Solutions for Assignment Unit 6

```
package textcollage;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Scanner;

import javax.imageio.ImageIO;
import javax.swing.BorderFactory;
import javax.swing.JColorChooser;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.KeyStroke;

/**
 * A panel that contains a large drawing area where strings
 * can be drawn.  The strings are represented by objects of
 * type DrawTextItem.  An input box under the panel allows
 * the user to specify what string will be drawn when the
 * user clicks on the drawing area.
 *
 * NEW FEATURES:
 * 1. added support for right click to undo (remove item)
 * 2. added support for undo as many levels as allowed
 * 3. each left click puts text with random background color, border, font, etc.
 * 4. save and open command supports all new features
 */
/**
 * @author Anonymous For assessment purpose
 *
 */
public class DrawTextPanel extends JPanel  {

 // As it now stands, this class can only show one string at at
 // a time!  The data for that string is in the DrawTextItem object
 // named theString.  (If it's null, nothing is shown.  This
 // variable should be replaced by a variable of type
 // ArrayList<DrawStringItem> that can store multiple items.

 private ArrayList<DrawTextItem> theStrings;  // changed to an ArrayList<DrawTextItem> !
```

```
private Color currentTextColor = Color.BLACK;  // Color applied to new strings.

private Canvas canvas;  // the drawing area.
private JTextField input;  // where the user inputs the string that will be added to the canvas
private SimpleFileChooser fileChooser;  // for letting the user select files
private JMenuBar menuBar; // a menu bar with command that affect this panel
private MenuHandler menuHandler; // a listener that responds whenever the user selects a menu command
private JMenuItem undoMenuItem;  // the "Remove Item" command from the edit menu


/**
 * An object of type Canvas is used for the drawing area.
 * The canvas simply displays all the DrawTextItems that
 * are stored in the ArrayList, strings.
 */
private class Canvas extends JPanel {
 Canvas() {
  setPreferredSize( new Dimension(800,600) );
  setBackground(Color.WHITE);
  setFont( new Font( "Serif", Font.BOLD, 24 ));
 }
 protected void paintComponent(Graphics g) {
  super.paintComponent(g);
  ((Graphics2D)g).setRenderingHint(RenderingHints.KEY_ANTIALIASING,
    RenderingHints.VALUE_ANTIALIAS_ON);
  if (theStrings != null)
   for (DrawTextItem s: theStrings)
    s.draw(g);
 }
}

/**
 * An object of type MenuHandler is registered as the ActionListener
 * for all the commands in the menu bar.  The MenuHandler object
 * simply calls doMenuCommand() when the user selects a command
 * from the menu.
 */
private class MenuHandler implements ActionListener {
 public void actionPerformed(ActionEvent evt) {
  doMenuCommand( evt.getActionCommand());
 }
}

/**
 * Creates a DrawTextPanel.  The panel has a large drawing area and
 * a text input box where the user can specify a string.  When the
 * user clicks the drawing area, the string is added to the drawing
 * area at the point where the user clicked.
 */
public DrawTextPanel() {
 fileChooser = new SimpleFileChooser();
 undoMenuItem = new JMenuItem("Remove Item");
 undoMenuItem.setEnabled(false);
 menuHandler = new MenuHandler();
 setLayout(new BorderLayout(3,3));
 setBackground(Color.BLACK);
 setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
 canvas = new Canvas();
 add(canvas, BorderLayout.CENTER);
 JPanel bottom = new JPanel();
 bottom.add(new JLabel("Text to add: "));
 input = new JTextField("Hello World!", 40);
 bottom.add(input);
 add(bottom, BorderLayout.SOUTH);
 canvas.addMouseListener( new MouseAdapter() {
```

```
  public void mousePressed(MouseEvent e) {
   doMousePress( e );
  }
 } );
}

/**
 * This method is called when the user clicks the drawing area.
 * A new string is added to the drawing area.  The center of
 * the string is at the point where the user clicked.
 * @param e the mouse event that was generated when the user clicked
 */
public void doMousePress( MouseEvent e ) {
 if (e.isMetaDown()) { //right click to remove an item
  removeItem();
  return;
 }
 String text = input.getText().trim();
 if (text.length() == 0) {
  input.setText("Hello World!");
  text = "Hello World!";
 }
 DrawTextItem s = new DrawTextItem( text, e.getX(), e.getY() );
 s.setTextColor(currentTextColor);  // Default is null, meaning default color of the canvas (black).

