Course IPJava: Image Processing with Java
Chapter C1: The Bitmap Project

Projekt bitmap1 mit leerem Fenster

Beschreibung für: Java Version 6, Update 6 und die Eclipse Platform Version 3.3.2.
Starten Sie Eclipse und wählen im Menu:
`File → New → Java Project → Project name: bitmap1 → Finish`
`File → New → Class → Name: bitmap1 → Finish`
Ersetzen Sie den Code von `bitmap1.java` durch:

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

public class Bitmap1
{
  public static void main(String[] args)
  {
    JFrame frame=new JFrame("Bitmap1");
    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);
  }
}
```

`Run → Run`. Das Ergebnis ist ein leeres weißes Fenster mit Überschrift. Schließen Sie dieses Fenster.

Zeichenfläche

Beenden Sie das Programm.
Schreiben Sie folgende neuen Klassen:

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

public class DrawArea extends JComponent
{
  BufferedImage image;
  int imgWidth, imgHeight;
  int newImgWidth, newImgHeight;
  int myWidth, myHeight, fontHeight=0;
  int click=-1;
  final int MAX_CLICKS=1;
  String[] imgInfo;
  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;
      }
    });
    addMouseListener(new MouseAdapter()
    {
      public void mousePressed(MouseEvent ev){}
    });
  }
  public void paintComponent(Graphics g)
  {
    super.paintComponent(g);
    if(fontHeight==0)
      fontHeight=g.getFont().getSize();
    switch(click)
    {
      default:
        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, String[] info)
  {
  }
  public void setMessage(String message)
  {
  }

  class AnimationPerformer implements ActionListener
  {
    public void actionPerformed(ActionEvent ev)
    {
    }
  }
}
```

Ergänzen Sie die Methode `public static void main(String[] args)` von `Bitmap1`, dass sie so aussieht:

```
    public static void main(String[] args)
    {
      JFrame frame=new JFrame("Bitmap (Bilder lesen und anzeigen)");
      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);
    }
```

Menüleiste

Beenden Sie das Programm.
Drei neue 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);
    }
  }
}
```

Zuweisen einer Menüleiste an `JFrame`:
Wechseln Sie dazu zur Datei `Bitmap1` im Fenster `C:\...\Bitmap1.java`
Fügen Sie die folgende Zeile als 3. Anweisung (unter die Instantiierung der `drawArea`) in die Methode `public static void main(String[] args)` von `Bitmap1` ein:

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

Erproben Sie den neuen Menüpunkt `Exit`.

Bild lesen und anzeigen

Beenden Sie das Programm.
Wechseln Sie zur Datei `DrawArea`: Fenster `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, String[] info)
{
  imgWidth=width;
  imgHeight=height;
  imgInfo=info;
  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);
  }
  else
  {
    image=new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    image.getRaster().setPixels(0, 0, width, height, img);
  }
  click=0;
  repaint();
}
```

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

```
public void setMessage(String message)
{
  this.message=message;
  click=-1;
  repaint();
}
```

Erweitern Sie nun die Methode `public void paintComponent(Graphics g)` derselben Klasse, bis sie so aussieht:

```
public void paintComponent(Graphics g)
{
  super.paintComponent(g);
  if(fontHeight==0)
    fontHeight=g.getFont().getSize();
  switch(click)
  {
    case 0://Information
      g.setColor(Color.RED);
      g.drawString("Click on left mouse button!", 0, fontHeight);
      g.setColor(Color.BLACK);
      for(int i=0; i<imgInfo.length; i++)
        g.drawString(imgInfo[i], 0, (i+3)*(fontHeight+2));
      break;
    case 1://Raw display
      g.drawImage(image, 0, 0, this);
      g.setColor(Color.RED);
      g.drawString("Click on left mouse button!", 0, fontHeight);
      g.setColor(Color.BLACK);
      break;
    case 2://Center and fit image to DrawArea size
      g.setColor(Color.RED);
      g.drawString("Change Window Size. Click on left mouse button!",
          0, fontHeight);
      g.setColor(Color.BLACK);
      break;
```

```
      case 3://Horizontal stretch
        g.setColor(Color.RED);
        g.drawString("Change Window Size. Click on left mouse button!",
            0, fontHeight);
        g.setColor(Color.BLACK);
        break;
      case 4://Vertical stretch
        g.setColor(Color.RED);
        g.drawString("Change Window Size. Click on left mouse button!",
            0, fontHeight);
        g.setColor(Color.BLACK);
        break;
      case 5://Full size
        g.setColor(Color.RED);
        g.drawString("Change Window Size. Click on left mouse button!",
            0, fontHeight);
        g.setColor(Color.BLACK);
        break;
      case 6://Mirror
        g.setColor(Color.RED);
        g.drawString("Change Window Size. Click on left mouse button!",
            0, fontHeight);
        g.setColor(Color.BLACK);
        break;
      case 7://Zoom animation
        break;
      default:
        g.setColor(Color.RED);
        g.drawString(message, 0, fontHeight);
        g.setColor(Color.BLACK);
    }
  }
```

