JToogleButton és la classe base dels botons amb dos estats; JRadioButton que s'utilitza per definir un grup de botons d'opció única; per agrupar botons d'opció única s'utilitza la classe ButtonGroup, primer es crea un objecte ButtonGroup (constructor sense arguments); a continuació s'afegeix JRadioButton amb el mètode de ButtonGroup, add(AbstracButton b).

Constructors de JRadioButton

- **JRadioButton()**: Constructor per defecte.
- **JRadioButton(String msg)**: Constructor amb text del botó.
- **JRadioButton(String msg, boolean sel)**: Constructor amb el text i si sel és true, el botó ja estaria seleccionat.

Exemple 9. Diversons botons

```
import javax.swing.*;
class PanelJRadio extends JPanel
{
    ButtonGroup grb;
    JRadioButton jr1, jr2, jr3;

    public PanelJRadio() {
        grb = new ButtonGroup();
        setLayout(new GridLayout(4,1));
        add (new JLabel("Selecció excloent"));

        // se crea botón de radio, se añade al panel y a la agrupación
        jr1 = new JRadioButton("Avió", false);
        add(jr1);
        grb.add(jr1);

        // se crea botón de radio, se añade al panel y a la agrupación
        jr2 = new JRadioButton("Tren", false);
        add(jr2);
        grb.add(jr2);

        // se crea botón de radio, se añade al panel y a la agrupación
        jr3 = new JRadioButton("Cotxe", false);
        add(jr3);
        grb.add(jr3);
    }
}
```

JCheckBox o també anomenat: casella de verificació, check o checkbox.

Constructors de JCheckBox

- **JCheckBox():** Constructor per defecte.
- **JCheckBox(String text):**: Constructor amb el text associat.
- **JCheckBox(String text, boolean sel):** Constructor amb text i si sel és true, la casella vindria ja marcada.

La classe **JComboBox** no està a la jerarquia de botons, no deriva d'AbstractButton; combina en un sol component un botó amb una llista d'elements. Una **JComboBox** s'utilitza per crear una llista desplegable a la que es poden afegir opcions, editarles o fer seleccions.

Constructors de JComboBox

- **JComboBox():** Constructor per defecte.
- **JComboBox(Object llista[]):** Constructor amb una llista d'objectes per paràmetre.

Alguns dels seus mètodes són:

- **public void addItem(Object q):** afegeix un element a la llista (pel final).
- **public insertItemAt(Object q, int indice):** insereix en índex l'element.
- **public void setEditable(boolean flag):** si flag és true l'element de llista seleccionat és editable.
- **public void setMaximumRowCount(int n):** posa el màxim d'elements a mostrar en el combo, si hi ha més elements apareix la barra d'scroll.
- **public Object getSelectedItem():** torna l'element seleccionat.

Els mètodes següents són per la gestió d'events al combobox:

- **public void actionPerformed(ActionEvent ev);**
- **public void addActionListener(ActionListener ae);**
- **public void addItemListener(ItemListener it);**
- **public void addStateChanged(ItemEvent ev);**

