TRƯỜNG ĐẠI HỌC FPT

Lecture 05
Two Dimensional Graphics
Part 3

Working with Images

Reference: Java-Tutorials/tutorial-2015/2d/images/index.html

Contents

1- Introduction

- There are a number of common tasks when working with images.
  - Loading an external GIF, PNG JPEG image format file into the internal image representation used by Java 2D.
  - Directly creating a Java 2D image and rendering to it.
  - Drawing the contents of a Java 2D image on to a drawing surface.
  - Saving the contents of a Java 2D image to an external GIF, PNG, or JPEG image file.

Introduction…

- The are two main classes that you must learn about to work with images:
  - The java.awt.Image class is the superclass that represents graphical images as rectangular arrays of pixels.
  - The java.awt.image.BufferedImage class, which extends the Image class to allow the application to operate directly with image data (for example, retrieving or setting up the pixel color). Applications can directly construct instances of this class.
- The BufferedImage class is a cornerstone of the Java 2D immediate-mode imaging API. It manages the image in memory and provides methods for storing, interpreting, and obtaining pixel data.
  Since BufferedImage is a subclass of Image it can be rendered by the Graphics and Graphics2D methods that accept an Image parameter.
- A BufferedImage is essentially an Image with an accessible data buffer. It is therefore more efficient to work directly with BufferedImage.
  A BufferedImage has a *ColorModel* and a *Raster* of image data. The ColorModel provides a color interpretation of the image's pixel data.

Introduction…

- The Raster performs the following functions:
  - Represents the rectangular coordinates of the image
  - Maintains image data in memory
  - Provides a mechanism for creating multiple subimages from a single image data buffer
  - Provides methods for accessing specific pixels within the image

2- The java.awt.Image Class

- public abstract class **Image** extends Object
- The abstract class Image is the superclass of all classes that represent graphical images. The image must be obtained in a platform-specific manner.
- https://docs.oracle.com/javase/8/docs/api/java/awt/Image.html
- Constructor: Image(void)
- Common Methods

3- The java.awt.image.BufferedImage

- public class **BufferedImage** extends Image implements WritableRenderedImage, Transparency
- The BufferedImage subclass describes an Image with an accessible buffer of image data. A BufferedImage is comprised of a ColorModel and a Raster of image data. The number and types of bands in the SampleModel of the Raster must match the number and types required by the ColorModel to represent its color and alpha components. AllBufferedImage objects have an upper left corner coordinate of (0, 0). Any Raster used to construct a BufferedImage must therefore have minX=0 and minY=0.This class relies on the data fetching and setting methods of Raster, and on the color characterization methods of ColorModel.

4- Reading/Loading an Image

- Loading a local image:

```
BufferedImage img = null;
try {
   img = ImageIO.read(new File("strawberry.jpg"));
} catch (IOException e) { }
```

Package javax.swing.imageio.ImageIO contains methods for reading/writing image files

- Loading a remote image:

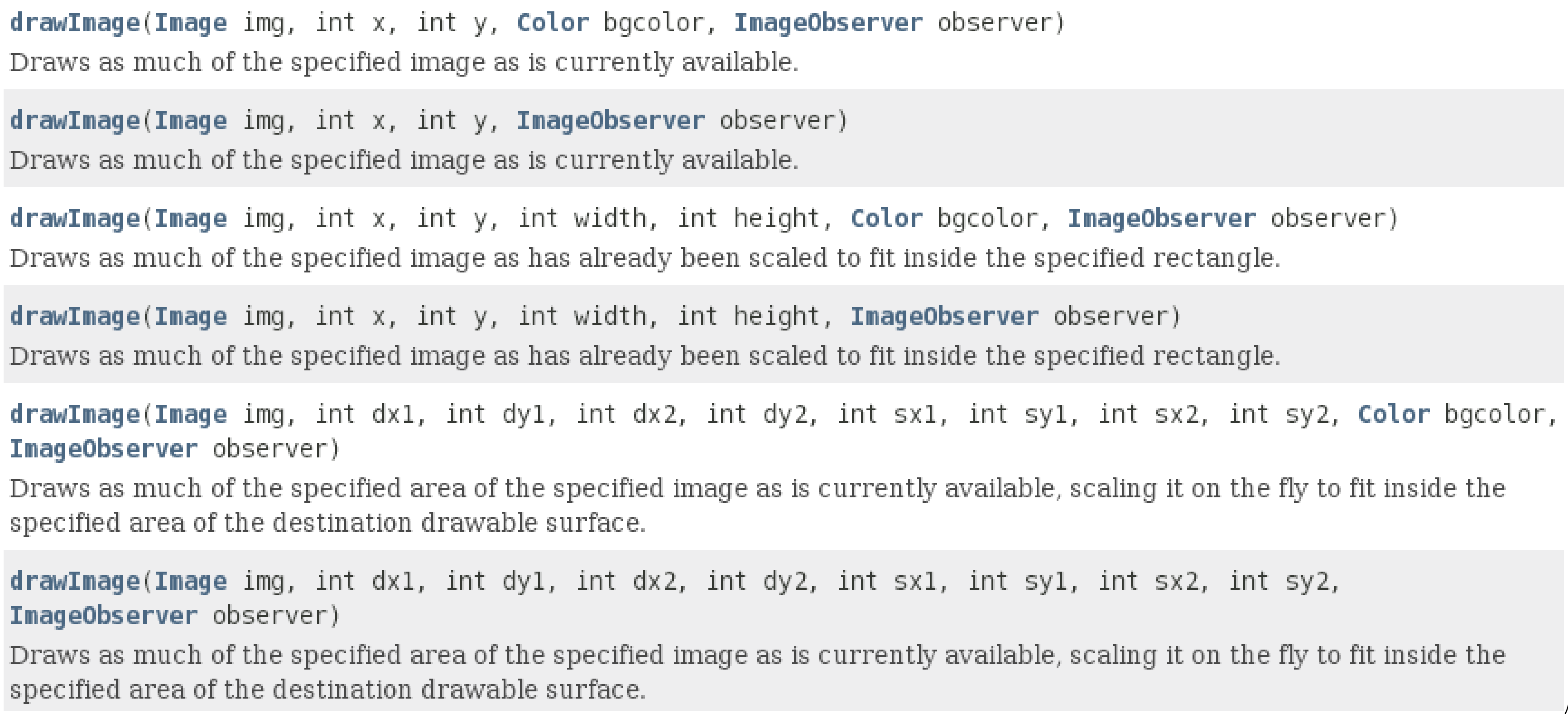
```
try {
   URL url = new
   URL(getCodeBase(),"examples/strawberry.jpg");
   img = ImageIO.read(url);
} catch (IOException e) { }
```

