

Overview

Building Graphical User Interfaces

- Constructing GUIs
- Interface components
- GUI layout
- Event handling

4.1

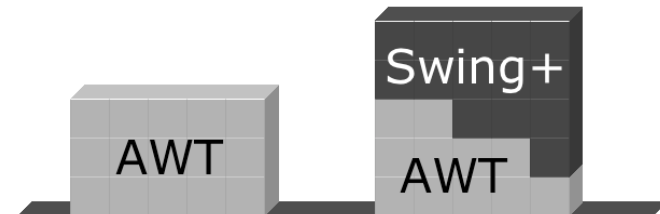
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

GUI Principles

- Components: GUI building blocks.
 - Buttons, menus, sliders, etc.
- Layout: arranging components to form a usable GUI.
 - Using layout *managers*.
- Events: reacting to user input.
 - Button presses, menu selections, etc.

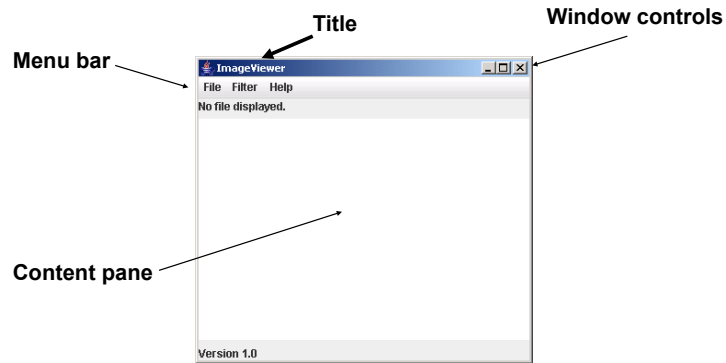
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

AWT and Swing



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Elements of a frame



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Creating a frame

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

public class ImageViewer
{
    private JFrame frame;

    /**
     * Create an ImageViewer show it on screen.
     */
    public ImageViewer()
    {
        makeFrame();
    }

    // rest of class omitted.
}
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

The content pane

```
/**
 * Create the Swing frame and its content.
 */
private void makeFrame()
{
    frame = new JFrame("ImageViewer");
    Container contentPane = frame.getContentPane();

    JLabel label = new JLabel("I am a label.");
    contentPane.add(label);

    frame.pack();
    frame.setVisible(true);
}
```



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Adding menus

- JMenuBar
 - Displayed below the title.
 - Contains the menus.
- JMenu
 - e.g. *File*. Contains the menu items.
- JMenuItem
 - e.g. *Open*. Individual items.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

```

private void makeMenuBar(JFrame frame)
{
    JMenuBar menubar = new JMenuBar();
    frame.setJMenuBar(menubar);

    // create the File menu
    JMenu fileMenu = new JMenu("File");
    menubar.add(fileMenu);

    JMenuItem openItem = new JMenuItem("Open");
    fileMenu.add(openItem);

    JMenuItem quitItem = new JMenuItem("Quit");
    fileMenu.add(quitItem);
}

```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Event handling

- Events correspond to user interactions with components.
- Components are associated with different event types.
 - Frames are associated with **WindowEvent**.
 - Menus are associated with **ActionEvent**.
- Objects can be notified when an event occurs.
 - Such objects are called *listeners*.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Centralised event receipt

- A single object handles all events.
 - Implements the **ActionListener** interface.
 - Defines an **actionPerformed** method.
- It registers as a listener with each component.
 - **item.addActionListener(this)**
- It has to work out which component has dispatched the event.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

ActionListener

```

public interface ActionListener
{
    public void actionPerformed(ActionEvent ev);
}

```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

```

public class ImageViewer implements ActionListener
{
    ...
    public void actionPerformed(ActionEvent e)
    {
        String command = e.getActionCommand();
        if (command.equals("Open"))
        {
            ...
        }
        else if (command.equals("Quit"))
        {
            ...
        }
        ...
    }
    ...
    private void makeMenuBar(JFrame frame)
    {
        ...
        openItem.addActionListener(this);
        ...
    }
}

```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Centralised event handling

- The approach works.
- It is used, so you should be aware of it.
- However ...
 - It does not scale well.
 - Identifying components by their text is fragile.
- An alternative approach is preferred.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Nested class syntax

- Class definitions may be nested.

```

- public class Enclosing
{
    ...
    private class Inner
    {
        ...
    }
}

```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Inner classes

- Instances of the inner class are localized within the enclosing class.
- Instances of the inner class have access to the private members of the enclosing class.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Anonymous inner classes

- Obey the rules of inner classes.
- Used to create one-off objects for which a class name is not required.
- Use a special syntax.
- The instance is always referenced via its supertype, as it has no subtype name.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Anonymous action listener

```
JMenuItem openItem = new JMenuItem("Open");  
  
openItem.addActionListener(new ActionListener()  
{  
    public void actionPerformed(ActionEvent e)  
    {  
        openFile();  
    }  
});
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Anonymous class elements

