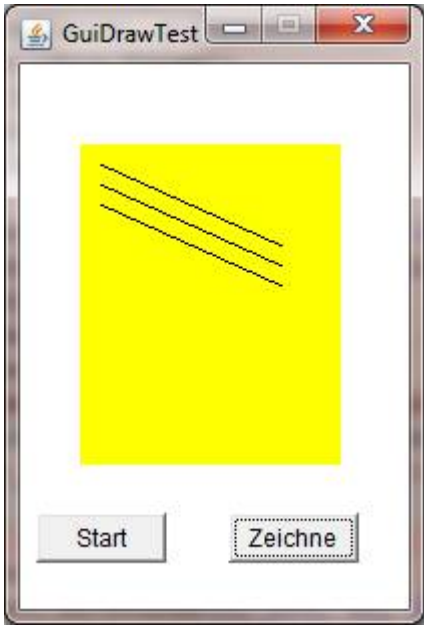
Problembeschreibung:



In einem Java-GUI-Formular sollen (z. B. auf einem Canvas-Objekt) einfache geometrische Figuren (z. B. Linien, Kreise, Ellipsen, ...) oder auch bezeichnender Text dargestellt werden.

Lösungsansatz:

1.) Die abstrakte Java-Klasse Graphics stellt hierzu geeignete Methoden bereit: z. B. drawLine(), drawOval(), drawRect(), drawString().

2.) Jede Klasse, die von der Java-Klasse Component abgeleitet ist (z. B. Canvas, Panel, ...), erbt die Methoden paint() und repaint(). In der Methode paint() können die Graphics-Methoden verwendet werden. Diese Methode soll aber nie vom Programm aufgerufen werden, sondern wird automatisch beim Start zur Darstellung z. B. des Canvas-Objektes verwendet. Soll der Inhalt erneut dargestellt werden, so wird die Methode repaint() aufgerufen.

Konkrete Implementierung:

Beschrieben wird hier das Vorgehen bei der Nutzung von AWT-Objekten (z. B. Canvas). Für Swing-Objekte (z.B. JCanvas) ist das Vorgehen analog.
Zuerst ist zu entscheiden, in welchem Java-Objekt gezeichnet werden soll. Wir entscheiden uns für die Klasse Canvas, da diese direkt von der Klasse Component abgeleitet ist. Da nun aber die Methode paint() der Component-Klasse den eigenen Bedürfnissen angepasst werden soll, muss die Methode neu implementiert, die alte Methode also überschrieben werden. Deshalb ist es nötig, eine aus der Klasse Canvas spezialisierte Klasse und darin die überladende Methode paint() zu erzeugen. Wir nennen die neue Klasse DrawCanvas. Ein Beispiel für ihre Implementierung ist weiter unten zu sehen.

Um zu unterschiedlichen Situationen mit Hilfe der Methode repaint() andere oder mehr Zeichenobjekte darzustellen, kann der Klasse DrawCanvas ein zusätzliches Attribut (hier z. B. step) spendiert werden. So kann in Abhängigkeit vom Wert des DrawCanvas-Attributs step die Methode paint() unterschiedlich oder zusätzlich arbeiten. Hier im Beispiel werden beim Start erst eine, dann nach Klick auf den Zeichne-Button zwei, dann drei Linien untereinander gezeichnet.

Auf dem Java-GUI-Formular muss nun ein Objekt dieser neuen Klasse DrawCanvas eingefügt werden. Beim erstmaligen Darstellen wird dann die in der Klasse DrawCanvas implementierte Methode paint() automatisch aufgerufen. Sollen Schritt für Schritt z. B. weitere Linien hinzukommen, sollte vor dem erneuten Aufruf der Methode repaint() der Attributwert von step angepasst (erhöht) werden (z.B. durch incStep()).

