

温州大学计算机与人工智能学院

Java程序设计（17网工）实验报告

实验名称	展示图片				
班 级	18电科2	姓 名	方涛涛	学 号	18211110208
实验地点		实验时间	2020-12-27,16:36:03	指导老师	

一、问题编号：

2706

地址：<http://10.132.254.54/problem/2706/>

二、问题描述：

中间用Label来展示图片。

上面是工具栏，有如下按钮：

（1）打开图片；（2）显示上一张图片；（3）显示下一张图片；（4）放大图片；（5）缩小图片。

如下图所示：

三、输入说明：

四、输出说明：

五、输入样例：

六、输出样例：

七、解答内容：

所用语言：

源代码：

```
001. import java.awt.Color;
002. import java.awt.Image;
003. import java.awt.event.ActionEvent;
004. import java.awt.event.ActionListener;
005. import java.awt.event.MouseEvent;
006. import java.awt.event.MouseListener;
007. import java.awt.event.MouseWheelEvent;
008. import java.awt.event.MouseWheelListener;
009. import java.awt.image.BufferedImage;
010. import java.io.File;
011. import java.io.FileFilter;
012. import java.io.IOException;
013. import java.util.ArrayList;
014. import java.util.List;
015.
016. import javax.imageio.ImageIO;
017. import javax.swing.ImageIcon;
018. import javax.swing.JFileChooser;
019. import javax.swing.JFrame;
020. import javax.swing.JLabel;
021. import javax.swing.JMenu;
```

```

022. import javax.swing.JMenuBar;
023. import javax.swing.JMenuItem;
024. import javax.swing.JPanel;
025. import javax.swing.JScrollPane;
026.
027.
028. public class Main extends JFrame implements ActionListener, MouseListener, MouseWheelListener {
029.     JMenuBar jmb;
030.     JMenu jm1, jm2, jm3;
031.     JMenuItem[] jmi = new JMenuItem[7];
032.     JLabel jl, jl1;
033.     JPanel jp;
034.     JScrollPane jsp;
035.     ImageIcon imgIco, showii;
036.     File f;
037.     JFileChooser jfc;
038.     float width;
039.     float height;
040.
041.     public static void main(String[] args) {
042.         Main mf = new Main();
043.         mf.init();
044.     }
045.
046.     public void init() {
047.         jfc = new JFileChooser();
048.         jmb = new JMenuBar();
049.         jm1 = new JMenu("文件");
050.         jm2 = new JMenu("编辑");
051.         jmi[0] = new JMenuItem("打开");
052.         jmi[0].addActionListener(this);
053.         jmi[1] = new JMenuItem("另存为");
054.         jmi[1].addActionListener(this);
055.         jmi[5] = new JMenuItem("上一张");
056.         jmi[5].addActionListener(this);
057.         jmi[6] = new JMenuItem("下一张");
058.         jmi[6].addActionListener(this);
059.         jmi[2] = new JMenuItem("放大");
060.         jmi[2].addActionListener(this);
061.         jmi[3] = new JMenuItem("缩小");
062.         jmi[3].addActionListener(this);
063.         jmi[4] = new JMenuItem("原图");
064.         jmi[4].addActionListener(this);
065.         jm1.add(jmi[0]);
066.         jm1.add(jmi[1]);
067.         jm1.add(jmi[5]);
068.         jm1.add(jmi[6]);
069.         jm2.add(jmi[2]);
070.         jm2.add(jmi[3]);
071.         jm2.add(jmi[4]);
072.         jmb.add(jm1);
073.         jmb.add(jm2);
074.
075.         jl = new JLabel("请选择图片", JLabel.CENTER);
076.         jl.setForeground(Color.gray);
077.         jl.addMouseWheelListener(this);
078.         jl.addMouseListener(this);
079.         jsp = new JScrollPane(jl);
080.
081.         this.setJMenuBar(jmb);
082.         this.add(jsp);
083.         this.setTitle("Photo");
084.         this.setSize(700, 500);
085.         this.setLocation(300, 200);
086.         this.setVisible(true);
087.         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
088.     }
089.
090.     public void ShowImg() {
091.         width = (float) (imgIco.getIconWidth() * 0.09);
092.         height = (float) (imgIco.getIconHeight() * 0.09);
093.         showii = new ImageIcon(
094.             imgIco.getImage().getScaledInstance((int) width, (int) height, 0));
095.         this.jl.setText("");
096.         this.jl.setIcon(showii);
097.     }
098.
099.     public void Zoom(int flag) {
100.         if ((width < 5000 && height < 4000) && (width > 5 && height > 4)) {
101.             if (flag == 0) {
102.                 width *= 1.3;
103.                 height *= 1.3;

