MouseEvent:

An event which indicates that a mouse action occurred in a component. A mouse action is considered to occur in a particular component if and only if the mouse cursor is over the unobscured part of the component's bounds when the action happens. For lightweight components, such as Swing's components, mouse events are only dispatched to the component if the mouse event type has been enabled on the component.

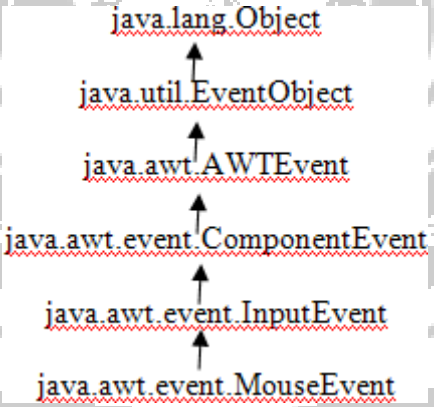
A mouse event type is enabled by adding the appropriate mouse-based EventListener to the component (Mouse Listener or Mouse Motion Listener), or by invoking Component.enableEvents (long) with the appropriate mask parameter

(AWTEvent.MOUSE_EVENT_MASK or AWTEvent.MOUSE_MOTION_EVENT_MASK).

If the mouse event type has not been enabled on the component, the corresponding mouse events are dispatched to the first ancestor that has enabled the mouse event type.Iif a MouseListener has been added to a component, or enableEvents(AWTEvent.MOUSE_EVENT_MASK) has been invoked, then all the events defined by MouseListener are dispatched to the component.

On the other hand, if MouseMotionListener has not been added and enableEvents has not been invoked with AWTEvent.MOUSE_MOTION_EVENT_MASK, then mouse motion events are not dispatched to the component. Instead the mouse motion events are dispatched to the first ancestor that has enabled mouse motion events.

The hierarchy of MouseEvent class is shown below.



- Mouse Events are
  - a mouse button is pressed
  - a mouse button is released
  - a mouse button is clicked (pressed and released)
  - the mouse cursor enters the unobscured part of component's geometry
  - the mouse cursor exits the unobscured part of component's geometry
- Mouse Motion Events are
  - the mouse is moved
  - the mouse is dragged

A MouseEvent object is passed to every MouseListener or MouseAdapter object which is

registered to receive the "interesting" mouse events using the component's addMouseListener method. The MouseAdapter objects implement the MouseListener interface. Each such listener object gets a MouseEvent containing the mouse event.

A MouseEvent object is also passed to every MouseMotionListener or MouseMotionAdapter object which is registered to receive mouse motion events using the component's addMouseMotionListener method. (MouseMotionAdapter objects implement the MouseMotionListener interface.) Each such listener object gets a MouseEvent containing the mouse motion event.

When a mouse button is clicked, events are generated and sent to the registered MouseListeners. The state of modal keys can be retrieved using InputEvent.getModifiers() and InputEvent.getModifiersEx(). The button mask returned by InputEvent.getModifiers() reflects only the button that changed state, not the current state of all buttons.. To get the state of all buttons and modifier keys, use InputEvent.getModifiersEx(). The button which has changed state is returned by getButton().

For example, if the first mouse button is pressed, events are sent in the following order:

id modifiers button

MOUSE_PRESSED: BUTTON1_MASK BUTTON1

MOUSE_RELEASED: BUTTON1_MASK BUTTON1

MOUSE_CLICKED: BUTTON1_MASK BUTTON1

When multiple mouse buttons are pressed, each press, release, and click results in a separate event.

For example, if the user presses button 1 followed by button 2, and then releases them in the same order, the following sequence of events is generated:

id modifiers button

MOUSE_PRESSED: BUTTON1_MASK BUTTON1

MOUSE_PRESSED: BUTTON2_MASK BUTTON2

MOUSE_RELEASED: BUTTON1_MASK BUTTON1

MOUSE_CLICKED: BUTTON1_MASK BUTTON1

MOUSE_RELEASED: BUTTON2_MASK BUTTON2

MOUSE_CLICKED: BUTTON2_MASK BUTTON2

If **button 2** is released first, the MOUSE_RELEASED/MOUSE_CLICKED pair for BUTTON2_MASK arrives first, followed by the pair for BUTTON1_MASK.

MOUSE_DRAGGED events are delivered to the Component in which the mouse button was pressed until the mouse button is released (regardless of whether the mouse position is within the bounds of the Component). Due to platform-dependent Drag&Drop implementations, MOUSE_DRAGGED events may not be delivered during a native Drag&Drop operation.

In a multi-screen environment mouse drag events are delivered to the Component even if the mouse position is outside the bounds of the Graphics Configuration associated with that Component. However, the reported position for mouse drag events in this case may differ from the actual mouse position:

- Inamulti-screenenvironmentwithoutavirtualdevice:Thereportedcoordinatesformouse drag events are clipped to fit within the bounds of the GraphicsConfiguration associated with the Component.
- In a multi-screen environment with a virtual device: The reported coordinates for mouse drag events are clipped to fit within the bounds of the virtual device associated with the Component.

The following program is an example for MouseEvent.

```
import java.awt.*; import
java.awt.event.*;
        pubic class MouseListenerExample extends Frame implements MouseListener{
Label l;
MouseListenerExample(){
  addMouseListener(this);
  l=new Label();
  l.setBounds(20,50,100,20);
  add(l);
  setSize(300,300);
  setLayout(null);
  setVisible(true);
}
public void mouseClicked(MouseEvent e) {
  l.setText("Mouse Clicked");
}
public void mouseEntered(MouseEvent e) {
  l.setText("Mouse Entered");
}
public void mouseExited(MouseEvent e) {
  l.setText("Mouse Exited");
}
public void mousePressed(MouseEvent e) {
  l.setText("Mouse Pressed");
}
public void mouseReleased(MouseEvent e) {
  l.setText("Mouse Released");
}
public static void main(String[] args) {
```

```
    new MouseListenerExample();
  }
  }
```

Output: