Course IPJava: Image Processing with Java
Chapter C4: The Lowpass Project

Projekt lowpass1 mit leerem Fenster

Im Directory `C:\temp` ein Sub-Directory `lowpass1` anlegen.
TextPad starten `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\lowpass1\Lowpass1.java`
Schreiben Sie folgende Zeilen:

```
import java.awt.Container;
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JFrame;

public class Lowpass1
{
  public static void main(String args[])
  {
    JFrame frame=new JFrame("Lowpass1");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(800,600);
    Container cp=frame.getContentPane();
    cp.setBackground(Color.WHITE);
    cp.setLayout(new BorderLayout());
    frame.setVisible(true);
  }
}
```

Übersetzen mit der Tastenkombination `Strg+1`. Ausführen mit der Tastenkombination `Strg+2`. Es öffnet sich ein Fenster, das sich nur minimieren, maximieren und schließen lässt. Die Klasse `Lowpass1` dient also lediglich dem Aufbau unserer GUI und (später) allen notwendigen Objekt-Instantiierungen.

Zeichenfläche

Beenden Sie das Programm.
Legen Sie eine neue Datei in TextPad an: `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\lowpass1\DrawArea.java`
Schreiben Sie folgende Klasse:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.JComponent;

public class DrawArea extends JComponent
{
  int myWidth, myHeight;
  int pWidth, pHeight;
  final int X_SIZE=11;
  final int Y_SIZE=12;
  byte[][] b=new byte[Y_SIZE][X_SIZE];
  Color[] color;
```

```
  public DrawArea()
  { Dimension size=getSize();
    myWidth=size.width;
    myHeight=size.height;
    addComponentListener(new ComponentAdapter()
    { public void componentResized(ComponentEvent ev)
      {
        Dimension size=getSize();
        myWidth=size.width;
        myHeight=size.height;
      }
    });
    fillColorArray();
    clear();
  }
  private void fillColorArray()
  { color=new Color[10];
    for(int i=0; i<color.length; i++)
      color[i]=new Color(i*25,i*25,i*25);
  }
  public void paintComponent(Graphics g)
  {
    super.paintComponent(g);
    setPixelMeasuresToArea();
    for(int y=0; y<Y_SIZE; y++)
      for(int x=0; x<X_SIZE; x++)
      {
        g.setColor(color[b[y][x]]);
        g.fillRect(x*pWidth, y*pHeight, pWidth, pHeight);
      }
  }
  /**
   * Sets pWidth and pHeight to values relative to the current DrawArea
   * size.
   */
  private void setPixelMeasuresToArea()
  {
    pWidth=myWidth/X_SIZE;
    pHeight=myHeight/Y_SIZE;
  }
  public void reset()
  {
    byte[][] tmp={{9,0,0,0,0,0,0,0,0,0,9},
                  {0,0,0,0,9,9,9,0,0,0,0},
                  {0,0,0,0,9,5,9,0,0,0,0},
                  {0,0,0,0,9,9,9,0,0,0,0},
                  {0,0,0,0,0,9,0,0,0,0,0},
                  {0,9,9,9,9,9,9,9,9,9,0},
                  {0,0,0,0,9,9,9,0,0,0,0},
                  {0,0,0,0,9,9,9,0,0,0,0},
                  {0,0,0,0,9,0,9,0,0,0,0},
                  {0,0,0,0,9,0,9,0,0,0,0},
                  {0,0,0,0,9,0,9,0,0,0,0},
                  {9,0,0,0,0,0,0,0,0,0,9}};
    b=tmp;
  }
  public void lowpass(int size, int weight)
  {
  }
  public void noise()
  {
  }
  public void clear()
  {
  }
}
```

Übersetzen mit `Strg+1`.

Es wäre ebenso möglich die neue Klasse in die Datei `Lowpass1.java` zu schreiben und nicht in ein extra File zu speichern. Allerdings ist eine zusätzliche Datei wesentlich übersichtlicher.
Nun müssen wir unserem `JFrame` noch eine Instanz der neuen Klasse als GUI-Komponente zuweisen.
Wechseln Sie dazu in TextPad zur Datei `Lowpass1`: `Fenster 1 C:\...\Lowpass1.java`
Ergänzen Sie die Methode `public static void main(String[] args)` von `Lowpass1`, so dass sie so aussieht:

