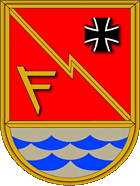
Schule Informationstechnik

der Bundeswehr



**Sprachausbildung Java**



**Übungen**

**"GUI - Swing"**

Aufgabe 1: Swing Applikation erstellen

Implementieren Sie ein Swing-Programm, welches

1. ein JFrame erzeugt,
2. das Fenster auf die Größe 600 x 200 bringt,
3. den Titel auf "Man nennt es "Das Fenster"" setzt und das
4. Fenster auf dem Bildschirm anzeigt.

Aufgabe 2: Ereignisse

Einige Softwareprodukte, wie etwa Winamp, haben für Fenster ein „Einrasten“ implementiert. Wenn man etwa 20 Pixel vom Rand entfernt ist, heftet sich das Fenster automatisch an den Rand. Wie wird so etwas prinzipiell funktionieren?

1. Tipp: Die maximale Auflösung liefert die Methode getScreenSize() der Klasse Toolkit. Ein Fenster kann man mit setPosition(...) auf eine absolute Stelle setzen.

2. Tipp: Es gilt nur noch, die aktuellen Koordinaten während des Verschiebens zu ermitteln. Finde dazu heraus, ob die Methode componentMoved() ein Schlüssel zu dem Problem ist.

Implementieren Sie das „Einrasten“ für das in Aufgabe 1 erstellte Fenster, indem Sie:

1. ComponentAdapter erweitern,
2. die Funktionalität in componentMoved() implementieren und
3. den ComponentAdapter am Fenster registrieren.

Aufgabe 3: MouseEvents

Erzeugen Sie ein Fenster und fangen Sie Maus-Events ab. Geben Sie die Koordinaten im Fenstertitel aus. Form: „x=2, y=3“.

Aufgabe 4: Fenster Ereignis

Erzeugen Sie ein Fenster. Die Applikation soll nun auf die Eigenschaften „Miniaturisieren“ und „Schließen“ der Fenster reagieren. Verwenden Sie dazu die Schnittstelle WindowListener oder die Klasse WindowAdapter.

* Die Applikation soll beim Versuch des Schließens beendet werden (System.exit(0)).
* Beim Versuch das Fenster zu minimieren, soll es selbstständig an eine zufällige andere Position auf dem Bildschirm springen.

Aufgabe 5: JClosableFrame

Implementieren Sie die Hilfsklasse JClosableFrame, die automatisch das Fenster schließt, falls der Benutzer das [X] im Fensterrahmen (oder Alt+F4) drückt. JClosableFrame soll eine Unterklasse von javax.swing.JFrame sein. Achten Sie auf die Behandlung der Ereignisse mit setDefaultCloseOperation(...). Sie können dann diese Klasse für Ihre eigenen Programme nutzen und müssen sich dann nicht mehr um die Schließen-Ereignisse kümmern. Zwar würde das Fenster mit setDefaultCloseOperation(...) auch geschlossen, aber wir könnten auf das Ende nicht mehr mit einer eigenen Aktion reagieren.

* Geben Sie, falls das Fenster geschlossen wird, eine Meldung auf dem Bildschirm aus.
* Nutzen Sie die Klasse javax.swing.JOptionPane und die Methode showConfirmDialog(...), um einen Bestätigungsdialog anzuzeigen. Die API-Dokumentation zeigt ein Beispiel für diese Methode. Man kann als String eine Nachricht einsetzen, wie etwa "Soll die Applikation wirklich beendet werden?"

Aufgabe 6: JFrame

Schreiben Sie ein Programm, welches ein javax.swing.JFrame mit einem javax.swing.JLabel auf dem Bildschirm ausgibt. Setzen Sie die Größe des Fensters auf 256 x 256 Pixel.

Fügen Sie dem javax.swing.JLabel Fenster einen java.awt.event.MouseMotionListener hinzu. Wird die Maus bewegt, sollen die Koordinaten auf den Bildschirm gebracht werden.

Erfragen Sie aus dem java.awt.event.MouseEvent den Verursacher des Ereignisses. (Das wird dann ja das javax.swing.JLabel gewesen sein.) Nutzen Sie die x- und y-Koordinaten, um daraus ein Farb-Objekt zu bauen: new java.awt.Color(x, y, 0). Setzen Sie dann diese Farbe mit Hilfe der Methode setBackground(...).