```
openItem.addActionListener(  
    new ActionListener()  
    {  
        public void actionPerformed(ActionEvent e)  
        {  
            openFile();  
        }  
    }  
);
```

Anonymous object creation

Class definition

Actual parameter

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

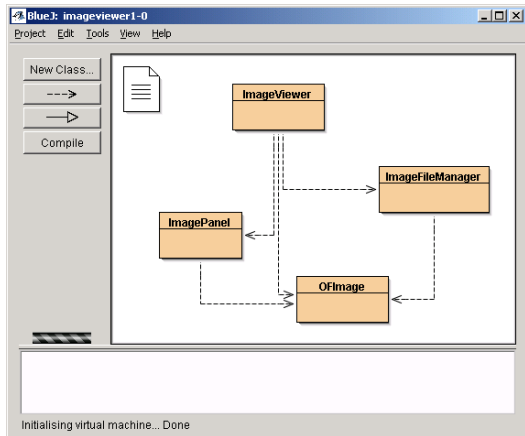
Exit on window close

```
frame.addWindowListener(new WindowAdapter()  
{  
    public void windowClosing(WindowEvent e)  
    {  
        System.exit(0);  
    }  
});
```

WindowAdapter provides a no-op implementation of the WindowListener interface.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

The imageviewer project



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Image processing



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Class responsibilities

- `ImageViewer`
 - Sets up the GUI structure.
- `ImageFileManager`
 - Static methods for image file loading and saving.
- `ImagePanel`
 - Displays the image within the GUI.
- `OFImage`
 - Models a 2D image.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

OFImage

- Our subclass of `BufferedImage`.
- Represents a 2D array of pixels.
- Important methods:
 - `getPixel`, `setPixel`
 - `getWidth`, `getHeight`
- Each pixel has a color.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Adding an ImagePanel

```
public class ImageViewer
{
    private JFrame frame;
    private ImagePanel imagePanel;

    ...

    private void makeFrame()
    {
        Container contentPane = frame.getContentPane();
        imagePanel = new ImagePanel();
        contentPane.add(imagePanel);
    }

    ...
}
```

Loading an image

```
public class ImageViewer
{
    private JFrame frame;
    private ImagePanel imagePanel;

    ...

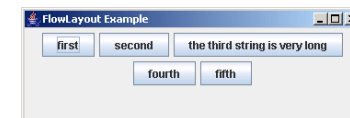
    private void openFile()
    {
        File selectedFile = ...;
        OFImage image =
            ImageFileManager.loadImage(selectedFile);
        imagePanel.setImage(image);
        frame.pack();
    }

    ...
}
```

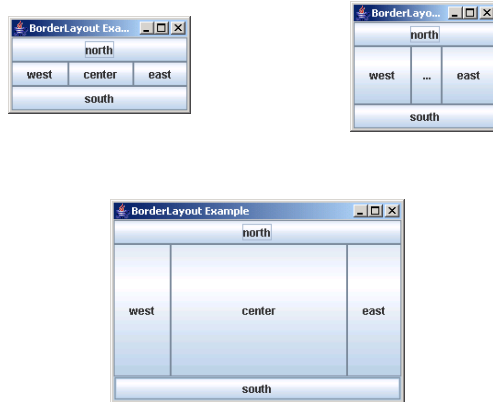
Layout managers

- Manage limited space for competing components.
 - BorderLayout, BorderLayout,
 - GridLayout, BorderLayout,
 - GridBagLayout.
- Manage Container objects, e.g. a content pane.
- Each imposes its own style.

FlowLayout



BorderLayout



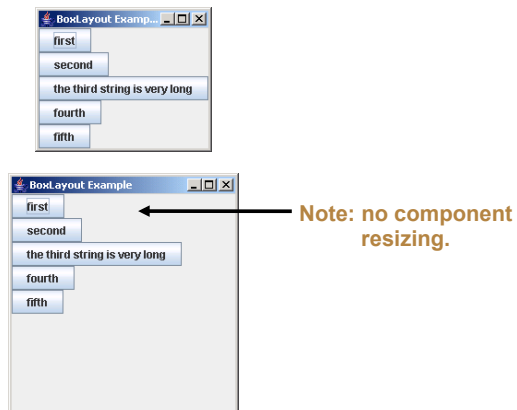
Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

GridLayout



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

BoxLayout



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Nested containers

- Sophisticated layouts can be obtained by nesting containers.
- Use `JPanel` as a basic container.
- Each container will have its own layout manager.
- Often preferable to using a `GridBagLayout`.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Struts and Glue

- Invisible components used as spacing.
- Available from the `Box` class.
- Strut: fixed size.
 - `Component createHorizontalStrut(int width)`
 - `Component createVerticalStrut(int height)`
- Glue: fills available space.
 - `Component createHorizontalGlue()`
 - `Component createVerticalGlue()`

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Dialogs

- Modal dialogs block all other interaction.
 - Forces a response from the user.
- Non-modal dialogs allow other interaction.
 - This is sometimes desirable.
 - May be difficult to avoid inconsistencies.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

JOptionPane standard dialogs

- Message dialog
 - Message text plus an OK button.
- Confirm dialog
 - Yes, No, Cancel options.
- Input dialog
 - Message text and an input field.
- Variations are possible.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

A message dialog