The method getCodeBase() will create a connection to server to get the resource

You can use the class java.awt.Toolkit to read an image file as following"
Image Img =Toolkit.getDefaultToolkit().getImage(FileName);

5- Drawing an Image

- Use the method drawImage(…) of the Graphics class
- boolean Graphics.drawImage(image, x, y,…);
- x, y: left corner of the drawing area
- The observer parameter notifies the application of updates to an image that is loaded asynchronously. The observer parameter is not frequently used directly and is not needed for the BufferedImage class, so it usually is null.



```
drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer)
```
Draws as much of the specified image as is currently available.

```
drawImage(Image img, int x, int y, ImageObserver observer)
```
Draws as much of the specified image as is currently available.

```
drawImage(Image img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer)
```
Draws as much of the specified image as has already been scaled to fit inside the specified rectangle.

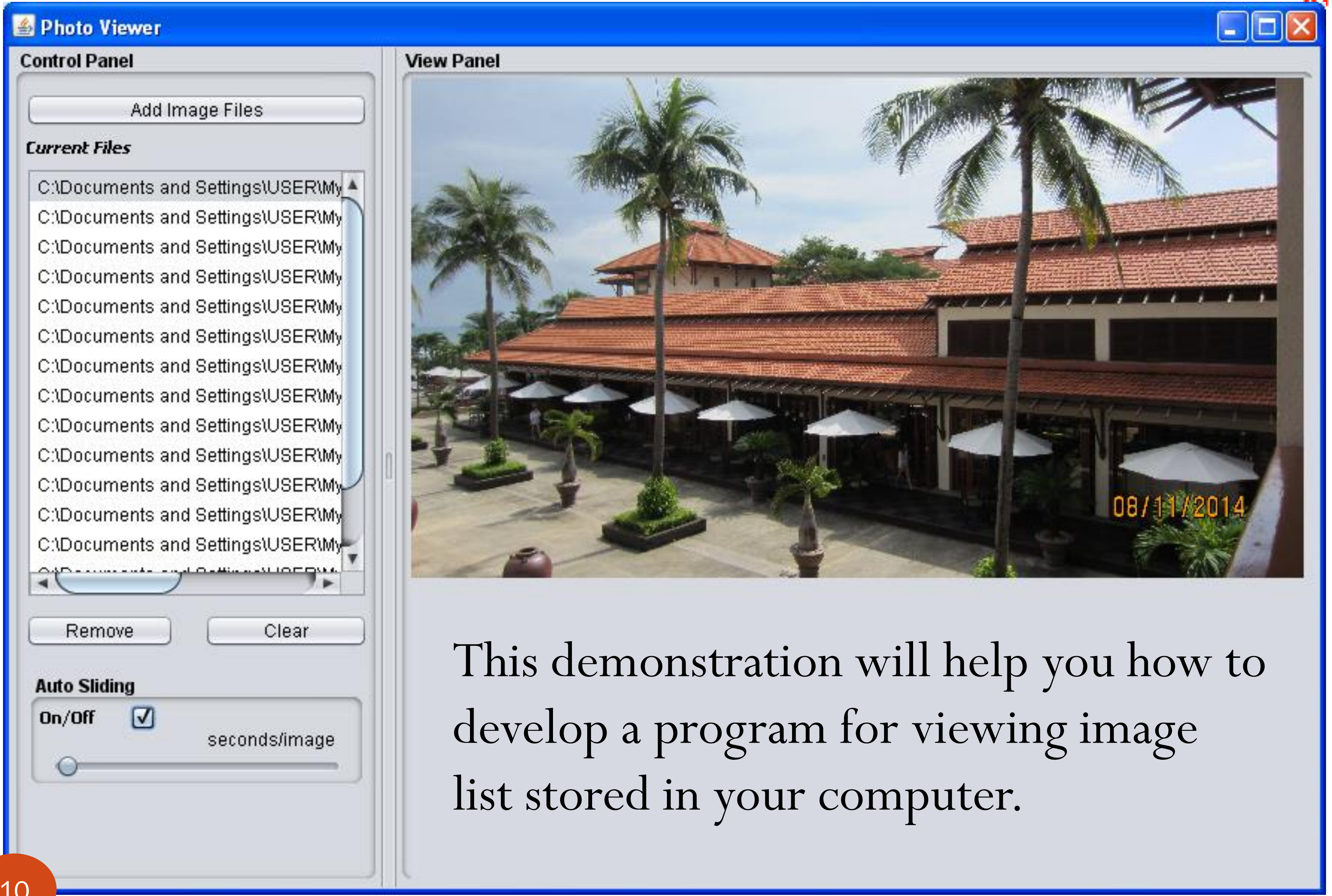
```
drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)
```
Draws as much of the specified image as has already been scaled to fit inside the specified rectangle.

```
drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color bgcolor,
ImageObserver observer)
```
Draws as much of the specified area of the specified image as is currently available, scaling it on the fly to fit inside the specified area of the destination drawable surface.

```
drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2,
ImageObserver observer)
```
Draws as much of the specified area of the specified image as is currently available, scaling it on the fly to fit inside the specified area of the destination drawable surface.

