Course IPJava: Image Processing with Java
Chapter C3: The Filter Project

Projekt filter1 mit leerem Fenster

Im Directory `C:\temp` ein Sub-Directory `filter1` anlegen.
TextPad starten `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\filter1\Filter1.java`
Schreiben Sie folgende Zeilen:

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

public class Filter1
{
  public static void main(String args[])
  {
    JFrame frame=new JFrame("Filter1");
    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 `Filter1` dient also lediglich dem Aufbau unserer GUI und (später) allen notwendigen Objekt-Instantiierungen.

Zeichenfläche, Vorbereitungen für 6 Bilder

Beenden Sie das Programm.
Um mit dem später eingelesenen Bild arbeiten zu können (es zum Beispiel auf ein sechstel der Grö;ße des Fensters zu verkleinern), benötigen wir eine "Zeichenfläche", die wir als zusätzliche Komponente in unser `JFrame` einbauen und dessen Größe wir jedes mal neu erfassen, wenn sich die Größe des Fensters verändert.
Legen Sie dazu eine neue Datei in TextPad an: `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\filter1\DrawArea.java`
Schreiben Sie folgende Klasse:

```
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.awt.image.IndexColorModel;
import javax.swing.JComponent;
import java.util.Arrays;
```

```
public class DrawArea extends JComponent
{
  BufferedImage image;
  BufferedImage noiseImg;
  BufferedImage lowpassImg;
  BufferedImage highVerticalImg;
  BufferedImage highHorizontalImg;
  BufferedImage highGradientImg;
  IndexColorModel grayICM;
  int[] grayArray;//gray values
  int imgWidth, imgHeight;
  int myWidth, myHeight, fontHeight=0;
  String message="Open a .bmp file";
  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;
      }
    });
    //create a gray value palette:
    byte[] color=new byte[256];
    byte[] black=new byte[256];
    for(int i=0; i<256; i++)
    {
      color[i]=(byte)i;
      black[i]=0;
    }
    grayICM=new IndexColorModel(8, 256, color, color, color);
    //-----------------------------
  }
  public void paintComponent(Graphics g)
  {
    super.paintComponent(g);
    if(fontHeight==0)
      fontHeight=g.getFont().getSize();
    if(image!=null)
    {
      int x=myWidth/3;
      int y=myHeight/2;
      g.drawImage(              image,   0, 0, x, y, this);
      if(noiseImg!=null)
        g.drawImage(         noiseImg,   x, 0, x, y, this);
      if(lowpassImg!=null)
        g.drawImage(       lowpassImg, 2*x, 0, x, y, this);
      if(highVerticalImg!=null)
        g.drawImage(  highVerticalImg,   0, y, x, y, this);
      if(highHorizontalImg!=null)
        g.drawImage(highHorizontalImg,   x, y, x, y, this);
      if(highGradientImg!=null)
        g.drawImage(  highGradientImg, 2*x, y, x, y, this);
    }
    else
    {
      g.setColor(Color.RED);
      g.drawString(message, 0, fontHeight);
      g.setColor(Color.BLACK);
    }
  }
```

```
  public void setImage(int[] img, int[] palette, int colorDepth,
      int width, int height)
  {
  }
  public void setMessage(String message)
  {
  }
  private void setNoiseImage()
  {
  }
  private void setLowpassImage()
  {
  }
  private void setHighpassImages()
  {
  }
}
```

Übersetzen mit `Strg+1`.
Es wäre ebenso möglich die neue Klasse in die Datei `Filter1.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 `Filter1`: `Fenster 1 C:\...\Filter1.java`
Ergänzen Sie die Methode `public static void main(String[] args)` von `Filter1`, so dass sie so aussieht:

```
    public static void main(String[] args)
    {
      JFrame frame=new JFrame("Filter (Rauschen, Tiefpass, Hochpass)");
      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`. Sie werden feststellen, dass sich (bis auf den zur Zeit noch bedeutungslosen Text `Open a .bmp file`) das Aussehen und Verhalten der Applikation überhaupt nicht verändert hat.
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 `Filter1` - ansonsten erhalten Sie eine Fehlermeldung.

Menüleiste

Beenden Sie das Programm.
Um nun `*.bmp` Dateien auf unsere Zeichenfläche laden zu können, benötigen wir eine Menüleiste mit dem Menü `File` und dem Menüpunkt `Read`.
Legen Sie dazu eine neue Datei in TextPad an: `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\filter1\MenuBar.java`
Schreiben Sie folgende zwei Klassen:

```
import java.awt.Graphics;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.KeyEvent;
import javax.swing.*;
import javax.swing.filechooser.FileFilter;
import java.io.*;
```

```
public class MenuBar extends JMenuBar
{
  JMenu fileMenu;
  public MenuBar(DrawArea drawArea)
  {
    add(fileMenu=new FileMenu(drawArea));
  }
}
//============================================================================
class FileMenu extends JMenu
{
  static public final String READ_ITEM_CMD="readFile";
  static public final String EXIT_ITEM_CMD="exitProgram";
  JMenuItem readItem, exitItem;
  DrawArea drawArea;
  public FileMenu(DrawArea draw)
  {
    super("File");
    drawArea=draw;
    setMnemonic(KeyEvent.VK_F);
    ActionListener listener=new MenuActionListener();
    readItem=new JMenuItem("Read", KeyEvent.VK_R);
    exitItem=new JMenuItem("Exit");
    readItem.setAccelerator(KeyStroke.getKeyStroke("control R"));
    readItem.setActionCommand(READ_ITEM_CMD);
    exitItem.setActionCommand(EXIT_ITEM_CMD);
    readItem.addActionListener(listener);
    exitItem.addActionListener(listener);
    add(readItem);
    add(exitItem);
  }
  //--------------------------------------------------------------------------
  class MenuActionListener implements ActionListener
  {
    public void actionPerformed(ActionEvent ev)
    {
      String cmd=ev.getActionCommand();
      if(cmd.equals(FileMenu.READ_ITEM_CMD))
      {
      }
      else if(cmd.equals(FileMenu.EXIT_ITEM_CMD))
        System.exit(0);
    }
  }
}
```

Übersetzen mit `Strg+1`.
Nun müssen wir (wie vorher die `DrawArea`) unserem `JFrame` noch die Menüleiste zuweisen.
Wechseln Sie dazu in TextPad zur Datei `Filter1`: `Fenster 1 C:\...\Filter1.java`
Fügen Sie die folgende Code-Zeile als 3. Anweisung (unter die Instantiierung der `drawArea`) in die Methode `public static void main(String[] args)` von `Filter1` ein:

```
    frame.setJMenuBar(new MenuBar(drawArea));
```

Übersetzen mit `Strg+1`, ausführen mit `Strg+2`. Erproben Sie das neue Menü. Allerdings funktioniert bisher nur der Menüpunkt `Exit`.

Bild lesen und anzeigen

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

```
public void setImage(int[] img, int[] palette, int colorDepth,
    int width, int height)
{
  imgWidth=width;
  imgHeight=height;
  if(colorDepth==8)
  {
    int colorCount=palette.length/3;
    byte[] red=new byte[colorCount];
    byte[] green=new byte[colorCount];
    byte[] blue=new byte[colorCount];
    for(int i=0; i<colorCount; i++)
    {
      red[i]=(byte)palette[i*3];
      green[i]=(byte)palette[i*3+1];
      blue[i]=(byte)palette[i*3+2];
    }
    IndexColorModel icm=new IndexColorModel(8, colorCount, red, green, blue);
    image=new BufferedImage(
        width, height, BufferedImage.TYPE_BYTE_INDEXED, icm);
    image.getRaster().setPixels(0, 0, width, height, img);
    //8 bit gray values:
    int[] paletteGray=new int[colorCount];
    for(int i=0; i<colorCount; i++)
      paletteGray[i]=(palette[i*3]+palette[i*3+1]+palette[i*3+2])/3;
    grayArray=new int[img.length];
    for(int i=0; i<img.length; i++)
      grayArray[i]=paletteGray[img[i]];
    //-----------------------------
  }
  else
  {
    image=new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    image.getRaster().setPixels(0, 0, width, height, img);
    //24 bit gray values:
    grayArray=new int[img.length/3];
    for(int i=0; i<img.length; i+=3)
      grayArray[i/3]=(img[i]+img[i+1]+img[i+2])/3;
    //-------------------------------
  }
  setNoiseImage();
  setLowpassImage();
  setHighpassImages();
  repaint();
}
```

Erweitern Sie weiterhin die Methode `public void setMessage(String message)` der Klasse `DrawArea`, bis sie so aussieht:

```
public void setMessage(String message)
{
  this.message=message;
  image=null;
  repaint();
}
```

Übersetzen mit `Strg+1`.
Zum einlesen einer `*.bmp` Datei benötigen wir einen `JFileChooser`, der sich dann öffnet, wenn in der Menüleiste auf den Menüpunkt `Read` geklickt wird. Nachdem die Datei ausgewählt wurde und der OK-Button gedrückt wurde, wird die Datei mit Hilfe eines `BufferedInputStream` Byte für Byte eingelesen.
Wechseln Sie dazu in TextPad zur Datei `MenuBar`: `Fenster 3 C:\...\MenuBar.java`

Ergänzen Sie die `MenuActionListener` Klasse (innere Klasse von `FileMenu`), bis sie so aussieht:

```
    class MenuActionListener implements ActionListener
    {
      JFileChooser fileChooser;
      int[] bitmapFileHeader=new int[14];
      int[] bitmapInfoHeader=new int[40];
      int[] bitmapColorTable;
      int[] bitmapData;
      static final String NOT_BMP_FILE_ERROR_MSG=
        "File is not supported or corrupt. Choose a Windows Histo-File (.bmp).";
      static final String WRONG_BIT_COUNT_ERROR_MSG=
        "Choose a 8 or 24 bit Windows Histo-File (.bmp).";
      static final String FILE_NOT_FOUND_ERROR_MSG=
        "The chosen file does not exist.";
      static final String IO_ERROR_MSG=
        "An IOException occured when reading the file.";
      public void actionPerformed(ActionEvent ev)
      {
        String cmd=ev.getActionCommand();
        if(cmd.equals(FileMenu.READ_ITEM_CMD))
        {
          if(fileChooser==null)
          {
            fileChooser=new JFileChooser(".\\");
            fileChooser.setFileFilter(new FileFilter()
            {
              public boolean accept(File f)
              {
                String path=f.getAbsolutePath().toLowerCase();
                if ((f.isDirectory()) || (path.endsWith(".bmp")))
                  return true;
                return false;
              }
              public String getDescription()
              {
                return "Windows Histo (*.bmp)";
              }
            });
            fileChooser.setAcceptAllFileFilterUsed(false);
          }
          int option=fileChooser.showOpenDialog(drawArea.getTopLevelAncestor());
          if(option==JFileChooser.APPROVE_OPTION)
          {
            File f=fileChooser.getSelectedFile();
            BufferedInputStream input=null;
            try
            {
              input=new BufferedInputStream(
                  new FileInputStream(f));
              for(int i=0; i<bitmapFileHeader.length; i++)
                bitmapFileHeader[i]=input.read();
              char firstByte=(char)bitmapFileHeader[0];
              char secondByte=(char)bitmapFileHeader[1];
              if(firstByte!='B' || secondByte!='M')
              {
                drawArea.setMessage(NOT_BMP_FILE_ERROR_MSG);
                return;
              }
              int bfSize=getValueOfBytes(bitmapFileHeader, 2, 5);
              if(bfSize<=54)
              {
                drawArea.setMessage(NOT_BMP_FILE_ERROR_MSG);
                return;
              }
```