```
private void showAbout()  
{  
    JOptionPane.showMessageDialog(frame,  
        "ImageViewer\n" + VERSION,  
        "About ImageViewer",  
        JOptionPane.INFORMATION_MESSAGE);  
}
```



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Image filters

- Functions applied to the whole image.

```
int height = getHeight();
int width = getWidth();

for(int y = 0; y < height; y++)
{
    for(int x = 0; x < width; x++)
    {
        Color pixel = getPixel(x, y);
        alter the pixel's color value;
        setPixel(x, y, pixel);
    }
}
```

Adding further filters

```
private void makeLighter()
{
    if(currentImage != null)
    {
        currentImage.lighter();
        frame.repaint();
        showStatus("Applied: lighter");
    }
    else
    {
        showStatus("No image loaded.");
    }
}

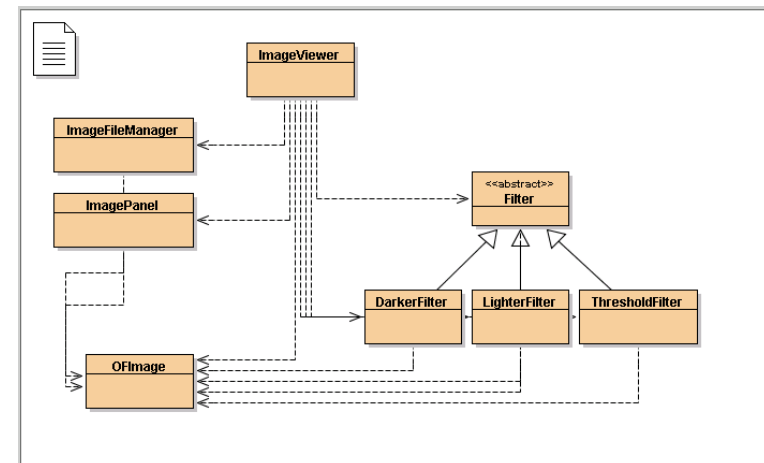
private void threshold()
{
    if(currentImage != null)
    {
        currentImage.threshold();
        frame.repaint();
        showStatus("Applied: threshold");
    }
    else
    {
        showStatus("No image loaded.");
    }
}
```



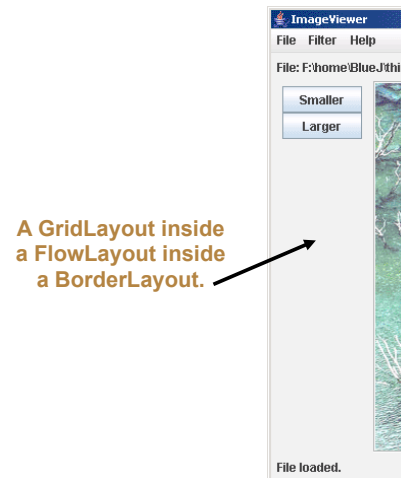
Adding further filters

- Define a `Filter` superclass (abstract).
- Create function-specific subclasses.
- Create a collection of subclass instances in `ImageViewer`.
- Define a generic `applyFilter` method.
- See *imageviewer2-0*.

imageviewer2-0



Buttons and nested layouts



Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Borders

- Used to add decoration around components.
- Defined in `javax.swing.border`
 - `BevelBorder`, `CompoundBorder`, `EmptyBorder`, `EtchedBorder`, `TitledBorder`.

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Adding spacing

```
JPanel contentPane = (JPanel) frame.getContentPane();
contentPane.setBorder(new EmptyBorder(6, 6, 6, 6));

// Specify the layout manager with nice spacing
contentPane.setLayout(new BorderLayout(6, 6));

imagePanel = new ImagePanel();
imagePanel.setBorder(new EtchedBorder());
contentPane.add(imagePanel, BorderLayout.CENTER);
```

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Other components

- Slider
- Spinner
- Tabbed pane
- Scroll pane

Objects First with Java - A Practical Introduction using BlueJ, © David J. Barnes, Michael Kölling

Review

- Aim for cohesive application structures.
 - Endeavour to keep GUI elements separate from application functionality.
- Pre-defined components simplify creation of sophisticated GUIs.
- Layout managers handle component juxtaposition.
 - Nest containers for further control.

Review

- Many components recognise user interactions with them.
- Reactive components deliver events to listeners.
- Anonymous inner classes are commonly used to implement listeners.