```
    public static void main(String[] args)
    {
      JFrame frame=new JFrame("Tiefpass");
      DrawArea drawArea=new DrawArea();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(800,600);
      Container cp=frame.getContentPane();
      cp.setBackground(Color.WHITE);
      cp.setLayout(new BorderLayout());
      cp.add(drawArea);
      frame.setVisible(true);
    }
```

Übersetzen mit `Strg+1`, ausführen mit `Strg+2`.
Wichtig:
Wenn die Java Runtime Environment eine Java-Klasse ausführt, dann sucht sie automatisch nach der `main()`-Methode in dieser Klasse. Führen Sie immer nur die Klasse mit der `main()`-Methode aus - in unserem Fall also immer nur die Klasse `Lowpass1` - ansonsten erhalten Sie eine Fehlermeldung.

Buttonleiste

Beenden Sie das Programm.
Legen Sie eine neue Datei in TextPad an: `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\lowpass1\ButtonBar.java`
Schreiben Sie folgende Klasse:

```
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.HashMap;
import javax.swing.JPanel;
import javax.swing.JButton;

public class ButtonBar extends JPanel
{
  static public final String[] BUTTON_NAME={"Homunculus", "Lowpass 3x3",
      "Lowpass 5x5", "Lowpass 7x7", "LPMidWeight", "Noise", "Clear"};
  static public final String[] BUTTON_CMD={"drawHomunculus", "useLowpass3",
      "useLowpass5", "useLowpass7", "setLPMidWeight", "useNoise", "clearArea"};
  JButton[] button=new JButton[BUTTON_NAME.length];
  DrawArea drawArea;
  int lpMidWeight=1;
  public ButtonBar(DrawArea draw)
  {
    drawArea=draw;
    setLayout(new GridLayout(0,1));
    ActionListener listener=new ButtonListener();
    for(int i=0; i<button.length; i++)
    {
      if(i==4)
        button[i]=new JButton(BUTTON_NAME[i]+"="+lpMidWeight);
      else
        button[i]=new JButton(BUTTON_NAME[i]);
      button[i].setActionCommand(BUTTON_CMD[i]);
      button[i].addActionListener(listener);
      add(button[i]);
    }
  }
```

```
  //---------------------------------------------------------------------------
  class ButtonListener implements ActionListener
  {
    HashMap commandMap=new HashMap(BUTTON_CMD.length);
    public ButtonListener()
    {
      for(int i=0; i<BUTTON_CMD.length;i++)
        commandMap.put(BUTTON_CMD[i], new Integer(i));
    }
    public void actionPerformed(ActionEvent ev)
    {
      int cmd=((Integer)commandMap.get(ev.getActionCommand())).intValue();
      switch(cmd)
      {
        case 0:
          drawArea.reset();
          drawArea.repaint();
          break;
      }
    }
  }
}
```

Übersetzen mit `Strg+1`.
Nun müssen wir (wie vorher die `DrawArea`) unserem `JFrame` noch die Buttonleiste zuweisen.
Wechseln Sie dazu in TextPad zur Datei `Lowpass1`: `Fenster 1 C:\...\Lowpass1.java`
Fügen Sie die folgenden Zeilen als vorletzte Anweisungen (vor die Zeile `frame.setVisible(true);`) in die Methode `public static void main(String[] args)` von `Lowpass1` ein:

```
    ButtonBar buttonBar=new ButtonBar(drawArea);
    cp.add(buttonBar, BorderLayout.EAST);
```

Übersetzen mit `Strg+1`, ausführen mit `Strg+2`. Erproben Sie die Buttonleiste. Allerdings funktioniert bisher nur der Button `Homunculus`.

Lowpass

Beenden Sie das Programm.
Wechseln Sie in TextPad zur Datei `DrawArea`: `Fenster 2 C:\...\DrawArea.java`
Version 2: Erweitern Sie die Methode `public void lowpass(int size, int weight)` der Klasse `DrawArea`, bis sie so aussieht:

```
    public void lowpass(int size, int weight)
    {
      int lowpassSize=size;
      int addMidWeight=weight-1;
      int norm, x, y, xx, yy;
      int d=lowpassSize/2;
      int sum;