            //   SOME OTHER OPTIONS THAT CAN BE APPLIED TO TEXT ITEMS:
            //
 int randomChoice = (int)(Math.random()*5);
 int fontStyle;
 switch (randomChoice) {
 case 0: fontStyle = Font.ITALIC; break;
 case 1: fontStyle = Font.BOLD; break;
 default: fontStyle = Font.ITALIC + Font.BOLD;
 }
 s.setFont( new Font( "Serif", fontStyle, (int)(Math.random()*12+8) ));

 //create different types of magnification
 s.setMagnification((int)(Math.random()*4+1));

 //create random border
 if (Math.random() > 0.3)
  s.setBorder(true);

 //create random rotation angle (0 to 360)
 s.setRotationAngle(Math.random()*360);

 //create random text transparency (0 to 1)
 s.setTextTransparency(Math.random()*0.25);

 //create random background color
 if (Math.random() > 0.5)
  s.setBackground(new Color((float)Math.random(), (float)Math.random(), (float)Math.random()));

 //create random background transparency (0 to 1)
 s.setBackgroundTransparency(Math.random()*0.90+0.10);

 if (theStrings == null)
  theStrings = new ArrayList<DrawTextItem>();
 theStrings.add(s);  // Set this string as the ONLY string to be drawn on the canvas!
 undoMenuItem.setEnabled(true);
 canvas.repaint();
}

/**
 * Returns a menu bar containing commands that affect this panel.  The menu
 * bar is meant to appear in the same window that contains this panel.
```

```
 */
public JMenuBar getMenuBar() {
 if (menuBar == null) {
  menuBar = new JMenuBar();

  String commandKey; // for making keyboard accelerators for menu commands
  if (System.getProperty("mrj.version") == null)
   commandKey = "control ";  // command key for non-Mac OS
  else
   commandKey = "meta ";  // command key for Mac OS

  JMenu fileMenu = new JMenu("File");
  menuBar.add(fileMenu);
  JMenuItem saveItem = new JMenuItem("Save...");
  saveItem.setAccelerator(KeyStroke.getKeyStroke(commandKey + "N"));
  saveItem.addActionListener(menuHandler);
  fileMenu.add(saveItem);
  JMenuItem openItem = new JMenuItem("Open...");
  openItem.setAccelerator(KeyStroke.getKeyStroke(commandKey + "O"));
  openItem.addActionListener(menuHandler);
  fileMenu.add(openItem);
  fileMenu.addSeparator();
  JMenuItem saveImageItem = new JMenuItem("Save Image...");
  saveImageItem.addActionListener(menuHandler);
  fileMenu.add(saveImageItem);

  JMenu editMenu = new JMenu("Edit");
  menuBar.add(editMenu);
  undoMenuItem.addActionListener(menuHandler); // undoItem was created in the constructor
  undoMenuItem.setAccelerator(KeyStroke.getKeyStroke(commandKey + "Z"));
  editMenu.add(undoMenuItem);
  editMenu.addSeparator();
  JMenuItem clearItem = new JMenuItem("Clear");
  clearItem.addActionListener(menuHandler);
  editMenu.add(clearItem);

  JMenu optionsMenu = new JMenu("Options");
  menuBar.add(optionsMenu);
  JMenuItem colorItem = new JMenuItem("Set Text Color...");
  colorItem.setAccelerator(KeyStroke.getKeyStroke(commandKey + "T"));
  colorItem.addActionListener(menuHandler);
  optionsMenu.add(colorItem);
  JMenuItem bgColorItem = new JMenuItem("Set Background Color...");
  bgColorItem.addActionListener(menuHandler);
  optionsMenu.add(bgColorItem);

 }
 return menuBar;
}

/**
 * Carry out one of the commands from the menu bar.
 * @param command the text of the menu command.
 */
private void doMenuCommand(String command) {
 if (command.equals("Save...")) { // save all the string info to a file
  saveFile();
 }
 else if (command.equals("Open...")) { // read a previously saved file, and reconstruct the list of strings
  openFile();
  canvas.repaint(); // (you'll need this to make the new list of strings take effect)
 }
 else if (command.equals("Clear")) {  // remove all strings
  theStrings = null;   // Remove the ONLY string from the canvas.
  undoMenuItem.setEnabled(false);
  canvas.repaint();
```

```
 }
 else if (command.equals("Remove Item"))
  removeItem();
 else if (command.equals("Set Text Color...")) {
  Color c = JColorChooser.showDialog(this, "Select Text Color", currentTextColor);
  if (c != null)
   currentTextColor = c;
 }
 else if (command.equals("Set Background Color...")) {
  Color c = JColorChooser.showDialog(this, "Select Background Color", canvas.getBackground());
  if (c != null) {
   canvas.setBackground(c);
   canvas.repaint();
  }
 }
 else if (command.equals("Save Image...")) {  // save a PNG image of the drawing area
  File imageFile = fileChooser.getOutputFile(this, "Select Image File Name", "textimage.png");
  if (imageFile == null)
   return;
  try {
   // Because the image is not available, I will make a new BufferedImage and
   // draw the same data to the BufferedImage as is shown in the panel.