```
int bfOffBits=getValueOfBytes(bitmapFileHeader, 10, 13);
if(bfOffBits<54)
{
  drawArea.setMessage(NOT_BMP_FILE_ERROR_MSG);
  return;
}
for(int i=0; i<bitmapInfoHeader.length; i++)
  bitmapInfoHeader[i]=input.read();
int biSize=getValueOfBytes(bitmapInfoHeader, 0, 3);
int biWidth=getValueOfBytes(bitmapInfoHeader, 4, 7);
int biHeight=getValueOfBytes(bitmapInfoHeader, 8, 11);
int biPlanes=getValueOfBytes(bitmapInfoHeader, 12, 13);
int biBitCount=getValueOfBytes(bitmapInfoHeader, 14, 15);
if(biBitCount!=8 && biBitCount!=24)
{
  drawArea.setMessage(WRONG_BIT_COUNT_ERROR_MSG);
  return;
}
int biCompression=getValueOfBytes(bitmapInfoHeader, 16, 19);
int biSizeImage=getValueOfBytes(bitmapInfoHeader, 20, 23);
int biXPelsPerMeter=getValueOfBytes(bitmapInfoHeader, 24, 27);
int biYPelsPerMeter=getValueOfBytes(bitmapInfoHeader, 28, 31);
int biClrUsed=getValueOfBytes(bitmapInfoHeader, 32, 35);
int biClrImportant=getValueOfBytes(bitmapInfoHeader, 36, 39);
if(biBitCount==8)
{
  //read palette:
  int paletteSize=bfOffBits-54;
  int sizeWithoutReservedBytes=paletteSize-paletteSize/4;
  bitmapColorTable=new int[sizeWithoutReservedBytes];
  for(int i=0; i<bitmapColorTable.length; i++)
  {
    //convert BGR to RGB and skip Reserved Byte
    if(i%3==0)//blue
      bitmapColorTable[i+2]=input.read();
    if(i%3==1)//green
      bitmapColorTable[i]=input.read();
    if(i%3==2)//red
    {
      bitmapColorTable[i-2]=input.read();
      input.skip(1);
    }
  }
  //read bitmapBytes:
  int emptyBytes=4-biWidth%4;
  if(emptyBytes==4)
    emptyBytes=0;
  bitmapData=new int[biWidth*biHeight];
  for(int i=biHeight-1; i>=0; i--)//turns over the rows
  {
    for(int j=0; j<biWidth; j++)
      bitmapData[i*biWidth+j]=input.read();
    input.skip(emptyBytes);
  }
}
else
```

```
            {
              int emptyBytes=4-(biWidth*3)%4;
              if(emptyBytes==4)
                emptyBytes=0;
              bitmapData=new int[(biWidth*3)*biHeight];
              for(int i=biHeight-1; i>=0; i--)//turns over the rows
              {
                for(int j=0; j<(biWidth*3); j++)//converts BGR to RGB
                {
                  if(j%3==0)//blue
                    bitmapData[i*(biWidth*3)+j+2]=input.read();
                  if(j%3==1)//green
                    bitmapData[i*(biWidth*3)+j]=input.read();
                  if(j%3==2)//red
                    bitmapData[i*(biWidth*3)+j-2]=input.read();
                }
                input.skip(emptyBytes);
              }
            }
            drawArea.setImage(bitmapData, bitmapColorTable, biBitCount,
                biWidth, biHeight);
          }
          catch(FileNotFoundException ex)
          {
            drawArea.setMessage(FILE_NOT_FOUND_ERROR_MSG);
            return;
          }
          catch(IOException ex)
          {
            drawArea.setMessage(IO_ERROR_MSG);
            return;
          }
          try
          {
            input.close();
          }
          catch(IOException ex)
          {
            drawArea.setMessage(IO_ERROR_MSG);
            return;
          }
        }
      }
      else if(cmd.equals(FileMenu.EXIT_ITEM_CMD))
        System.exit(0);
    }
    private int getValueOfBytes(final int[] bytes, int startByte, int stopByte)
    {
      int value=0;
      int exp=0;
      for(int i=startByte; i<stopByte+1; i++)
        value+=bytes[i]*(int)Math.pow(2,exp++*8);
      return value;
    }
  }
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Filter1`: `Fenster 1 C:\...\Filter1.java`. Ausführen mit `Strg+2`.

