Resizable component in Java Swing

In this part of the Java Swing tutorial, we will create a resizable component.

Resizable component

Resizable components are often used when creating charts or diagrams. A common resizable component is a chart in a spreadsheet application. The chart can be moved over a table widget of the application and resized.

In order to create a component that can be freely dragged over a panel, we use a panel with absolute positioning enabled. In our example, we will create a component that we can freely move over a parent window and resize.

Eight small rectangles are drawn on the border of our resizable component when it has the focus. The rectangles serve as dragging points, where we can draw the component and start resizing. The following example is based on an example from this blog.

ResizableComponentEx.java

```
package com.zetcode;

import java.awt.Color;
import java.awt.EventQueue;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class ResizableComponentEx extends JFrame {

    private Resizable res;

    public ResizableComponentEx() {

        initUI();
    }

    private void initUI() {

        JPanel pnl = new JPanel(null);
        add(pnl);

        JPanel area = new JPanel();
        area.setBackground(Color.white);
        res = new Resizable(area);
        res.setBounds(50, 50, 200, 150);
        pnl.add(res);

        addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent me) {

                requestFocus();
                res.repaint();
            }
        });

        setSize(350, 300);
        setTitle("Resizable component");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(new Runnable() {

            @Override
            public void run() {
                ResizableComponentEx ex = new ResizableComponentEx();
                ex.setVisible(true);
            }
        });
    }
}
```

`ResizableComponentEx` sets up the panel and the component.

```
JPanel pnl = new JPanel(null);
```

We use absolute positioning for a resizable component. By providing `null` to the constructor of a `JPanel`, we create a panel with absolute positioning.

```
addMouseListener(new MouseAdapter() {
    @Override
    public void mousePressed(MouseEvent me) {

        requestFocus();
        res.repaint();
    }
});
```

If we press on the parent panel, e.g outside the resizable component, we grab focus and repaint the component. The rectangles over the border will disappear.

ResizableBorder.java

```
package com.zetcode;

import java.awt.Color;
import java.awt.Component;
import java.awt.Cursor;
import java.awt.Graphics;
import java.awt.Insets;
import java.awt.Rectangle;
import java.awt.event.MouseEvent;
import javax.swing.SwingConstants;
import javax.swing.border.Border;

public class ResizableBorder implements Border {

    private int dist = 8;

    int locations[] = {
        SwingConstants.NORTH, SwingConstants.SOUTH, SwingConstants.WEST,
        SwingConstants.EAST, SwingConstants.NORTH_WEST,
        SwingConstants.NORTH_EAST, SwingConstants.SOUTH_WEST,
        SwingConstants.SOUTH_EAST
    };

    int cursors[] = {
        Cursor.N_RESIZE_CURSOR, Cursor.S_RESIZE_CURSOR,
Cursor.W_RESIZE_CURSOR,
        Cursor.E_RESIZE_CURSOR, Cursor.NW_RESIZE_CURSOR,
Cursor.NE_RESIZE_CURSOR,
        Cursor.SW_RESIZE_CURSOR, Cursor.SE_RESIZE_CURSOR
    };

    public ResizableBorder(int dist) {
        this.dist = dist;
    }

    @Override
    public Insets getBorderInsets(Component component) {
        return new Insets(dist, dist, dist, dist);
    }

    @Override
    public boolean isBorderOpaque() {
        return false;
    }

    @Override
    public void paintBorder(Component component, Graphics g, int x, int y,
            int w, int h) {

        g.setColor(Color.black);
        g.drawRect(x + dist / 2, y + dist / 2, w - dist, h - dist);

        if (component.hasFocus()) {

            for (int i = 0; i < locations.length; i++) {
                Rectangle rect = getRectangle(x, y, w, h, locations[i]);
                g.setColor(Color.WHITE);
                g.fillRect(rect.x, rect.y, rect.width - 1, rect.height - 1);
                g.setColor(Color.BLACK);
                g.drawRect(rect.x, rect.y, rect.width - 1, rect.height - 1);
            }
        }
    }

    private Rectangle getRectangle(int x, int y, int w, int h, int location)
{

        switch (location) {
            case SwingConstants.NORTH:
                return new Rectangle(x + w / 2 - dist / 2, y, dist, dist);
            case SwingConstants.SOUTH:
                return new Rectangle(x + w / 2 - dist / 2, y + h - dist,
dist,
                        dist);
            case SwingConstants.WEST:
                return new Rectangle(x, y + h / 2 - dist / 2, dist, dist);
            case SwingConstants.EAST:
                return new Rectangle(x + w - dist, y + h / 2 - dist / 2,
dist,
                        dist);
            case SwingConstants.NORTH_WEST:
                return new Rectangle(x, y, dist, dist);
            case SwingConstants.NORTH_EAST:
                return new Rectangle(x + w - dist, y, dist, dist);
            case SwingConstants.SOUTH_WEST:
                return new Rectangle(x, y + h - dist, dist, dist);
            case SwingConstants.SOUTH_EAST:
                return new Rectangle(x + w - dist, y + h - dist, dist,
dist);
        }
        return null;
    }

    public int getCursor(MouseEvent me) {

        Component c = me.getComponent();
        int w = c.getWidth();
        int h = c.getHeight();

        for (int i = 0; i < locations.length; i++) {
            Rectangle rect = getRectangle(0, 0, w, h, locations[i]);
            if (rect.contains(me.getPoint())) {
                return cursors[i];
            }
        }

        return Cursor.MOVE_CURSOR;
    }
}
```