      byte[][] temp=new byte[Y_SIZE][X_SIZE];
      for(y=0; y<Y_SIZE; y++)
        for(x=0; x<X_SIZE; x++)
          temp[y][x]=b[y][x];
      for(y=0; y<Y_SIZE; y++)
        for(x=0; x<X_SIZE; x++)
        {
          sum=addMidWeight*temp[y][x];
          norm=addMidWeight;
          for(yy=y-d; yy<=y+d; yy++)
          {
            if(yy<0 || yy>=Y_SIZE)
              continue;
            for(xx=x-d; xx<=x+d; xx++)
            {
              if(xx<0 || xx>=X_SIZE)
                continue;
              sum+=temp[yy][xx];
              norm++;
            }
          }
          b[y][x]=(byte)(sum/norm);
        }
    }
```

Übersetzen mit `Strg+1`.
Wechseln Sie in TextPad zur Datei `ButtonBar`: `Fenster 3 C:\...\ButtonBar.java`
Schreiben Sie folgende neuen Cases unterhalb des letzten `break;` von `case 0:` in der Methode `public void actionPerformed(ActionEvent ev)` der Event Handler Klasse `ButtonListener`:

```
    case 1:
      drawArea.lowpass(3, lpMidWeight);
      drawArea.repaint();
      break;
    case 2:
      drawArea.lowpass(5, lpMidWeight);
      drawArea.repaint();
      break;
    case 3:
      drawArea.lowpass(7, lpMidWeight);
      drawArea.repaint();
      break;
    case 4:
      lpMidWeight++;
      button[4].setText(BUTTON_NAME[4]+"="+lpMidWeight);
      break;
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Lowpass1`: `Fenster 1 C:\...\Lowpass1.java`. Ausführen mit `Strg+2`. Erproben Sie die 3 Tiefpässe und die Wirkung von hö;heren (> 9) Faktoren von `LPMidWeight`.

Noise und Clear

Beenden Sie das Programm.
Wechseln Sie in TextPad zur Datei `DrawArea`: `Fenster 2 C:\...\DrawArea.java`
Version 3: Erweitern Sie die Methode `public void noise()` der Klasse `DrawArea`, bis sie so aussieht:

```
public void noise()
{
  for(int y=0; y<Y_SIZE; y++)
    for(int x=0; x<X_SIZE; x++)
    {
      int noise=(int)(Math.random()*3)-1;
      noise+=b[y][x];
      if(noise<0)
        b[y][x]=0;
      else if (noise>9)
        b[y][x]=9;
      else
        b[y][x]=(byte)noise;
    }
}
```

Erweitern Sie die Methode `public void clear()` der Klasse `DrawArea`, bis sie so aussieht:

```
public void clear()
{
  for(int y=0; y<Y_SIZE; y++)
    for(int x=0; x<X_SIZE; x++)
      b[y][x]=0;
}
```

Übersetzen mit `Strg+1`.
Wechseln Sie in TextPad zur Datei `ButtonBar`: `Fenster 3 C:\...\ButtonBar.java`
Schreiben Sie folgende neuen Cases unterhalb des letzten `break;` von `case 4:` in der Methode `public void actionPerformed(ActionEvent ev)` der Event Handler Klasse `ButtonListener`:

```
case 5:
  drawArea.noise();
  drawArea.repaint();
  break;
case 6:
  lpMidWeight=1;
  button[4].setText(BUTTON_NAME[4]+"="+lpMidWeight);
  drawArea.clear();
  drawArea.repaint();
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Lowpass1`: `Fenster 1 C:\...\Lowpass1.java`. Ausführen mit `Strg+2`. Erproben Sie `Noise` (auch mehrfach drücken) und `Clear`.

Experimente

(1) Verändern Sie Pixel in der Homunculus-Matrix (nur Werte zwischen 0 und 9 !).
(2) Vergrö;ßern Sie `final int X_SIZE` von `11` auf `12` und `final int Y_SIZE` von `12` auf `13`. Fügen Sie an die Homunculus-Matrix entsprechend eine Spalte und eine Zeile an.

Weitere Aufgaben

Lesen sie die JavaTM 2 SDK, Standard Edition Documentation zu den Sprachelementen, die Sie geschrieben haben (siehe: http://java.sun.com/j2se/1.4.1/search.htm). Schauen Sie sich vor allem die Dokumentation zu den Klassen `java.awt.GridLayout`, `java.awt.BorderLayout`, `javax.swing.JButton`, `javax.swing.JPanel` und `java.util.HashMap` an.
Erarbeiten Sie sich die Logik der 4 verschachtelten `for`-Schleifen in der Methode `public void lowpass(int size, int weight)`.
Programmieren Sie eine vereinfachte Methode `myLowpass` der festen Größe 3x3, ohne Randbehandlung und ohne Mittengewichtung.
Erfinden und erproben Sie neue Varianten des Programms in Form von neuen Projekten `lowpass2`, `lowpass3` usw. nach obigem Muster.