Ex4 - La Grande Finale

Ben Mishali

305304867

DrawJFrame

*/\*\*  
 \* DrawJFrame.java  
 \* {****@link*** *JFrame} uses J-Objects -> {****@link*** *JButton},{****@link*** *JComboBox},{****@link*** *JCheckBox} and {****@link*** *JLabel} with {****@link*** *JPanel} to set the GUI.  
 \* also uses {****@link*** *Color} to set array of colors, {****@link*** *string} to set array of Colors names and Shapes names  
 \** ***@extends*** *{****@link*** *JFrame}.  
 \** ***@author*** *Ben Mishali  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
  
public class DrawFrame extends JFrame {  
 private JButton undoButton;  
 private JButton clearButton;  
 private JComboBox<String> colorChoices;  
 private JComboBox<String> shapeChoices;  
 private JCheckBox filledCheckBox;  
 private JLabel statusLabel=new JLabel("(" + 0 + "," + 0 + ")");  
 private JPanel upperMenu;  
 private Color[] myColors = new Color[13];  
 private static final String *shapesNames*[] = {"Line","Oval","Rectangle"};  
 private String[] colorsName = new String[13];  
 private DrawPanel panel=new DrawPanel(statusLabel,this);  
  
 */\*\*  
 \*constructor initializes the GUI with setGUI function  
 \*/* public DrawFrame() {  
 setGUI();  
 }  
 */\*\*  
 \*initializes our menus, buttons, panel & labeltext  
 \*/* public void setGUI() {  
 initMenus();  
 initButtons();  
 colorsName= initColors();  
 statusLabel.setOpaque(true);  
 upperMenu.add(undoButton);  
 upperMenu.add(clearButton);  
 upperMenu.add(colorChoices);  
 upperMenu.add(shapeChoices);  
 upperMenu.add(filledCheckBox);  
 this.add(upperMenu,BorderLayout.*NORTH*);  
 this.add(statusLabel,BorderLayout.*SOUTH*);  
 this.add(panel,BorderLayout.*CENTER*);  
 panel.setBackground(Color.*WHITE*);  
 }  
 */\*\*  
 \*initializes our menus like buttons and filled Check Box  
 \*/* public void initMenus()  
 {  
 upperMenu = new JPanel();  
 filledCheckBox = new JCheckBox("filled");  
 undoButton = new JButton("Undo");  
 clearButton = new JButton("Clear");  
 initComboCheck();  
 }  
 */\*\*  
 \*initializes our ComboCheck menus - one of colors & one for shapes  
 \*also uses {****@link*** *ItemListener} and {****@link*** *ActionListener} to set {****@link*** *JComboBox} and {****@link*** *JCheckBox} events.  
 \*/* public void initComboCheck() {  
 colorChoices = new JComboBox<String>(initColors());  
 shapeChoices = new JComboBox<String>(*shapesNames*);  
 colorChoices.setMaximumRowCount(8);  
 colorChoices.addItemListener(  
 new ItemListener() {  
 @Override  
 public void itemStateChanged(ItemEvent event) {  
 if (event.getStateChange() == ItemEvent.*SELECTED*) {  
 panel.setDrawingColor(myColors[colorChoices.getSelectedIndex()]);  
 }  
  
 }  
 } // end anonymous inner class  
 ); // end call to addItemListener  
 shapeChoices.addItemListener(  
 new ItemListener() {  
 @Override  
 public void itemStateChanged(ItemEvent event) {  
 if (event.getStateChange() == ItemEvent.*SELECTED*) {  
 panel.setDrawingShape(*shapesNames*[shapeChoices.getSelectedIndex()]);  
 }  
 }  
 } // end anonymous inner class  
 ); // end call to addItemListener  
 filledCheckBox.addItemListener(  
 new ItemListener() {  
 @Override  
 public void itemStateChanged(ItemEvent event) {  
 if (filledCheckBox.isSelected())  
 panel.setFilled(true);  
 else  
 panel.setFilled(false);  
 }  
 }  
 );  
 }  
 */\*\*  
 \*initializes our Buttons  
 \*also uses {****@link*** *ActionListener} to set {****@link*** *JButton} events.  
 \*/* public void initButtons() {  
 undoButton.addActionListener(  
 new ActionListener()  
 {  
 @Override  
 public void actionPerformed(ActionEvent event)  
 {  
 if(panel.getIndex()!=0) {  
 panel.setIndex(panel.getIndex() - 1);  
 panel.repaint();  
 }  
 }  
 }  
 );  
 clearButton.addActionListener(  
 new ActionListener()  
 {  
 @Override  
 public void actionPerformed(ActionEvent event)  
 {  
 panel.setIndex(0);  
 panel.setCurrentShape(null);  
 panel.repaint();  
 }  
 }  
 );  
  