FPT Fpt University
TRƯỜNG ĐẠI HỌC FPT

Demonstration



C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My

This demonstration will help you how to develop a program for viewing image list stored in your computer.

Demonstration…

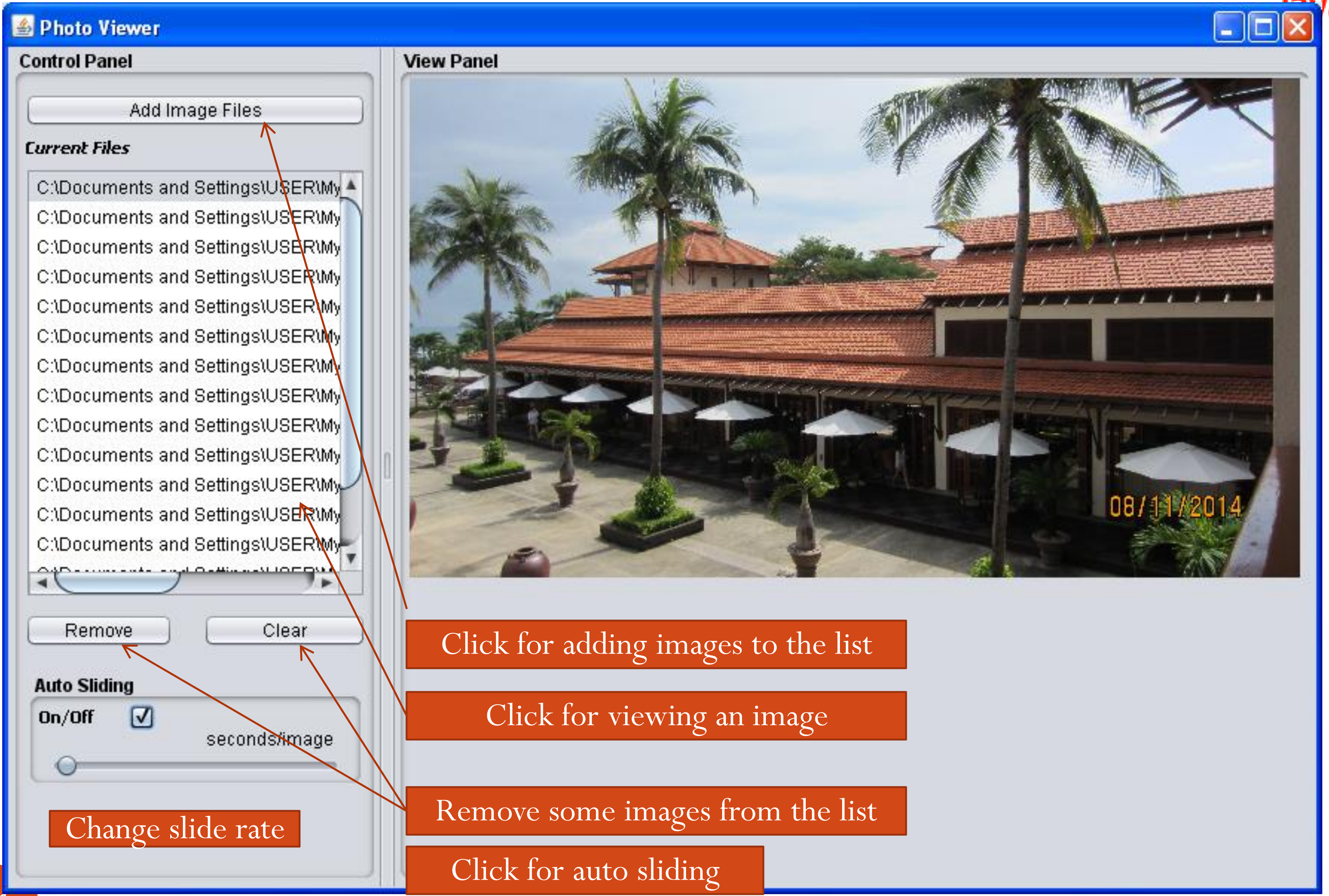


Photo Viewer
Control Panel
Add Image Files
Current Files
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
C:\Documents and Settings\USER\My
Remove
Clear
Auto Sliding
On/Off
seconds/image
View Panel
08/11/2014

Click for adding images to the list

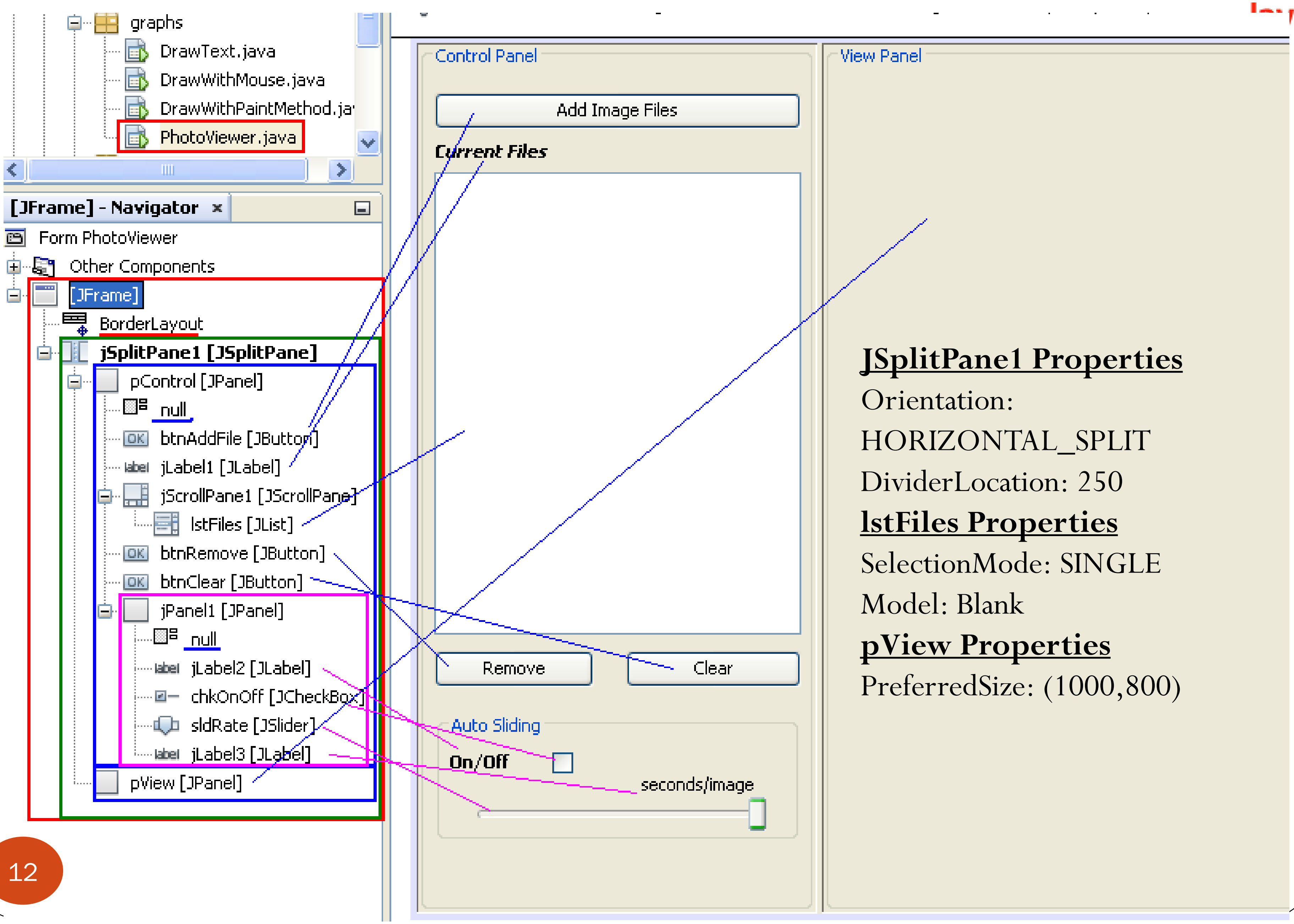
Click for viewing an image

Remove some images from the list

Click for auto sliding

Change slide rate

Design



graphs
DrawText.java
DrawWithMouse.java
DrawWithPaintMethod.ja
PhotoViewer.java

[JFrame] - Navigator
Form PhotoViewer
Other Components
[JFrame]
BorderLayout
jSplitPane1 [JSplitPane]
pControl [JPanel]
null
btnAddFile [JButton]
jLabel1 [JLabel]
jScrollPane1 [JScrollPane]
lstFiles [JList]
btnRemove [JButton]
btnClear [JButton]
jPanel1 [JPanel]
null
jLabel2 [JLabel]
chkOnOff [JCheckBox]
sldRate [JSlider]
jLabel3 [JLabel]
pView [JPanel]