Erproben Sie die hinzugefügte Funktionalität des Menüpunktes `Read`. Testen Sie außerdem den Accelerator (Tastenkombination als Alternative für einen Menüpunkt) `Strg+R`. Lesen Sie verschiedene `*.bmp` Dateien und erproben Sie das Programm.

Beachten Sie, das unser `JFileChooser` nur in der Lage ist, Files mit der Endung `.bmp` anzuzeigen. Versuchen Sie das Verhalten zu ändern, indem Sie die Zeile `fileChooser.setAcceptAllFileFilterUsed(false);` in der Methode `public void actionPerformed(ActionEvent ev)` der Klasse `MenuActionListener` auskommentieren.

Code für Rauschen

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

```
    private void setNoiseImage()
    {
      noiseImg=new BufferedImage(imgWidth, imgHeight,
          BufferedImage.TYPE_BYTE_INDEXED, grayICM);
      int[] filteredArray=new int[grayArray.length];
      int amplitude=256;
      int halfAmplitude=amplitude/2;
      int gray;
      for(int i=0; i<grayArray.length; i++)
      {
        int random=(int)(Math.random()*amplitude)-halfAmplitude;
        gray=grayArray[i];
        gray+=random;
        if(gray<0)
          gray=0;
        else if(gray>255)
          gray=255;
        filteredArray[i]=gray;
      }
      noiseImg.getRaster().setPixels(0, 0, imgWidth, imgHeight, filteredArray);
    }
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Filter1`: `Fenster 1 C:\...\Filter1.java`.
Ausführen mit `Strg+2`. Erproben Sie das Verrauschen des Bildes.

Code für Tiefpass

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

```
    private void setLowpassImage()
    {
      lowpassImg=new BufferedImage(imgWidth, imgHeight,
          BufferedImage.TYPE_BYTE_INDEXED, grayICM);
      int lowpassSize=11;
      int addMidWeight=0;
      int norm=lowpassSize*lowpassSize+addMidWeight;
      int d=lowpassSize/2;
      int sum;
      int[] filteredArray=new int[grayArray.length];
      Arrays.fill(filteredArray, 0);
      for(int v=d; v<imgHeight-d; v++)
        for(int h=d; h<imgWidth-d; h++)
        {
          sum=addMidWeight*grayArray[(v*imgWidth)+h];
          for(int y=v-d; y<=v+d; y++)
            for(int x=h-d; x<=h+d; x++)
              sum+=grayArray[(y*imgWidth)+x];
          filteredArray[(v*imgWidth)+h]=sum/norm;
        }
      lowpassImg.getRaster().setPixels(0, 0, imgWidth, imgHeight, filteredArray);
    }
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Filter1`: `Fenster 1 C:\...\Filter1.java`.
Ausführen mit `Strg+2`. Erproben Sie die Verwaschung des Bildes.