Aufgabe 7: JButton

Ein javax.swing.JFrame soll über einen javax.swing.JButton auf dem Bildschirm verfügen. Wird der Button drückt, soll die Applikation geschlossen werden.

Implementieren Sie dieses Verhalten.

Aufgabe 8: GridLayout

Wie kann eine Telefontastatur mit Hilfe eines java.awt.GridLayout gezeichnet werden?

1. Nutzen Sie einen Konstruktor, mit dem man etwas Abstand zwischen die Tasten bekommt.
2. Setzen Sie mit geschachtelten Layouts über die Tastatur ein javax.swing.JTextField und unter die Tastatur ein javax.swing.JTextArea-Objekt. Wie muss man die Layoutmanager verschachteln? Denken Sie an javax.swing.JPanel, um einen Container für Komponenten zu bilden.
3. Werden Schaltflächen gedrückt, sollen im oberen Textfeld die gedrückten Ziffern gesammelt werden. Nutzen Sie dafür intern einen String, der bei jeder gedrückten Taste wächst.
4. Erweiterung: Werden in der Textzeile bis zu acht Ziffern eingesetzt, so sollen im Textfeld zu einer Telefonnummer mögliche Merkhilfen genannt werden.

Die Merkhilfe basiert auf der Idee, dass auf einer Telefontaste zusätzliche Buchstaben stehen, sodass aus einer Telefonnummer eine Zeichenkette als Gedächtnisstütze wird. Folgende Zuweisung wird vorgenommen: 1:1, 2:ABC, 3:DEF, 4:GHI, 5:JKL, 6:MNO, 7:PQRS, 8:TUV, 9:WXYZ.

Stellen Sie alle möglichen Merkhilfen für diese Telefonnummer in der JTextArea dar.

Aufgabe 9: ActionListener

Schreiben Sie eine Swing-Applikation zum Umrechnen von DM-Beträgen in Euro-Beträge. Bauen Sie dazu eine grafische Oberfläche mit zwei Textfeldern (javax.swing.JTextField) auf. Wenn man in der oberen Textbox (Euro) etwas schreibt, dann sollte in der unteren Textbox (DM) der Wert angezeigt werden. Die Umrechnung sollte umgekehrt genauso funktionieren. Implementieren Sie für die Ereignisbehandlung einen ActionListener.

Nutzen Sie ein java.awt.BorderLayout, um die Textfelder anzuordnen. (Man kann auch mit einem GridLayout-Manager rechts den Text (Euro/DM) und links die Textfelder platzieren).

* Anstatt erst bei einer Aktivierung der Return-Taste die Berechnung vorzunehmen, kann man auch bei jedem gedrückten Zeichen die Umrechnung machen.
* Erweitern Sie das Programm so, dass fehlerhafte Zeichen aus der Eingabe gelöscht werden.

Aufgabe 10: Scrollbars / Textfelder

1. Ein Fenster soll einen horizontalen Schieberegler javax.swing.JScrollBar enthalten. Implementieren Sie eine Ereignisbehandlung und geben Sie den selektierten Wert auf dem Bildschirm aus.
2. Der Schieberegler soll die Werte 0 bis 255 annehmen können.
3. Erweiterung: Das Fenster soll zusätzlich ein java.awt.Canvas-Objekt enthalten.
4. Geben Sie dem Hintergrund die Standard-Farbe Schwarz. Sie Aufgabe 5.
5. Wenn der Schieberegler bewegt wird, soll sich die Farbe im Textfeld ändern. Es soll nur der Blau-Wert verändert werden.

Aufgabe 11: UPN Taschenrechner

Viele einfache Programmiersprachen erlauben arithmetische Ausdrücke in umgekehrt Polnischer Notation (UPN). Ein bekanntes Beispiel dafür ist Postscript.

Programmieren Sie einen Taschenrechner, der mittels des java.util.StringTokenizer einen String (wie etwa „12 34 + 23 \*“) parst und das Ergebnis auswertet.