Ergänzen Sie die Methode `public void mousePressed(MouseEvent ev)` des anonymen `MouseListeners` im Constructor von `DrawArea`, bis sie so aussieht:

```
  public void mousePressed(MouseEvent ev)
  {
    if(click==-1)
      return;
    if(click==MAX_CLICKS)
      click=0;
    else
      click++;
    repaint();
  }
```

Zum Lesen 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 zur Datei `MenuBar`: Fenster `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 Bitmap-File (.bmp).";
    static final String WRONG_BIT_COUNT_ERROR_MSG=
      "Choose a 8 or 24 bit Windows Bitmap-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 Bitmap (*.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);
  }
}
String[] info=
{
  "BITMAPFILEHEADER:",
  "-----------------",
  "firstByte: "+firstByte,
  "secondByte: "+secondByte,
  "bfSize: "+bfSize+" bytes",
  "bfOffBits: "+bfOffBits,
  "",
```

```
                "BITMAPINFOHEADER:",
                "-----------------",
                "biSize: "+biSize+" bytes",
                "biWidth: "+biWidth+" pixel",
                "biHeight: "+biHeight+" pixel",
                "biPlanes: "+biPlanes,
                "biBitCount: "+biBitCount+" bit color depth",
                "biCompression: "+biCompression,
                "biSizeImage: "+biSizeImage+" bytes (readed), "+
                    (biBitCount==8?
                    (biWidth*biHeight)+" bytes (calculated: biWidth*biHeight)":
                    (biWidth*3*biHeight)+" bytes (calculated: "+
                    "(biWidth*3)*biHeight)"),
                "biXPelsPerMeter: "+biXPelsPerMeter+" pixels per meter, "+
                    biXPelsPerMeter/(100/2.54)+" dpi",
                "biYPelsPerMeter: "+biYPelsPerMeter+" pixels per meter, "+
                    biXPelsPerMeter/(100/2.54)+" dpi",
                "biClrUsed: "+biClrUsed+" colors in palette",
                "biClrImportant: "+biClrImportant
              };
              drawArea.setImage(bitmapData, bitmapColorTable, biBitCount,
                  biWidth, biHeight, info);
            }
            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;
      }
    }
```

Erproben Sie den Menüpunkt `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.

Zentrieren und auf Zeichenfläche einpassen

Wechseln Sie zur Datei `DrawArea` im Fenster `C:\...\DrawArea.java`.
Version 2: Schreiben Sie in der Methode `public void paintComponent(Graphics g)` unterhalb der Zeile `case 2://Center and fit image to DrawArea size` folgende weitere Zeilen:

```
int w=imgWidth, h=imgHeight;
double x=myWidth, y=myHeight;
if(y/h<x/w)
{
  w=(int)((y/h)*w);
  h=myHeight;
}
else
{
  h=(int)((x/w)*h);
  w=myWidth;
}
g.drawImage(image, myWidth/2-w/2, myHeight/2-h/2, w, h, this);
```

Erhöhen sie die Kostante `MAX_CLICKS` in `DrawArea`:

```
static final int MAX_CLICKS=2;
```

Erproben Sie die Zentrierung.

Horizontal strecken

Wechseln Sie zur Datei `DrawArea` im Fenster `C:\...\DrawArea.java`.
Version 3: Schreiben Sie in der Methode `public void paintComponent(Graphics g)` unterhalb der Zeile `case 3://Horizontal stretch` folgende weitere Zeilen:

```
g.drawImage(image, 0, myHeight/2-myHeight/10, myWidth, myHeight/5, this);
```

Erhöhen sie außerdem die Kostante `MAX_CLICKS` in `DrawArea`:

```
static final int MAX_CLICKS=3;
```

Erproben Sie die Streckung.

Vertikal strecken

Wechseln Sie zur Datei `DrawArea` in Fenster `C:\...\DrawArea.java`.
Version 4: Schreiben Sie in der Methode `public void paintComponent(Graphics g)` unterhalb der Zeile `case 4://Vertical stretch` folgende weitere Zeilen:

```
g.drawImage(image, myWidth/2-myWidth/10, 0, myWidth/5, myHeight, this);
```

Erhöhen sie außerdem die Kostante `MAX_CLICKS` in `DrawArea`:

```
static final int MAX_CLICKS=4;
```

Erproben Sie die Streckung.

Maximale Größe

Wechseln Sie zur Datei `DrawArea` im Fenster `C:\...\DrawArea.java`.
Version 5: Schreiben Sie in der Methode `public void paintComponent(Graphics g)` unterhalb der Zeile `case 5://Full size` folgende weitere Zeilen:

```
g.drawImage(image, 0, 0, myWidth, myHeight, this);
```

Erhöhen sie außerdem die Kostante `MAX_CLICKS` in `DrawArea`:

```
static final int MAX_CLICKS=5;
```

Ziehen Sie am Fensterrand, um die Vergrößerung zu erproben.

Spiegeln

Wechseln Sie zur Datei `DrawArea` im Fenster `C:\...\DrawArea.java`
Version 6: Schreiben Sie in der Methode `public void paintComponent(Graphics g)` unterhalb der Zeile `case 6://Mirror` folgende weitere Zeilen:

```
g.drawImage(image, myWidth/2, myHeight/2, myWidth/2, myHeight/2, this);
g.drawImage(image, myWidth/2, myHeight/2, myWidth/2,-myHeight/2, this);
g.drawImage(image, myWidth/2, myHeight/2,-myWidth/2, myHeight/2, this);
g.drawImage(image, myWidth/2, myHeight/2,-myWidth/2,-myHeight/2, this);
```

Erhöhen sie außerdem die Kostante `MAX_CLICKS` in `DrawArea`:

```
static final int MAX_CLICKS=6;
```

Erproben Sie die Spiegelungen.

Zoomanimation

Beenden Sie das Programm.
Version 7: Um das geladene Bild zu animieren, benötigen wir einen `Timer`. Der `Timer` bekommt beim Konstruktoraufruf einen `ActionListener` übergeben, dessen Methode `public void actionPerformed(ActionEvent ev)` ausgeführt wird, wenn der Timer gestartet wird und die solange wiederholt wird, bis der Timer wieder gestoppt wird.
Wechseln Sie zur Datei `DrawArea` im Fenster `C:\...\DrawArea.java`
Schreiben Sie folgende Zeilen in den noch leeren Rumpf der innere Klasse `AnimationPerformer`, dass sie so aussieht:

```
    class AnimationPerformer implements ActionListener
    {
      int x;
      int y;
      int step=0;
      Graphics g;
      public void actionPerformed(ActionEvent ev)
      {
        x=myWidth/20;
        y=myHeight/20;
        g=getGraphics();
        g.drawImage(image, 0, 0, myWidth-step*x, myHeight-step*y, DrawArea.this);
        step++;
        if(step>19)
        {
          g.setColor(Color.RED);
          g.drawString("Change Window Size. Click on left mouse button!",
              0, fontHeight);
          g.setColor(Color.BLACK);
          animationTimer.stop();
          step=0;
        }
      }
      public void setStep(int s)
      {
        step=s;
      }
    }
```

Schreiben Sie folgende Zeilen unter die Zeile `String message="Open a .bmp file";` aber noch vor dem Konstruktor `public DrawArea(){...}`

```
    AnimationPerformer animation;
    Timer animationTimer;
```

Schreiben Sie folgende Zeilen im Konstruktor von `DrawArea` vor der Anweisung `Dimension size=getSize();`:

```
    animation=new AnimationPerformer();
    animationTimer=new Timer(200, animation);
```

Erweitern Sie die Methode `public void mousePressed(MouseEvent ev)` der anonymen `MouseListener` Klasse, dass sie so aussieht:

```
    public void mousePressed(MouseEvent ev)
    { if(click==-1) return;
      if(click==MAX_CLICKS) click=0;
      else click++;
      if(click==0 && animationTimer.isRunning())
      { animationTimer.stop();
        animation.setStep(0);
      }
      repaint();
    }
```

Schreiben Sie in der Methode `public void paintComponent(Graphics g)` unterhalb der Zeile `case 7://Zoom animation` folgende weitere Zeilen:

```
    if(animationTimer.isRunning())
    {
      animationTimer.stop();
      animation.setStep(0);
    }
    animationTimer.start();
```

Erhöhen sie außerdem die Kostante `MAX_CLICKS` in `DrawArea`:

```
    static final int MAX_CLICKS=7;
```

Erproben Sie die Zoom-Animation.

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 Dokumentation zu den Klassen: `javax.swing.Timer`, `java.io.BufferedInputStream`, `java.awt.image.BufferedImage`, `java.awt.image.IndexColorModel`.
Vollziehen Sie die Methode `public void actionPerformed(ActionEvent ev)` in der Klasse `MenuActionListener` nach.
Wichtig: Die Rolle der Methode
`private int getValueOfBytes(final int[] bytes, int startByte, int stopByte)`
in der Klasse `MenuActionListener`.
Verändern Sie die Bildposition in `case 2`.
Verändern Sie die Streckungen in `case 3` und `case 4`.
Verändern Sie die Schrittweite und die Zoomecke in `case 5`.
Verändern Sie die Geschwindigkeit des `animationTimer`.

Erfinden und erproben Sie neue Varianten des Programms in Form von neuen Projekten `bitmap2`, `bitmap3` usw. nach obigem Muster.