|  |
| --- |
| Code:  import javax.swing.event.\*;  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  import javax.swing.border.\*;  public class GUI implements ActionListener, MouseListener, MouseMotionListener  {  JFrame f;    int[] pointA, pointB;    boolean lineDraw, change;  String current;    JPanel mainPanel, paintPanel, btnPanel;  JButton btnLine, btnRect;    public GUI()  {  f = new JFrame("PAINT");    lineDraw = true;  change = false;  current = "line";    pointA = new int[2];  pointB = new int[2];    btnPanel = new JPanel(new FlowLayout());    btnLine = new JButton("LINE");  btnRect = new JButton("RECTANGLE");    mainPanel = new JPanel(new BorderLayout());  mainPanel.setPreferredSize(new Dimension(400, 400));  mainPanel.setBorder(new EtchedBorder(EtchedBorder.LOWERED));    PaintClass paint = new PaintClass();  paint.setBorder(new SoftBevelBorder(SoftBevelBorder.LOWERED));    paint.addMouseListener(this);  paint.addMouseMotionListener(this);    btnLine.addActionListener(this);  btnRect.addActionListener(this);    btnPanel.add(btnLine);  btnPanel.add(btnRect);    mainPanel.add(paint, BorderLayout.CENTER);  mainPanel.add(btnPanel, BorderLayout.SOUTH);    f.add(mainPanel);    f.pack();  f.setLocationRelativeTo(null);  f.setVisible(true);    }    /\*\*  \* Method Name : pointCoordinate  \* @param int[] coordinate - store the x and y mouse coordinates  \* @param int x - x coordinate  \* @param int y - y coordinate  \* @return void  \*/  public static void pointCoordinate(int[] coordinate, int x, int y)  {  coordinate[0] = x;  coordinate[1] = y;  }    public void actionPerformed(ActionEvent e)  {  //Change the drawing  change = true;    //Draw a line on the screen  if(e.getSource() == btnLine)  {  lineDraw = true;    //Do not change the drawing if the current painting button is selected  if(!current.equals("line"))  change = false;  }    //Draw a rectangle on the screen  else  {  lineDraw = false;    //Do not change the drawing if the current painting button is selected  if(!current.equals("rect"))  change = false;  }  }    /\*\*  \* MouseListener and MouseMotionListeners that are not used  \*/  public void mouseClicked(MouseEvent e){}  public void mouseEntered(MouseEvent e){}  public void mouseExited(MouseEvent e){}  public void mouseReleased(MouseEvent e){}  public void mouseMoved(MouseEvent e){}    /\*\*  \* Purpose : Get the starting and final x and y coordinates of  \* the mouse when the mouse button is pressed  \*/  public void mousePressed(MouseEvent e)  {  pointCoordinate(pointA, e.getX(), e.getY());  pointCoordinate(pointB, e.getX(), e.getY());  }    /\*\*  \* Purpose : Get the final mouse coordinate with the mouse is  \* dragged while pressing  \*/  public void mouseDragged(MouseEvent e)  {  //Store the x and y coordinates of the mouse in the pointB array  pointCoordinate(pointB, e.getX(), e.getY());    //If line button is selected, change the string to "line"  if(lineDraw)  current = "line";    //If rectangle button is selected, change the string to "rect"  else  current = "rect";    //Change the drawing on the screen  change = true;  }    //Main  public static void main(String[] args)  {  new GUI();  }    //PAINT CLASS -- INTERNAL CLASS --> INHERITANCE  class PaintClass extends JPanel  {    //Mouse initial and final coordinates  int x1, y1, x2, y2;    public PaintClass()  {}    public void paintComponent(Graphics g)  {  super.paintComponent(g);    //Set the background to white color  g.setColor(Color.WHITE);  g.fillRect(0, 0, getWidth(), getHeight());    g.setColor(Color.BLACK);    //Get the Starting point  x1 = pointA[0];  y1 = pointA[1];    //Get the dragging point  x2 = pointB[0];  y2 = pointB[1];    //Draw the Line  if(lineDraw)  {  //if the user has started drawing the line  if(current.equals("line") && change)  {  line(g, x1, x2, y1, y2);  }    //if only the line drawing option is selected  else  rectangle(g, x1, x2, y1, y2);  }    //Draw the Rectangle  else  {  //If the user has started drawing the rectangle  if(current.equals("rect") && change)  {  rectangle(g, x1, x2, y1, y2);  }    //If only the rect drwaing option is selected  else  line(g, x1, x2, y1, y2);  }    repaint();    }    //Draw the line  public void line(Graphics g, int x1, int x2, int y1, int y2)  {  g.setColor(Color.BLUE);  g.drawLine(x1, y1, x2, y2);  }    //Draw the rectangle  public void rectangle(Graphics g, int x1, int x2, int y1, int y2)  {  g.setColor(Color.RED);    //Bottom Right  g.fillRect(x1, y1, x2 - x1, y2 - y1);    //Bottom Left  g.fillRect(x2, y1, x1 - x2, y2 - y1);    //Top Left  g.fillRect(x2, y2, x1 - x2, y1 - y2);    //Top Right  g.fillRect(x1, y2, x2 - x1, y1 - y2);  }    }    } |
| Screenshots: |