UPN: https://de.wikipedia.org/wiki/Umgekehrte\_polnische\_Notation

Aufgabe 12: JList

Schreiben Sie ein Programm zur Euro-Umrechnung. Dabei soll eine javax.swing.JList zu sehen sein, in welcher die andere Währung ausgewählt werden kann. Wird in ein Eingabefeld ein Betrag in Euro eingesetzt, soll der Betrag in einem anderen Feld in der angewählten Währung ausgegeben werden. Bei der European Central Bank gibt es die aktuellen Kurse (https://www.ecb.europa.eu/stats/policy\_and\_exchange\_rates/euro\_reference\_exchange\_rates/html/index.en.html).

Implementieren Sie mindestens: USD, JPY, GBP, RUB, AUD, ISK.

Aufgabe 13: Graphics

* Implementieren Sie eine paintComponent(...)-Methode (JComponent), um eine Linie anzuzeigen. Über die Zeichenmethoden kann man unter der Dokumentation von java.awt.Graphics mehr erfahren.
* Wie kann man Punkte darstellen? Schreiben Sie eine Methode drawPoint( Graphics g, int x, int y ), die einen Punkt auf dem Bildschirm ausgibt.

http://www.oracle.com/technetwork/java/painting-140037.html

Aufgabe 14: Kreise

Der bekannte Kreisalgorithmus von Bresenham kann noch weiter beschleunigt werden. Der Algorithmus kommt ohne Multiplikation oder Lookup-Tabellen aus.

* Implementieren Sie nachfolgendes Programm in Java.

// FCircle --------------------------------------------

// Draws a circle using Frith's Algorithm.

void FCircle(int x, int y, int radius, uchar color)

{

int balance, xoff, yoff;

xoff = 0;

yoff = radius;

balance = -radius;

do {

SetPixel(x+xoff, y+yoff, color);

SetPixel(x-xoff, y+yoff, color);

SetPixel(x-xoff, y-yoff, color);

SetPixel(x+xoff, y-yoff, color);

SetPixel(x+yoff, y+xoff, color);

SetPixel(x-yoff, y+xoff, color);

SetPixel(x-yoff, y-xoff, color);

SetPixel(x+yoff, y-xoff, color);

if ((balance += xoff++ + xoff) >= 0)

balance -= --yoff + yoff;

} while (xoff <= yoff);

} // FCircle //

* Vergleichen Sie die Geschwindigkeit der Bibliotheksfunktion und FCircle.
* Zeichnen Sie eine Punktefolge, die durch eine Iterationsvorschrift gegeben wird. Wähle eine der drei Abbildungen zufällig aus:
  + (x,y) -> (x/2,y/2)
  + (x,y) -> (1/2+x/2,1/2+y/2)
  + (x,y) -> (1/2+x/2,y/2)
* Zeichne etwa 50.000 Punkte dieser Iteration. Multipliziere den Punkt vor dem Zeichnen mit 200.

Aufgabe 15: Farben

Schreiben Sie ein Programm, das ein quadratisches Canvas mit 1000 zufälligen Punktepaaren füllt. Markieren Sie diejenigen Punkte, die in einen beschriebenen Kreis fallen, rot (setColor(Color.RED)) aus java.awt.Graphics, die anderen schwarz. Speichern Sie die Punkte in einem Array, damit nicht bei jedem paintComponent(...) neue Punkte gezeichnet werden.

Aufgabe 16: Graphics2D

Schreiben Sie ein Programm, welches mit der 2D-API ein zufälliges Linienmuster zeichnet.

* Schalten das Antialiasing ein und aus.
* Zeichnen Sie die Linien mit einer Strichstärke von 3 Pixeln.
* Die Linien sollen ein Muster bekommen: zwei Pixel frei und ein Pixel gesetzt.

Aufgabe 17: 2d-Objekte

* Programmieren Sie ein Beispiel mit AffineTransformation, das etwa ein Rechteck der Größe 100×100 Pixel skaliert und um 45 Grad rotiert wird.
* Zeichnen Sie das Rechteck mit Linien der Stärke 5 und gestrichelt. Konfigurieren Sie das Zeichnen so, dass die Linienecken rund sind.
* Nutzen Sie GeneralPath um einen Stift in die Mitte des Schirms zu setzen, dann 10 Pixel nach rechts, um 45 Grad gedreht, dann 10 Pixel nach links zu gehen. Affine Transformationen soll es nicht geben.