JSplitPane1 Properties

Orientation: HORIZONTAL_SPLIT

DividerLocation: 250

lstFiles Properties

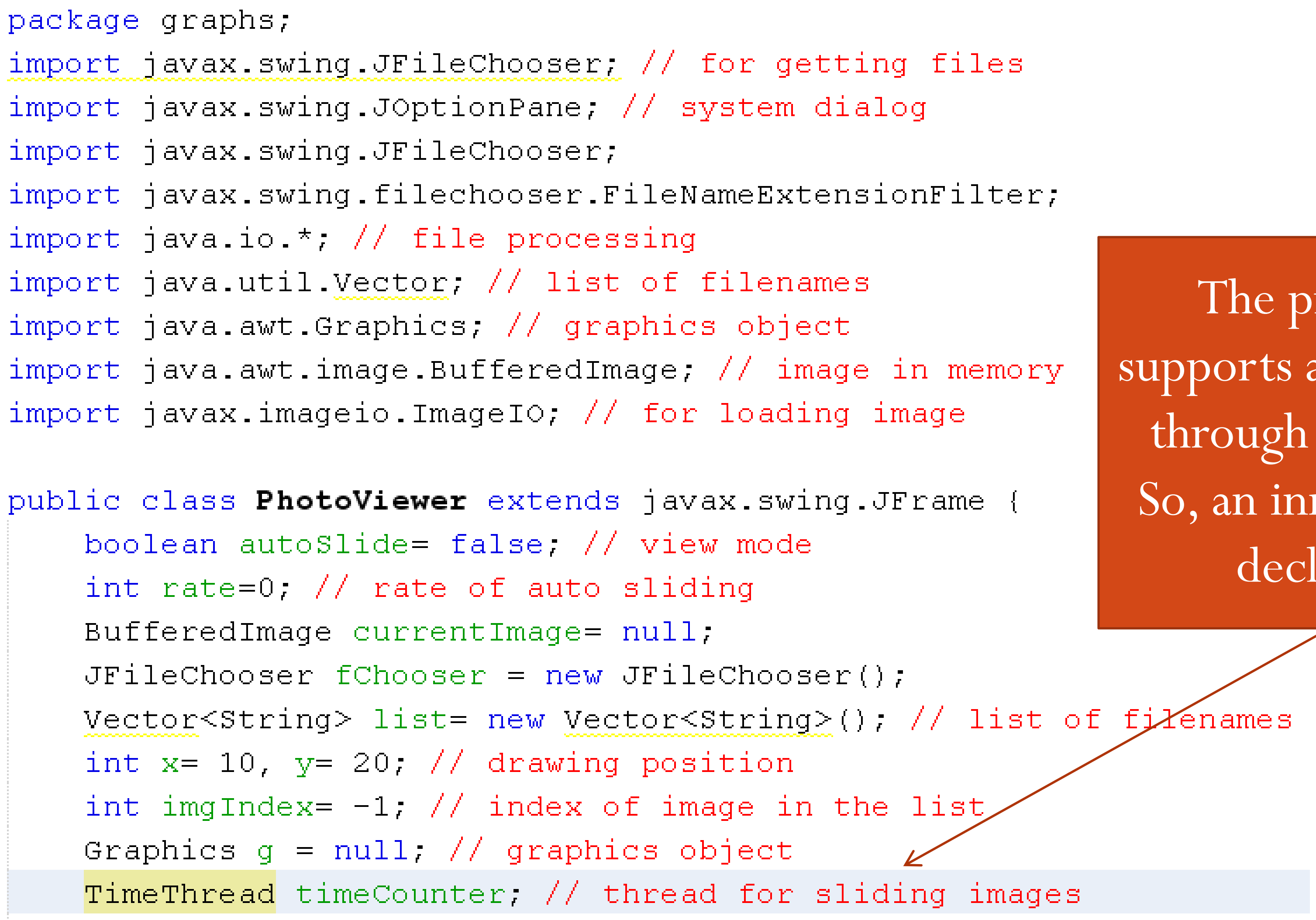
SelectionMode: SINGLE

Model: Blank

pView Properties

PreferredSize: (1000,800)

