11.B. Eventos.

1. Eventos.

1.5. Eventos de ratón.

Similarmente a los eventos de teclado, los eventos del ratón se generan como respuesta a que el usuario pulsa o libera un botón del ratón, o lo mueve sobre un componente.

`MouseListener` (oyente de ratón)	
Método	**Causa de la invocación**
`mousePressed (MouseEvent e)`	Se ha pulsado un botón del ratón en un componente.
`mouseReleased (MouseEvent e)`	Se ha liberado un botón del ratón en un componente.
`mouseClickerd (MouseEvent e)`	Se ha pulsado y liberado un botón del ratón sobre un componente.
`mouseEntered (KeyEvent e)`	Se ha entrado (con el puntero del ratón) en un componente.
`mouseExited (KeyEvent e)`	Se ha salido (con el puntero del ratón) de un componente.

`MouseMotionListener` (oyente de ratón)	
Método	**Causa de la invocación**
`mouseDragged (MouseEvent e)`	Se presiona un botón y se arrastra el ratón.
`mouseMoved (MouseEvent` ratón sobre `e)`	Se mueve el puntero del un componente.

`MouseWheelListener` (oyente de ratón)	
Método	**Causa de la invocación**
`MouseWheelMoved (MouseWheelEvent e)`	Se mueve la rueda del ratón.

En el siguiente códigoo podemos ver una demostración de un formulario con dos botones. Implementamos un oyente `MouseListener` y registramos los dos botones para detectar tres de los cinco eventos del interface.

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

public class MiMarcoRaton extends javax.swing.JFrame {

  /** Constructor: crea nuevo marco miMarcoRaton */
  public MiMarcoRaton() {
    initComponents();
```

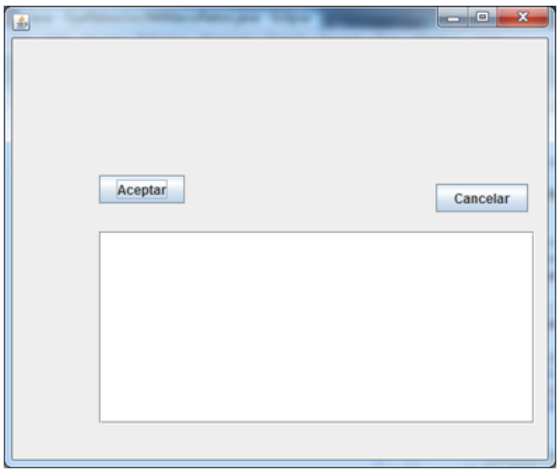
```
    // Crear el objeto oyente de ratón
    OyenteRaton oyenteRat = new OyenteRaton() ;

    // Registrar el oyente en el botón de Aceptar
    jButton1.addMouseListener(oyenteRat);
    // Registrar el oyente en el botón de Cancelar
    jButton2.addMouseListener(oyenteRat);
}

// Implementar la clase oyente que implemente el interface MouseListener
// Se deja en blanco el cuerpo de mouseEntered y de mouseExited, ya que
// no nos interesan en este ejemplo. Cuando se desea escuchar algún
// tipo de evento, de deben implementar todos los métodos del interface
// para que la clase no tenga que ser definida como abstracta
class OyenteRaton implements MouseListener{
    // Gestionar evento de pulsación de cualquier tecla
    public void mousePressed(MouseEvent e) {
        escribir("Botón de ratón pulsado", e) ;
    }
    public void mouseReleased(MouseEvent e) {
        escribir("Botón de ratón liberado", e);
    }
    public void mouseEntered(MouseEvent e) {
    }
    public void mouseExited(MouseEvent e) {
    }
    public void mouseClicked(MouseEvent e) {
        escribir("Click en el botón del ratón", e);
    }

    void escribir(String eventDescription, MouseEvent e) {
        // Escribir en el área de texto la descripción  que
        // se recibe como parámetro
        jTextArea1.append(eventDescription + ".\n");