Exemple 10. ComboBox

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

class PanelJCombo extends JPanel
{
    private JComboBox jcb;
    public PanelJCombo() {
        jcb = new JComboBox();
        setLayout(new BorderLayout());
        jcb.addItem("MAD - BCN");
        jcb.addItem("MAD - AGP");
        jcb.addItem("MAD - XRY");
        jcb.addItem("BRU - TFN");
        jcb.addItem("LEN - BCN");
        jcb.addItem("ROM - BCN");
        jcb.setMaximumRowCount(4);
        add(jcb,BorderLayout.NORTH);
    }
}
```

Components de text

Els components Swing que s'utilitzen per editar o mostrar text formen una jerarquia de classes la base de la qual és JTextComponent; aquesta última és una classe abstracta que es troba al paquet **java.swing.text**

JTextComponent

Els components de text suporten una ampla varietat de caracters de còdis alfabètics; en ells es pot inserir, esborrar o seleccionar caracters; és text modificable per l'usuari; la classe disposa del constructor **JTextComponent()** que crea una component de text editable.

Els seus mètodes són:

- **String getText():** torna el text que té el component.
- **String getText(int dspl,int lon):** torna el text del component a partir del desplaçament dspl i de la longitud lon.
- **void setText(String txt):**: substituieix el text del component per txt; si txt és null o és una cadena buida, esborra el text del component.
- **void setEditable(boolean b):** un component de text és editable per omisió; amb aquest mètode s'especifica si és o no.

JTextField, JPasswordText

La classe **JTextField** representa un camp de text modificable per l'usuar; amb aquest component s'edita una línia de text amb l'ample, alineació i tipus de lletra que es desitge.

JPasswordField es deriva de JTextField; representa un camp de text amb la particularitat de que emmascara els caracters quan es visualitza; s'utilitza per editar una clau secreta, contrasenya o password; per omisió, cada caracter de un JPasswordField es substitueix per un '*'.

Els constructors de **JTextField** són:

- **JTextField():** camp de text buit de 0 columnes.
- **JTextField(int cols):** camp de text buit de cols columnes.
- **JTextField(String msg):** camp de text ajustat a la cadena msg.
- **JTextField(String msg,int col):** camp de text amb la cadena msg i de cols columnes.

Els constructors de **JPasswordField** tenen els mateixos arguments que JTextField; els seus mètodes són:

- **void setFont(Font tipo)**
- **void setHorizontalAlignment(int alig):** alineació del text; els valors possibles d'alig son les constants de **SwingConstants**: RIGHT, LEFT, CENTER, TRAILING, LEADING (aquest és el predeterminat)
- **void setColumns(int cols):** posa el número de columnes preferit per al camp.

JPasswordField hereta els mètodes anteriors i a més a més disposa d'aquestos:

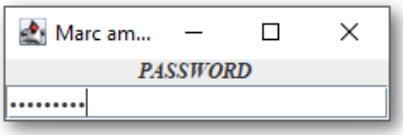
- **void setEchoChar(char c):** coloca c per emmascarar els caracters del camp.
- **char getEchoChar():** torna el caracter que emmascara; per defecte '*'.
- **char[] getPassword():** torna la cadena del camp en un array de caracters.

Exemple 11. Passwords i Labels

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

public class MarcPassWord extends JFrame {
    private static final int AMPLE=300, ALT=150;
    private JPasswordField clau = null;
    private JLabel et1 = null;
    private JLabel res = null;

    public MarcPassWord(String c) {
        super(c);
        setSize(ANCHO,ALTO);
        creaComponentes();
        pack();
    }

    private void creaComponents() {
        clau = new JPasswordField(20);
        et1= new JLabel();

        // oyente para proceso de la acción del usuario
        clave.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                procesoAccionUser(evt);
            }
        });

        // pone el campo de texto con la clave
        add(clau, BorderLayout.CENTER);
```

```
        // crea y pone la etiqueta en el marco
        et1.setFont(new java.awt.Font("Times New Roman", 3, 12));
        et1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        et1.setText("PASSWORD ");
        et1.setToolTipText("Exemple");
        add(et1,BorderLayout.NORTH);

        // método que se ejecuta al actuar el usuario sobre el campo
        private void procesoAccionUser(ActionEvent evt) {
            char pas[];
            pas = clau.getPassword();
            res= new JLabel(" ");
            res.setFont(new Font("Book Antiqua", 3, 14));
            if (pas.length == 0) {
                System.out.println("Teclejar Password ");
                et1.setText("PASSWORD(teclejar) ");
            }
            else {
                clau.setEditable(false);
                res.setText("Es valida la clau");
            }
            add(res, BorderLayout.SOUTH);
            validate();
            pack();
        }

        public static void main(String args[]) {
            MarcPassWord marc;
            marc = new MarcPassWord("Marc amb password");
            marc.setVisible(true);
            marc.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
    }
}
```

Resultat



JTextArea

El component **JTextArea** s'utilitza amb la finalitat de mostrar moltes línies de text; disposa de mètodes per fixar l'ample de cada línia i l'acció a realitzar si la línia que s'insereix és major que l'ample prefixat; també permet decidir si es trenquen o no les paraules al canvi de línia.

Aquest component no disposa de barra d'scroll **JScrollPane**, cal crear l'scroll i associar-lo al component; per exemple: es crea el component areaText:

```
JTextArea areaText = new JTextArea();
// A continuació es crea l'scroll i s'associa a **areaText**:
JScrollPane barra = new JScrollPane(areaText);
```

```
// Per últim, l'scroll es posa al marc
areaText.add(barra);
```

Els constructors són:

- **JTextArea():** crea el component amb cadena nula, zero files i columnes.
- **JTextArea(int filas,int cols):** crea amb cadena nula i el número de files i columnes especificat.
- **JTextArea(String t):** crea el component amb cadena t i zero files i columnes.
- **JTextArea(String t,int filas, int col):** crea el component amb cadena t i el número de files i columnes especificat.

Els seus mètodes són:

- **public void append(String t):** afegeix la cadena t al final del document.
- **public void insert(String t,int p):** insereix la cadena t a partir de la posició p.
- **void replaceRange(String t, int inici, int fi):** substitueix el text del document al rang inici-fi per la cadena t.
- **public void setColumns(int cols):** fixa l'ample de cada línia.
- **public void setLineWrap(boolean f):** si f és true activa el canvi automàtica de línia.
- **public void setWrapStyleWord(boolean f):** si f és true no 'trenca' les paraules en el canvi de línia.