Code

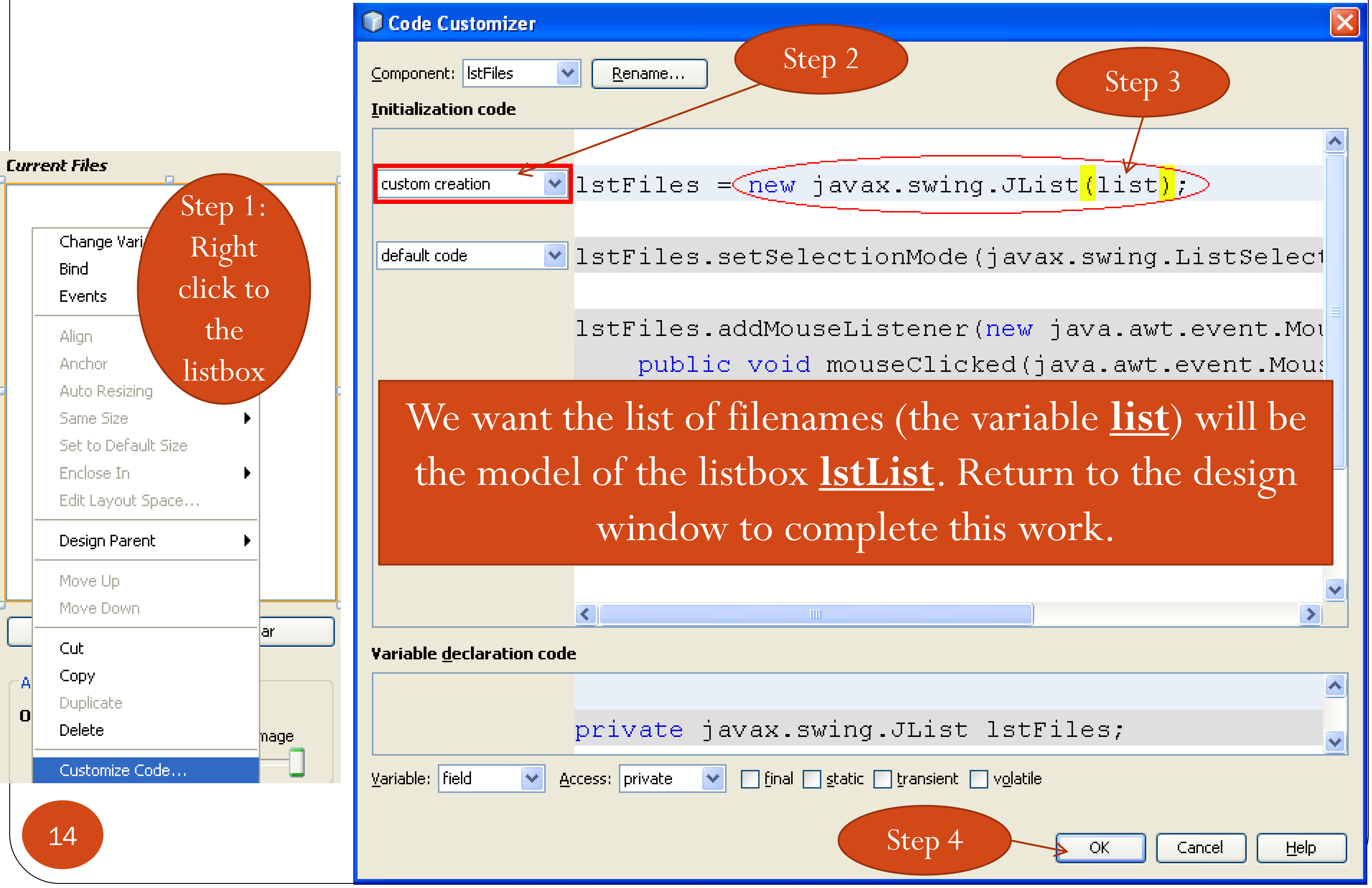


```
package graphs;
import javax.swing.JFileChooser; // for getting files
import javax.swing.JOptionPane; // system dialog
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileNameExtensionFilter;
import java.io.*; // file processing
import java.util.Vector; // list of filenames
import java.awt.Graphics; // graphics object
import java.awt.image.BufferedImage; // image in memory
import javax.imageio.ImageIO; // for loading image

public class PhotoViewer extends javax.swing.JFrame {
    boolean autoSlide= false; // view mode
    int rate=0; // rate of auto sliding
    BufferedImage currentImage= null;
    JFileChooser fChooser = new JFileChooser();
    Vector<String> list= new Vector<String>(); // list of filenames
    int x= 10, y= 20; // drawing position
    int imgIndex= -1; // index of image in the list
    Graphics g = null; // graphics object
    TimeThread timeCounter; // thread for sliding images
```

The program supports auto-sliding through a thread. So, an inner class is declared.