Code für die Hochpässe

Beenden Sie das Programm.
Wechseln Sie in TextPad zur Datei `DrawArea`: `Fenster 2 C:\...\DrawArea.java`
Version 4: Erweitern Sie die Methode `private void setHighpassImages()` der Klasse `DrawArea`, bis sie so aussieht:

```
    private void setHighpassImages()
    {
      highVerticalImg=new BufferedImage(imgWidth, imgHeight,
          BufferedImage.TYPE_BYTE_INDEXED, grayICM);
      highHorizontalImg=new BufferedImage(imgWidth, imgHeight,
          BufferedImage.TYPE_BYTE_INDEXED, grayICM);
      highGradientImg=new BufferedImage(imgWidth, imgHeight,
          BufferedImage.TYPE_BYTE_INDEXED, grayICM);
      int sumLeft, sumRight;
      int sumTop, sumBottom;
      int diff;
      int[][] filteredArray=new int[3][grayArray.length];
      for(int v=1; v<imgHeight-1; v++)
        for(int h=1; h<imgWidth-1; h++)
        {
          sumLeft=sumRight=sumTop=sumBottom=0;
          for(int j=-1; j<=1; j++)
          {
            sumLeft+=grayArray[(v+j)*imgWidth+(h-1)];
            sumRight+=grayArray[(v+j)*imgWidth+(h+1)];
            sumTop+=grayArray[(v-1)*imgWidth+(h+j)];
            sumBottom+=grayArray[(v+1)*imgWidth+(h+j)];
          }
          //high vertical
          diff=Math.abs(sumLeft-sumRight);
          if(diff>255)
            filteredArray[0][(v*imgWidth)+h]=255;
          else
            filteredArray[0][(v*imgWidth)+h]=diff;
          //high horizontal
          diff=Math.abs(sumTop-sumBottom);
          if(diff>255)
            filteredArray[1][(v*imgWidth)+h]=255;
          else
            filteredArray[1][(v*imgWidth)+h]=diff;
          //high gradient
          double diffH=sumLeft-sumRight;
          double diffV=sumTop-sumBottom;
          diff=(int)(0.5+Math.sqrt(diffH*diffH+diffV*diffV));
          if(diff>255)
            filteredArray[2][(v*imgWidth)+h]=255;
          else
            filteredArray[2][(v*imgWidth)+h]=diff;
        }
      highVerticalImg.getRaster().setPixels(0, 0, imgWidth, imgHeight,
          filteredArray[0]);
      highHorizontalImg.getRaster().setPixels(0, 0, imgWidth, imgHeight,
          filteredArray[1]);
      highGradientImg.getRaster().setPixels(0, 0, imgWidth, imgHeight,
          filteredArray[2]);
    }
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Filter1`: `Fenster 1 C:\...\Filter1.java`.
Ausführen mit `Strg+2`. Erproben Sie die Hochpassfilter.

Experimente

(1) Variieren Sie die `amplitude` zwischen 10 und 1000.
(2) Variieren Sie `lowpassSize` zwischen 3 und 100.
(3) Variieren Sie `addMidWeight` zwischen 0 und 1000.
(4) Nehmen Sie das Noise-Bild als Eingang für den Lowpass (neuer `grayArray2` notwendig).
(5) Nehmen Sie das Lowpassbild als Eingang für den Highpass (noch ein neuer `grayArray3` notwendig).

Beispielbilder

Das Programm sollte alle 8bit bzw. 24bit Bitmaps lesen und anzeigen. Falls Sie keine `*.bmp` Dateien auf Ihrer Harddisk finden, benutzen Sie folgende Beispielbilder:
Download: Butterfly.bmp 217 kB 24Bit-TrueColor-Bild
Download: Madonna.bmp 18 kB 8Bit-Grauwert-Bild
Download: Lena256.bmp 66 kB 8Bit-Grauwert-Bild
Download: Lena512.bmp 258 kB 8Bit-Grauwert-Bild
Download: Angiography.bmp 66 kB 8Bit-Grauwert-Bild

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 zur Klasse `java.util.Arrays` an und verstehen Sie in diesem Zusammenhang die Zeile `Arrays.fill(filteredArray, 0);`.
Erhö;hen sie die Anzahl der Bilder durch anzeigen mehrerer Rauschbilder mit verschiedener `amplitude` und/oder mehrerer Tiefpassbilder mit verschiedenem `lowpassSize` und `addMidWeight`. Passen Sie Positionen und Hö;hen der Bilder an die Anzahl an.
Erproben Sie andere Hochpassfilter mit schräg über Eck stehenden (+45 Grad -45 Grad) Nachbarn.
Erfinden und erproben Sie neue Varianten des Programms in Form von neuen Projekten `filter2`, `filter3` usw. nach obigem Muster.