`ResizableBorder` is responsible for drawing the border of the component and determining the type of the cursor to use.

```
int locations[] = {
    SwingConstants.NORTH, SwingConstants.SOUTH, SwingConstants.WEST,
    SwingConstants.EAST, SwingConstants.NORTH_WEST,
    SwingConstants.NORTH_EAST, SwingConstants.SOUTH_WEST,
    SwingConstants.SOUTH_EAST
};
```

These are locations where the rectangles are drawn. These locations are also grabbing points, where it is possible to grab the component and resize it.

```
g.setColor(Color.black);
g.drawRect(x + dist / 2, y + dist / 2, w - dist, h - dist);
```

In the `paintBorder()` method, we draw the border of the resizable component. The upper code draws the outer border of the component.

```
if (component.hasFocus()) {

    for (int i = 0; i < locations.length; i++) {
        Rectangle rect = getRectangle(x, y, w, h, locations[i]);
        g.setColor(Color.WHITE);
        g.fillRect(rect.x, rect.y, rect.width - 1, rect.height - 1);
        g.setColor(Color.BLACK);
        g.drawRect(rect.x, rect.y, rect.width - 1, rect.height - 1);
    }
}
```

The eight rectangles are drawn only in case that the resizable component currently has focus.

```
private Rectangle getRectangle(int x, int y, int w, int h, int location) {

    switch (location) {
        case SwingConstants.NORTH:
            return new Rectangle(x + w / 2 - dist / 2, y, dist, dist);
        case SwingConstants.SOUTH:
            return new Rectangle(x + w / 2 - dist / 2, y + h - dist, dist,
                    dist);
        ...
}
```

The `getRectangle()` method returns the coordinates of a rectangle.

```
public int getCursor(MouseEvent me) {

    Component c = me.getComponent();
    int w = c.getWidth();
    int h = c.getHeight();

    for (int i = 0; i < locations.length; i++) {
        Rectangle rect = getRectangle(0, 0, w, h, locations[i]);
        if (rect.contains(me.getPoint())) {
            return cursors[i];
        }
    }

    return Cursor.MOVE_CURSOR;
}
```

The `getCursor()` method gets the cursor type for the grab point in question.

Resizable.java

```
package com.zetcode;

import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Cursor;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.event.MouseEvent;
import javax.swing.JComponent;
import javax.swing.event.MouseInputAdapter;
import javax.swing.event.MouseInputListener;


public class Resizable extends JComponent {

    public Resizable(Component comp) {
        this(comp, new ResizableBorder(8));
    }

    public Resizable(Component comp, ResizableBorder border) {

        setLayout(new BorderLayout());
        add(comp);
        addMouseListener(resizeListener);
        addMouseMotionListener(resizeListener);
        setBorder(border);
    }

    private void resize() {
        if (getParent() != null) {
            ((JComponent) getParent()).revalidate();
        }
    }

    MouseInputListener resizeListener = new MouseInputAdapter() {

        @Override
        public void mouseMoved(MouseEvent me) {
            if (hasFocus()) {
                ResizableBorder border = (ResizableBorder) getBorder();
                setCursor(Cursor.getPredefinedCursor(border.getCursor(me)));
            }
        }

        @Override
        public void mouseExited(MouseEvent mouseEvent) {
            setCursor(Cursor.getDefaultCursor());
        }

        private int cursor;
        private Point startPos = null;

        @Override
        public void mousePressed(MouseEvent me) {

            ResizableBorder border = (ResizableBorder) getBorder();
            cursor = border.getCursor(me);
            startPos = me.getPoint();
            requestFocus();
            repaint();
        }

        @Override
        public void mouseDragged(MouseEvent me) {

            if (startPos != null) {

                int x = getX();
                int y = getY();
                int w = getWidth();
                int h = getHeight();

                int dx = me.getX() - startPos.x;
                int dy = me.getY() - startPos.y;

                switch (cursor) {
                    case Cursor.N_RESIZE_CURSOR:
                        if (!(h - dy < 50)) {
                            setBounds(x, y + dy, w, h - dy);
                            resize();
                        }
                        break;

                    case Cursor.S_RESIZE_CURSOR:
                        if (!(h + dy < 50)) {
                            setBounds(x, y, w, h + dy);
                            startPos = me.getPoint();
                            resize();
                        }
                        break;

                    case Cursor.W_RESIZE_CURSOR:
                        if (!(w - dx < 50)) {
                            setBounds(x + dx, y, w - dx, h);
                            resize();
                        }
                        break;

                    case Cursor.E_RESIZE_CURSOR:
                        if (!(w + dx < 50)) {
                            setBounds(x, y, w + dx, h);
                            startPos = me.getPoint();
                            resize();
                        }
                        break;

                    case Cursor.NW_RESIZE_CURSOR:
                        if (!(w - dx < 50) && !(h - dy < 50)) {
                            setBounds(x + dx, y + dy, w - dx, h - dy);
                            resize();
                        }
                        break;

                    case Cursor.NE_RESIZE_CURSOR:
                        if (!(w + dx < 50) && !(h - dy < 50)) {
                            setBounds(x, y + dy, w + dx, h - dy);
                            startPos = new Point(me.getX(), startPos.y);
                            resize();
                        }
                        break;

                    case Cursor.SW_RESIZE_CURSOR:
                        if (!(w - dx < 50) && !(h + dy < 50)) {
                            setBounds(x + dx, y, w - dx, h + dy);
                            startPos = new Point(startPos.x, me.getY());
                            resize();
                        }
                        break;

                    case Cursor.SE_RESIZE_CURSOR:
                        if (!(w + dx < 50) && !(h + dy < 50)) {
                            setBounds(x, y, w + dx, h + dy);
                            startPos = me.getPoint();
                            resize();
                        }
                        break;

                    case Cursor.MOVE_CURSOR:
                        Rectangle bounds = getBounds();
                        bounds.translate(dx, dy);
                        setBounds(bounds);
                        resize();
                }

                setCursor(Cursor.getPredefinedCursor(cursor));
            }
        }

        @Override
        public void mouseReleased(MouseEvent mouseEvent) {
            startPos = null;
        }
    };
}
```