Auf der zweiten Seite dieser Information ist als Beispiel die Implementierung einer spezialisierten Frame-Klasse GuiDrawTest beschrieben. Das Formular enthält ein DrawCanvas-Objekt, einen Start-Button (DrawCanvas-Objekt neu mit Anfangszustand darstellen) und einen Zeichne-Button (zusätzliche Elemente auf dem DrawCanvas-Objekt ergänzend zeichnen). Soll das Beispiel mit dem JavaEditor nachvollzogen werden, genügt es, die Klasse DrawCanvas manuell zu erstellen, dann die Klasse GuiDrawTest wie gewohnt interaktiv mit den beiden Buttons und einem einfachen Canvas-Objekt zu erzeugen und anschließend von Hand im Code nur die Zeile zur Deklaration und Initialisierung des Canvas-Objektes anzupassen:

```
private DrawCanvas dc_test = new DrawCanvas();
```

Hilfreiche Links:

Java JDK API: http://docs.oracle.com/javase/8/docs/api/
dort ggf. die FRAMES einschalten, dann im Frame links oben das gewünschte Package (z.B. java.awt) auswählen, dann im Frame links unten die gewünschte Klasse (z. B. Graphics oder Component oder Canvas) auswählen, dann im Haupt-Frame rechts unten die Klassenbeschreibungen studieren.

Oracle Tutorial: Painting in AWT and Swing:
http://www.oracle.com/technetwork/java/painting-140037.html

Java-Code der spezialisierten Beispiel-Klasse DrawCanvas:

```
import java.awt.*;

public class DrawCanvas extends Canvas {
  // Attribut, um unterschiedlich oder ergänzend zu zeichnen:
  private int step = 0;

  // Methode, um den Attributwert neu zu setzen:
  public void setStep (int pStep){
    this.step = pStep;
  }

  // Methode, um den Attributwert zu erhöhen:
  public void incStep (){
    this.step++;
  }

  // Überladen der ererbten Component-Methode:
  @Override
  public void paint( Graphics g )
  {
    // Aufruf der Methode der Oberklasse (Component.paint())
    super.paint( g );
    // Zeichen-Methoden-Aufrufe abhängig vom Wert des Attributes step:
    if (this.step >= 0) {
      g.drawLine( 10, 10, 100, 50 );
    } // end of if
    if (this.step > 0) {
      g.drawLine( 10, 20, 100, 60 );
    } // end of if
    if (this.step > 1) {
      g.drawLine( 10, 30, 100, 70 );
    } // end of if
  }
}
```

Java-Code der Beispiel-Klasse GuiDrawTest:

```
import java.awt.*;
import java.awt.event.*;

public class GuiDrawTest extends Frame {
  // Attribute:
  private Button bt_start = new Button();
  private Button bt_draw = new Button();
  private DrawCanvas dc_test = new DrawCanvas();

  // Konstruktor:
  public GuiDrawTest(String title) {
    super(title);
    addWindowListener(new WindowAdapter() {
      public void windowClosing(
        WindowEvent evt) { dispose(); }
      });
    int frameWidth = 200;
    int frameHeight = 300;
    setSize(frameWidth, frameHeight);
    Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
    int x = (d.width - getSize().width) / 2;
    int y = (d.height - getSize().height) / 2;
    setLocation(x, y);
    setResizable(false);
    Panel cp = new Panel(null);
    add(cp);
    // Anfang Komponenten
    dc_test.setBounds(30, 40, 130, 160);
    dc_test.setBackground(Color.YELLOW);
    cp.add(dc_test);
    bt_draw.setBounds(104, 224, 65, 25);
    bt_draw.setLabel("Zeichne");
    bt_draw.addActionListener(
      new ActionListener() {
        public void actionPerformed(
          ActionEvent evt) {
            bt_draw_ActionPerformed(evt);
          }
      });
    cp.add(bt_draw);
    bt_start.setBounds(8, 224, 65, 25);
    bt_start.setLabel("Start");
    bt_start.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) { bt_start_ActionPerformed(evt); }
      });
    cp.add(bt_start);
    // Ende Komponenten

    setVisible(true);
  } // end of public GuiDrawTest

  // Anfang Methoden
  public void bt_draw_ActionPerformed(ActionEvent evt) {
    // TODO hier Quelltext einfügen
    this.dc_test.incStep ();
    this.dc_test.repaint();
  } // end of bt_draw_ActionPerformed

  public void bt_start_ActionPerformed(ActionEvent evt) {
    // TODO hier Quelltext einfügen
    this.dc_test.setStep (0);
    this.dc_test.repaint();
  } // end of bt_start_ActionPerformed
  // Ende Methoden

  public static void main(String[] args) {
    new GuiDrawTest("GuiDrawTest");
  } // end of main
} // end of class GuiDrawTest
```