  
 }  
  
 */\*\*  
 \*indexing our arrays of colors names and Colors Objects  
 \*also uses {****@link*** *ActionListener} to set {****@link*** *JButton} events.  
 \*/* private final String[] initColors() {  
 String names[] = {"Black","Blue","Cyan","Dark Gray","Gray","Green","Light Gray","Magenta","Orange","Pink","Red","White","Yellow"};  
 myColors[0]= new Color(0,0,0); //Black  
 myColors[1]= new Color(0,0,255); //Blue  
 myColors[2]= new Color(0,255,255); //Cyan  
 myColors[3]= new Color(169,169,169); //Dark Gray  
 myColors[4]= new Color(128,128,128); //Gray  
 myColors[5]= new Color(0,128,0); //Green  
 myColors[6]= new Color(211,211,211); //Light Gray  
 myColors[7]= new Color(255,0, 255); //Magenta  
 myColors[8]= new Color(255, 165,0); //Orange  
 myColors[9]= new Color(255, 192, 203); //Pink  
 myColors[10]= new Color(255,0,0); //Red  
 myColors[11]= new Color(255, 255, 255); //White  
 myColors[12]= new Color(255, 255,0); //Yellow  
 return names;  
 }  
}

DrawPanel

*/\*\*  
 \* DrawPanel.java  
 \* Program that uses classes MyLine, MyOval and MyRectangle to draw random shapes.  
 \** ***@author*** *Ben Mishali  
 \** ***@extends*** *Jpanel  
 \*/*import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.awt.event.MouseListener;  
  
import static javax.swing.SwingUtilities.*isLeftMouseButton*;  
  
public class DrawPanel extends JPanel {  
 private MyShape shapes[] = new MyShape[*MAXSHAPES*]; // array of Shapes  
 private int x1, y1, x2, y2, x1\_moved, y1\_moved, index;  
 private Color currentColor;  
 private String currentType;  
 private boolean currentFilled;  
 private MyShape currentShape;  
 private JLabel statusLabelTemp;  
 private JFrame tempFrameThis;  
 private static final int *MAXSHAPES* =100;  
  
  
  
  
 public DrawPanel(JLabel statusLabel, JFrame frameThis) {  
 currentFilled = false;  
 currentType = "Line";  
 currentColor = Color.*black*;  
 currentShape = new MyLine(x1, y1, x2, y2, currentColor);  
 this.addMouseListener(myMouseAdapter);  
 this.addMouseMotionListener(myMouseAdapter);  
 statusLabelTemp = statusLabel;  
 tempFrameThis = frameThis;  
  
 }  
  
  
 public void addShapeToArr() {  
 shapes[index] = currentShape;  
 index++;  
 }  
 */\*\*  
 \* The method used for initializes our Mouse Adapter actions  
 \* by overriding this method.  
 \* <p>  
 \* This method adds an anonymous MouseAdapter to  
 \* the returned MouseAdapter.  
 \* also uses {****@link*** *MouseAdapter} to take only the events that we needs like {****@link*** *#addMouseListener(MouseListener)} and {****@link*** *#addShapeToArr()} ...  
 \* to add this Shape to the array of shapes  
 \*/* MouseAdapter myMouseAdapter = new MouseAdapter() {  
 @Override  
 public void mouseDragged(MouseEvent e) {  
 if (*isLeftMouseButton*(e) == true) {  
 x2 = e.getX();  
 y2 = e.getY();  
  