        // Comprobamos cuál de los dos botones es y lo escribimos
        if (e.getComponent().getName().equals(jButton1.getName()) )
          jTextArea1.append("Es el botón Aceptar.\n");
        else
```

```
        jTextArea1.append("Es el botón Cancelar.\n");
    }

}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
private void initComponents() {

    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jButton1.setText("Aceptar");
    jButton1.setName("Aceptar"); // NOI18N

    jButton2.setText("Cancelar");
    jButton2.setName("Cancelar"); // NOI18N

    jTextArea1.setColumns(20);
    jTextArea1.setRows(5);
    jScrollPane1.setViewportView(jTextArea1);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(80, 80, 80)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addGroup(layout.createSequentialGroup()
```

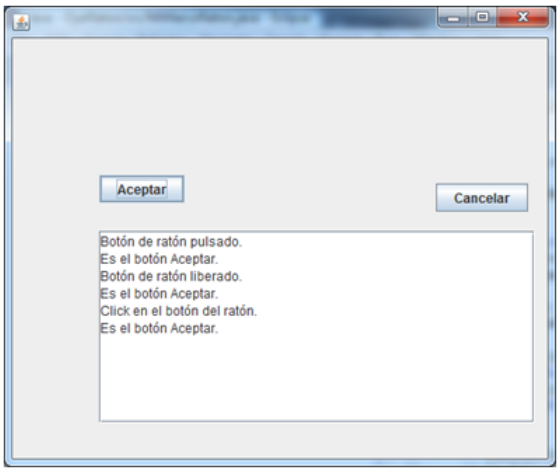
```
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 402, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
            .addGroup(layout.createSequentialGroup()
                .addComponent(jButton1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(jButton2)
                .addGap(18, 18, 18))))
    );
    layout.setVerticalGroup(
      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(layout.createSequentialGroup()
        .addGap(126, 126, 126)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
          .addGroup(layout.createSequentialGroup()
            .addComponent(jButton1)
            .addGap(26, 26, 26))
          .addGroup(layout.createSequentialGroup()
            .addComponent(jButton2)
            .addGap(18, 18, 18)))
        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 177, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(33, Short.MAX_VALUE))
    );

    pack();
  }// </editor-fold>//GEN-END:initComponents

  /**
  * @param args the command line arguments
  */
  public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
        new MiMarcoRaton().setVisible(true);
      }
    });
  }

  // Variables declaration - do not modify//GEN-BEGIN:variables
```

```
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration//GEN-END:variables

}
```

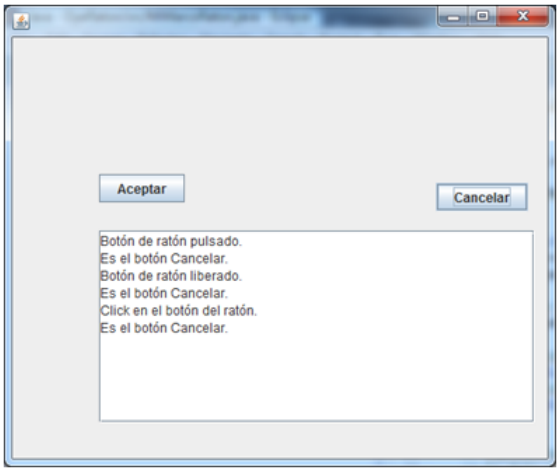
La salida es:



Al clicar sobre el botón Aceptar:



Al clicar sobre el botón Cancelar:



Como se ve en el código, se deja en blanco el cuerpo de `mouseEntered` y de `mouseExited`, ya que no nos interesan en este ejemplo. Cuando se desea escuchar algún tipo de evento, de deben implementar todos los métodos del interface para que la clase no tenga que ser definida como abstracta. Para evitar tener que hacer esto, podemos utilizar adaptadores.

Para saber más

En el enlace que ves a continuación, hay también un ejemplo interesante de la programación de eventos del ratón.

La programación del ratón

Autoevaluación

Cuando el usuario deja de pulsar una tecla se invoca a keyReleased(KeyEvent e)**.**

- ○ Verdadero.
- ○ Falso.