

11 Graphical User Interfaces

Overview

- Java API for GUIs
- Constructing GUIs
- Interface components
- Event handling
- *Example: image viewer*

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 2

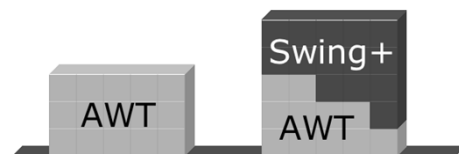
GUI Principles

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

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 3

AWT and Swing



Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 4

Creating a frame

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

public class ImageViewer
{
    private JFrame frame;

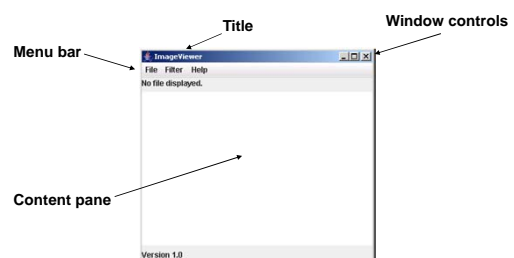
    /**
     * Create an ImageViewer show it on screen.
     */
    public ImageViewer()
    {
        makeFrame();
    }

    // rest of class omitted.
}
```

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 5

Elements of a frame



Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 6

The content pane

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

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

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



Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 7

Adding menus

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

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 8

```
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);
}
```

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 9

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*.

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 10

Centralized 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.

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 11

```
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);
        ...
    }
}
```

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 12

Centralized 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.

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 13

Nested class syntax

- Class definitions may be nested.

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

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 14

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.

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 15

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.

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 16


Anonymous action listener



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

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 17

Anonymous class elements

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

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 18

Exit on window close

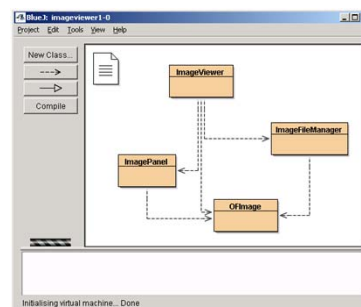
```
frame.addWindowListener(new WindowAdapter() {  
    public void windowClosing(WindowEvent e)  
    {  
        System.exit(0);  
    }  
});
```

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

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 19

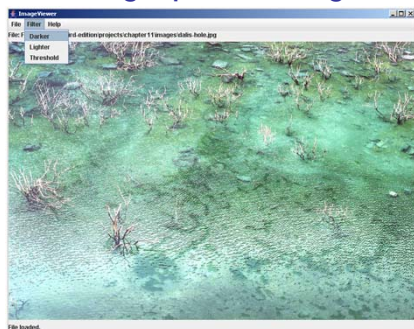
The imageviewer project



Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 20

Image processing



Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 21

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.

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 22

OFImage

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

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 23

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);  
    }  
  
    ...  
}
```

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 24

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();
    }

    ...
}
```

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 25

Review

- Aim for clean application structures.
 - *Keep GUI elements separate from application functionality!*
- Pre-defined components simplify creation of sophisticated GUIs.
- Many components recognize user interactions with them.
- Reactive components deliver events to listeners.

Object oriented programming, DAT042, D2, 12/13, lp 1

Lecture 11 26