 statusLabelTemp.setText("(" + x2 + "," + y2 + ")");  
 switch (currentType) {  
 case "Oval": //Oval Type  
 currentShape = new MyOval(x1, y1, x2, y2, currentColor, currentFilled);  
  
 break;  
 case "Rectangle": // Rect Type  
 currentShape = new MyRectangle(x1, y1, x2, y2, currentColor, currentFilled);  
  
 break;  
 case "Line": // Line Type  
 currentShape = new MyLine(x1, y1, x2, y2, currentColor);  
  
 break;  
 }  
 repaint();  
 }  
 }  
 @Override  
 public void mouseMoved(MouseEvent e) {  
 x1\_moved = e.getX();  
 y1\_moved = e.getY();  
 statusLabelTemp.setText("(" + x1\_moved + "," + y1\_moved + ")");  
  
 }  
  
 @Override  
 public void mouseReleased(MouseEvent e) {  
 try {//cannot be happened but maybe for the future  
 if (*isLeftMouseButton*(e) == true && x1 != e.getX() && y1 != e.getY()) {  
 addShapeToArr();  
 currentShape = null;  
 repaint();  
 }  
 }  
 catch (ArrayIndexOutOfBoundsException ex) {  
 index--;  
 repaint();  
 }  
 }  
  
 @Override  
 public void mousePressed(MouseEvent e) {  
 //If left button pressed  
 if (*isLeftMouseButton*(e) == true) {  
 if(index<*MAXSHAPES*) {  
 x1 = e.getX();  
 y1 = e.getY();  
 }  
 else {  
 JOptionPane.*showMessageDialog*(tempFrameThis,String.*format*("Cannot exceed the current number of shapes: %d",index),"Capacity Overload",JOptionPane.*WARNING\_MESSAGE*);  
 }  
 }  
  
 }  
 };  
 */\*\*  
 \*set the Index of the array of shapes that needs to draw.  
 \** ***@param*** *index  
 \*/* public void setIndex(int index) {  
 this.index = index;  
 }  
 */\*\*  
 \*Get the Index of the array of shapes that needs to draw.  
 \*/* public int getIndex() {  
 return this.index;  
 }  
 */\*\*  
 \*set the Drawing Shape that needs to draw.  
 \** ***@param*** *shape  
 \*/* public void setDrawingShape(String shape) {  
 this.currentType = shape;  
 }  
 */\*\*  
 \*set the Drawing Color for the shape that needs to draw.  
 \** ***@param*** *color  
 \*/* public void setDrawingColor(Color color) {  
 this.currentColor = color;  
 }  
 */\*\*  
 \*set the Filled or not for the shape that needs to draw.  
 \** ***@param*** *filled  
 \*/* public void setFilled(boolean filled) {  
 this.currentFilled = filled;  
 }  
 */\*\*  
 \*set the Current Shape that needs to draw.  
 \** ***@param*** *currentShape  
 \*/* public void setCurrentShape(MyShape currentShape) {  
 this.currentShape = currentShape;  
 }  
  
 */\*\*  
 \*uses the override of the paintComponent function that uses Graphics to draw on our JPanel  
 \** ***@param*** *g  
 \*/* @Override  
 public void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 for (int i=0;i<index;i++)  
 if (shapes[i] != null) {  
 shapes[i].draw(g);  
 }  
 if(currentShape!=null)  
 currentShape.draw(g);  
 } // end method paintComponent  
 } // end class DrawPanel

TestDraw

import javax.swing.\*;  
import java.awt.\*;  
  
*/\*\*  
 \* TestDraw.java  
 \* Program that test our program and set the settings of the {****@link*** *JFrame} - also hold the main function  
 \** ***@author*** *Ben Mishali  
 \*/*public class TestDraw  
{  
 public static void main(String[] args)  
 {  
 DrawFrame frame = new DrawFrame();

frame.setTitle("Shape Drawings");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setSize(500, 500);  
 frame.getContentPane().setBackground(Color.*white*);  
 frame.setVisible(true);  
 }// end main  
} // end class TestDraw