The `Resizable` class represents the component that is being resized and moved on the window.

```
private void resize() {
    if (getParent() != null) {
      ((JComponent)getParent()).revalidate();
    }
}
```

The `resize()` method is called, after we have resized the component. The `revalidate()` method will cause the component to be redrawn.

```
MouseInputListener resizeListener = new MouseInputAdapter() {

    @Override
    public void mouseMoved(MouseEvent me) {
        if (hasFocus()) {
            ResizableBorder border = (ResizableBorder) getBorder();
            setCursor(Cursor.getPredefinedCursor(border.getCursor(me)));
        }
    }
    ...
    }
```

We change the cursor type, when we hover the cursor over the grab points. The cursor type changes only if the component has focus.

```
@Override
public void mousePressed(MouseEvent me) {

    ResizableBorder border = (ResizableBorder) getBorder();
    cursor = border.getCursor(me);
    startPos = me.getPoint();
    requestFocus();
    repaint();
}
```

If we click on the resizable component, we change the cursor, get the starting point of dragging, give focus to the component, and redraw it.

```
int x = getX();
int y = getY();
int w = getWidth();
int h = getHeight();

int dx = me.getX() - startPos.x;
int dy = me.getY() - startPos.y;
```

In the `mouseDragged()` method, we determine the x and y coordinates of the cursor and width and height of the component. We calculate the distances that we make during the mouse drag event.

```
case Cursor.N_RESIZE_CURSOR:
    if (!(h - dy < 50)) {
        setBounds(x, y + dy, w, h - dy);
        resize();
    }
    break;
```

For all resizing we ensure that the component is not smaller than 50 px. Otherwise, we could make it so small that we would eventually hide the component. The `setBounds()` method relocates and resizes the component.

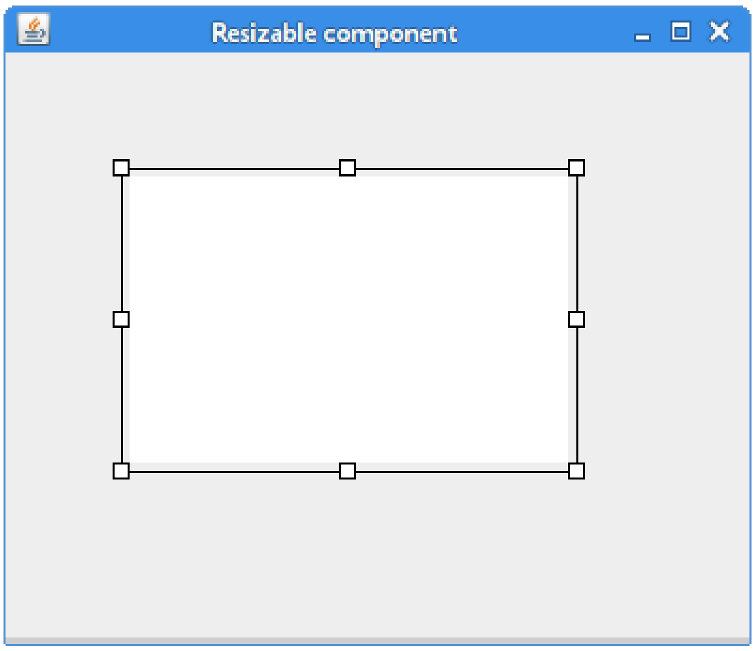

Figure: Resizable component

In this part of the Java Swing tutorial, we have created a resizable component.