```

```

104.         } else if (flag == 1) {
105.             width *= 0.7;
106.             height *= 0.7;
107.         } else if (flag == 2) {
108.             width = (float) (imgIco.getIconWidth() * 0.09);
109.             height = (float) (imgIco.getIconHeight() * 0.09);
110.         }
111.         showii = new ImageIcon(imgIco.getImage().getScaledInstance((int) width, (int) height, 0));
112.         this.jl.setIcon(showii);
113.     } else if (width > 5000 || height > 4000) {
114.         width *= 0.8;
115.         height *= 0.8;
116.     } else if (width < 5 || height < 3) {
117.         width *= 1.2;
118.         height *= 1.2;
119.     }
120. }
121.
122. public void Save() {
123.     Image img = showii.getImage();
124.     try {
125.
126.         BufferedImage choosed = ImageIO.read(f);
127.         BufferedImage bi = new BufferedImage((int) width, (int) height, choosed.getType());
128.         bi.getGraphics().drawImage(img, 0, 0, null);
129.         int result = jfc.showSaveDialog(this);
130.         File save = jfc.getSelectedFile();
131.         if (result == 0 && save != null) {
132.             String[] n = save.getName().split("\\.");
133.             ImageIO.write(bi, (n.length == 1 ? "gif" : n[1]), save);
134.         }
135.     } catch (IOException e) {
136.         e.printStackTrace();
137.     }
138. }
139.
140.
141. @Override
142. public void actionPerformed(ActionEvent e) {
143.     if (e.getSource() == jmi[0] || e.getSource() == jl) {
144.         jfc.setCurrentDirectory(jfc.getCurrentDirectory());
145.         int result = jfc.showOpenDialog(this);
146.         f = jfc.getSelectedFile();
147.         if (result == 0 && f != null) {
148.             mytools.addFile(f);
149.             new getFileList(jfc.getCurrentDirectory()).start();
150.             imgIco = new ImageIcon(f.getPath());
151.             ShowImg();
152.         }
153.     } else if (f != null && e.getSource() == jmi[2]) {
154.         Zoom(0);
155.     } else if (f != null && e.getSource() == jmi[3]) {
156.         Zoom(1);
157.     } else if (f != null && e.getSource() == jmi[4]) {
158.         Zoom(2);
159.     } else if (f != null && e.getSource() == jmi[5]) {
160.         f = mytools.getLast(f);
161.         imgIco = new ImageIcon(f.getPath());
162.         ShowImg();
163.     } else if (f != null && e.getSource() == jmi[6]) {
164.         f = mytools.getNext(f);
165.         imgIco = new ImageIcon(f.getPath());
166.         ShowImg();
167.     } else if (f != null && e.getSource() == jmi[1]) {
168.         Save();
169.     }
170. }
171.
172. @Override
173. public void mouseWheelMoved(MouseWheelEvent e) {
174.     if (f != null) {
175.         if (e.getWheelRotation() == 1) {
176.             Zoom(0);
177.         } else if (e.getWheelRotation() == -1) {
178.             Zoom(1);
179.         }
180.     }
181. }
182.
183. @Override
184. public void mouseClicked(MouseEvent e) {
185.     if (f != null) {

```

```

186.         if (e.getClickCount() == 1 && e.getX() < 200) {
187.             f = mytools.getLast(f);
188.             imgIco = new ImageIcon(f.getPath());
189.             ShowImg();
190.         } else if (e.getClickCount() == 1 && e.getX() > 500) {
191.             f = mytools.getNext(f);
192.             imgIco = new ImageIcon(f.getPath());
193.             ShowImg();
194.         }
195.     }
196. }
197.
198. @Override
199. public void mousePressed(MouseEvent e) {
200.
201. }
202.
203. @Override
204. public void mouseReleased(MouseEvent e) {
205.
206. }
207.
208. @Override
209. public void mouseEntered(MouseEvent e) {
210.
211. }
212.
213. @Override
214. public void mouseExited(MouseEvent e) {
215.
216. }
217. }
218.
219. class getFileList extends Thread {
220.     File CurrentDirectory;
221.
222.     public getFileList(File f) {
223.         this.CurrentDirectory = f;
224.     }
225.
226.     public void run() {
227.         File[] cd = CurrentDirectory.listFiles(mytools.myFilenameFilter());
228.         for (int i = 0; i < cd.length; i++) {
229.             mytools.addFile(cd[i]);
230.         }
231.     }
232. }
233.
234. class mytools {
235.     public static List<File> fileBuffer = new ArrayList<File>();
236.
237.     public static void addFile(File f) {
238.         if (!fileBuffer.contains(f))
239.             fileBuffer.add(f);
240.     }
241.
242.     public static File getNext(File f) {
243.         if (fileBuffer.indexOf(f) + 1 < fileBuffer.size())
244.             return fileBuffer.get(fileBuffer.indexOf(f) + 1);
245.         else
246.             return fileBuffer.get(0);
247.     }
248.
249.     public static File getLast(File f) {
250.         if (fileBuffer.indexOf(f) - 1 >= 0)
251.             return fileBuffer.get(fileBuffer.indexOf(f) - 1);
252.         else
253.             return fileBuffer.get(fileBuffer.size() - 1);
254.     }
255.
256.
257.     public static FilenameFilter myFilenameFilter() {
258.         FilenameFilter ff = new FilenameFilter() {
259.             @Override
260.             public boolean accept(File dir, String name) {
261.                 if (name.endsWith(".jpg") || name.endsWith(".jpeg")
262.                     || name.endsWith(".gif") || name.endsWith(".png")
263.                     || name.endsWith(".bmp"))
264.                     return true;
265.                 return false;
266.             }
267.         };

```

```
268. |  
269. |  
270. | }  
    }  
    return ff;
```

八、判题结果

RE - 运行错误

判题结果补充说明:

test id:6223,result:RE, usedtime:152MS, usedmem:3364KB,score:100 Exception in thread "main" java.awt.HeadlessException: No X11 DISPLAY variable was set, but this program performed an operation which requires it. at
java.awt.GraphicsEnvironment.checkHeadless(GraphicsEnvironment.java:173) at java.awt.Window.<init>(Window.java:547) at
java.awt.Frame.<init>(Frame.java:419) at java.awt.Frame.<init>(Frame.java:384) at javax.swing.JFrame.<init>(JFrame.java:174) at Main.
<init>(Main.java:28) at Main.main(Main.java:42)