Code



```
lstFiles = new javax.swing.JList(list);
lstFiles.setSelectionMode(javax.swing.ListSelect
lstFiles.addMouseListener(new java.awt.event.Mou
    public void mouseClicked(java.awt.event.Mous
```

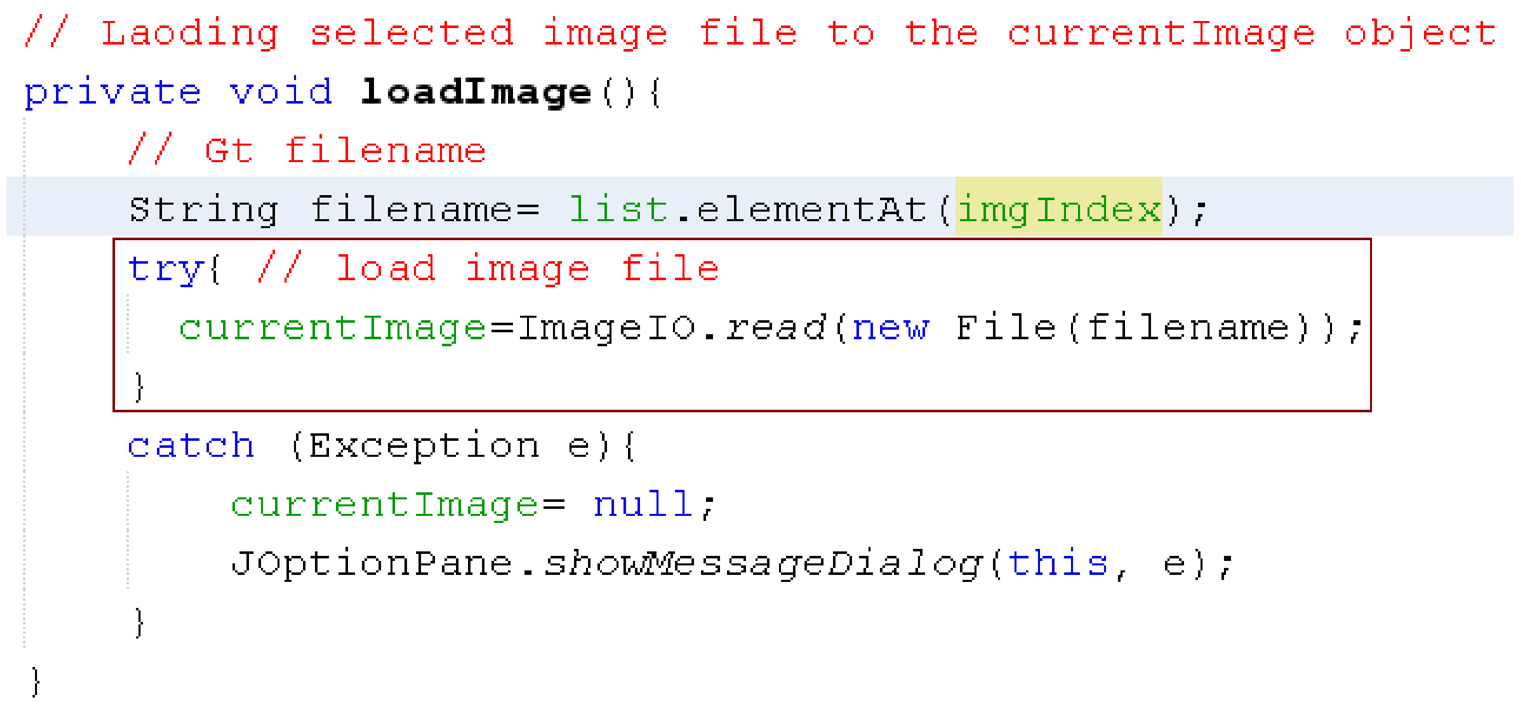
```
private javax.swing.JList lstFiles;
```

We want the list of filenames (the variable **list**) will be the model of the listbox **lstList**. Return to the design window to complete this work.

Code

```
// Constructor
public PhotoViewer() {
     initComponents();
     this.setSize(1000,600);
     // setup filter for mages files to the file chooser
     FileNameExtensionFilter filter =
     new FileNameExtensionFilter("Image files","GIF","JPG","JPEG","PNG");
     fChooser.setFileFilter(filter);
     // user can selects some files
     fChooser.setMultiSelectionEnabled(true);
     // drawing on PView, get it's graphic object
     g = this.pView.getGraphics();
     // Initially, user can not select auto-sliding mode
     this.sldRate.setEnabled(false);
     // Default rate of sliding = 1 images/sec
     this.sldRate.setValue(1);
}
```

Code



```
// Laoding selected image file to the currentImage object
private void loadImage(){
    // Gt filename
    String filename= list.elementAt(imgIndex);
    try{ // load image file
      currentImage=ImageIO.read(new File(filename));
    }
    catch (Exception e){
        currentImage= null;
        JOptionPane.showMessageDialog(this, e);
    }
}
```