   // A BufferedImage is an image that is stored in memory, not on the screen.
   // There is a convenient method for writing a BufferedImage to a file.
   BufferedImage image = new BufferedImage(canvas.getWidth(),canvas.getHeight(),
     BufferedImage.TYPE_INT_RGB);
   Graphics g = image.getGraphics();
   g.setFont(canvas.getFont());
   canvas.paintComponent(g);  // draws the canvas onto the BufferedImage, not the screen!
   boolean ok = ImageIO.write(image, "PNG", imageFile); // write to the file
   if (ok == false)
    throw new Exception("PNG format not supported (this shouldn't happen!).");
  }
  catch (Exception e) {
   JOptionPane.showMessageDialog(this,
     "Sorry, an error occurred while trying to save the image:\n" + e);
  }
 }
}

/**
 * When Command equal "Remove Item" remove the last item from the canvas one by one. Ctrl-Z and right click
 * are both supported.
 */
private void removeItem() {
 if (theStrings.size() > 0)
  theStrings.remove(theStrings.size()-1); // remove the most recently added string
 if (theStrings.size() == 0)

              undoMenuItem.setEnabled(false);
 canvas.repaint();
}

/**
 * Save the current canvas into a text file
 */
private void saveFile() {
 File saveAs = fileChooser.getOutputFile(this, "Save As", "Text Collage.txt");
 try {
  PrintWriter out = new PrintWriter(saveAs);

  out.println("New text collage file");
  out.println(canvas.getBackground().getRed());
  out.println(canvas.getBackground().getGreen());
  out.println(canvas.getBackground().getBlue());
```

```
  if (theStrings != null)
   for (DrawTextItem s: theStrings) {
    out.println("theString:");
    out.println(s.getString());
    out.println(s.getX());
    out.println(s.getY());
    out.println(s.getFont().getName());
    out.println(s.getFont().getStyle());
    out.println(s.getFont().getSize());
    out.println(s.getTextColor().getRed());
    out.println(s.getTextColor().getGreen());
    out.println(s.getTextColor().getBlue());
    out.println(s.getTextTransparency());
    if (s.getBackground() == null) {
     out.println("-1");
     out.println("-1");
     out.println("-1");
    }
    else {
     out.println(s.getBackground().getRed());
     out.println(s.getBackground().getGreen());
     out.println(s.getBackground().getBlue());
    }
    out.println(s.getBackgroundTransparency());
    out.println(s.getBorder());
    out.println(s.getMagnification());
    out.println(s.getRotationAngle());
   }
  out.close();
 } catch (FileNotFoundException e) {
  JOptionPane.showMessageDialog(this, "Can't write to the file \"" + saveAs + "\".");
  System.out.println("Error message: " + e);
 }
}

/**
 * Open a saved text file and read the background color as well as the text
 * strings.
 */
private void openFile() {
 File openFile = fileChooser.getInputFile(this, "Open Saved File");
 try {
  Scanner in = new Scanner(openFile);
  if (!in.nextLine().equals("New text collage file")) {
   JOptionPane.showMessageDialog(this, "Not a valid file \"" + openFile + "\".");
   return;
  }
  Color savedBg = new Color(in.nextInt(), in.nextInt(), in.nextInt());
  ArrayList<DrawTextItem> newStrings = new ArrayList<DrawTextItem>();
  DrawTextItem newItem;
  in.nextLine();  //skip to the next line before read a new line
  while (in.hasNext() && in.nextLine().equals("theString:")) {
   newItem = new DrawTextItem(in.nextLine(),
     in.nextInt(), in.nextInt());
   in.nextLine();  //skip to the next line before read a new line
   newItem.setFont(new Font(in.nextLine(), in.nextInt(), in.nextInt()));
   newItem.setTextColor(new Color(in.nextInt(), in.nextInt(), in.nextInt()));
   newItem.setTextTransparency(in.nextDouble());
   int r = in.nextInt();
   int g = in.nextInt();
   int b = in.nextInt();
   if (r == -1)
    newItem.setBackground(null);
   else
    newItem.setBackground(new Color(r, g, b));
```

```
    newItem.setBackgroundTransparency(in.nextDouble());
    newItem.setBorder(in.nextBoolean());
    newItem.setMagnification(in.nextDouble());
    newItem.setRotationAngle(in.nextDouble());
    in.nextLine();  //skip to the next line before read a new line
    newStrings.add(newItem);
   }
   //if no exception occurred, replace the current background and strings
   canvas.setBackground(savedBg);
   theStrings = newStrings;
  } catch (FileNotFoundException e) {
   JOptionPane.showMessageDialog(this, "Can't read the file \"" + openFile + "\".");
   System.out.println("Error message: " + e);
  }

 }
}
```

Last modified: Wednesday, 13 May 2020, 4:14 PM