Code

```
// Show currentImage to the pnel pView
private void showImage(){
    if (currentImage != null) {// Calculate the drawing position
       g.clearRect(x, y, pView.getWidth()-x, pView.getHeight()-y);// clear old image
       int imgWidth = currentImage.getWidth(); // get real image size
       int imgHeight= currentImage.getHeight();
       double ratio = 1.0*imgWidth/imgHeight;
       int areaWidth= this.pView.getWidth()-2*x; // Area for showing
       int areaHeight= this.pView.getHeight()-2*y;
       // the image is narrower than drawing area
       if( imgWidth<=areaWidth && imgHeight<=areaHeight){
           areaWidth= imgWidth;
           areaHeight= imgHeight;
       }
       else if (imgWidth>imgHeight){ //  horizontal image
          if (imgWidth<areaWidth) areaWidth= imgWidth;
              areaHeight = (int)(areaWidth/ratio);
          }
       else { // vertical image
           if (imgHeight<areaHeight) areaHeight= imgHeight;
               areaWidth = (int)(areaHeight*ratio);
       }
       g.drawImage(currentImage, x,y,areaWidth,areaHeight,this.pView);
    }
}
```

Code

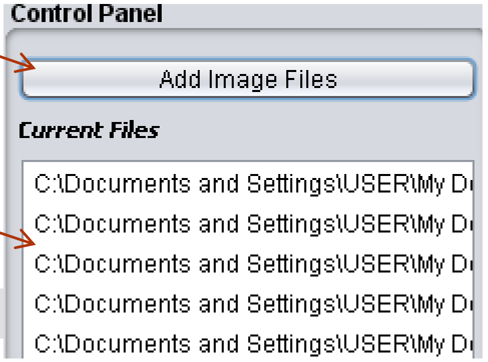
```
// Inner class- Thread for auto-sliding
class TimeThread extends Thread {
    @Override
    public void run(){
        imgIndex= lstFiles.getSelectedIndex();
        int n= lstFiles.getModel().getSize(); // number of files
        if (n>0 && autoSlide){ // auto-slide condition
            while (imgIndex<n){ // sliding to the end of the list
                try{
                    loadImage();
                    showImage();
                    imgIndex++;
                    sleep(1000*rate); // 1000 = 1 sec
                }
                catch (Exception e){
                    JOptionPane.showMessageDialog(null, e);
                }
            }
        }
    }
}
```

```
// Sliding images
private void slidingImage(){
    timeCounter = new TimeThread() ; // creat a thread
    timeCounter.start(); // the thread runs
}
```

TRƯỜNG ĐẠI HỌC FPT

Code

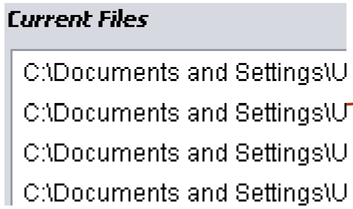
```
private void btnAddFileActionPerformed(java.awt.event.ActionEvent evt) {
    // Getting image filenames
    int returnVal= fChooser.showOpenDialog(this); // open file dialog
    if (returnVal==JFileChooser.APPROVE_OPTION){
       File[] files= fChooser.getSelectedFiles();
       for (File f: files) list.add(f.getAbsolutePath());
       lstFiles.setSelectedIndex(0);// the first filename
       this.lstFiles.updateUI();
    }
}
```


Control Panel
Add Image Files
Current Files
C:\Documents and Settings\USER\My D
C:\Documents and Settings\USER\My D
C:\Documents and Settings\USER\My D
C:\Documents and Settings\USER\My D
C:\Documents and Settings\USER\My D

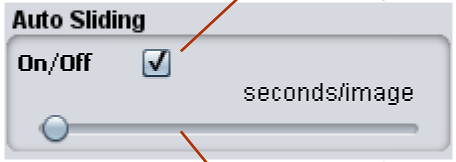
```
private void btnRemoveActionPerformed(java.awt.event.ActionEvent evt) {
    // Remove an image from the list
    int[] indices = lstFiles.getSelectedIndices();
    for (int i= indices.length-1; i>=0; i--)
        list.removeElementAt(i);
    lstFiles.updateUI();
}
```

```
private void btnClearActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    list.removeAllElements();
    lstFiles.updateUI();
}
```

Code


Current Files
C:\Documents and Settings\U
C:\Documents and Settings\U
C:\Documents and Settings\U
C:\Documents and Settings\U

```java
private void lstFilesMouseClicked(java.awt.event.MouseEvent evt) {
    // Show image
    imgIndex = lstFiles.getSelectedIndex();
    this.loadImage();
    this.showImage();
}
```



```java
private void chkOnOffMouseClicked(java.awt.event.MouseEvent evt) {
    // Change view mode
    if (chkOnOff.isSelected() && lstFiles.getSelectedIndex()>=0){
        this.autoSlide=true;
        this.sldRate.setEnabled(true);
        this.rate = this.sldRate.getValue();
        slidingImage();
    }
    else {
        if(timeCounter!=null) timeCounter=null;
        autoSlide= false;
        this.sldRate.setEnabled(false);
    }
}
```

```java
private void sldRateMouseReleased(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    this.rate = sldRate.getValue();
}
```

Summary

- Introduction
- The java.awt.**Image** Class
- The java.awt.image.**BufferedImage** Class
- The java.awt
- Reading/Loading an Image
- Drawing an Image
- Demonstrations

Summary

- Introduction
- Reading/Loading an Image
- Drawing an Image
- Creating and Drawing to an Image
- Demonstration

Thank You