Learning Goals for Today
Learning Goals

• Be able to name key User Interface “Anti-Patterns”

• Be able to identify key User Interface Principles

• Understand structure and behavior of basic Java Swing UI Elements

• Be able to read and understand basic Java Swing UI code
User Interface Antipatterns
1. Block user input even if it is unnecessary to do so
2. Block user input as often and as long as possible
3. Let the system freeze at irregular intervals without apparent reason
4. Make it illogical
5. Do not let users interrupt time-consuming and/or resource-hungry processes
6. Leave out functionality that would make the user’s life easier
7. Destroy the work context after each system reaction
8. Set bad default values

*adapted from: http://www.sapdesignguild.org/community/design/golden_rules.asp
UI ANTI-Patterns* (2/2)

9. Don’t listen to end-users

10. Mouse-only – forget the keyboard

11. Hide important information and functions from the user’s view

12. Use lots of jargon

13. Use lots of abbreviations – especially when there is enough space for the full spelling

14. Make it slow

15. Keep the user busy doing unnecessary work.

*adapted from: http://www.sapdesignguild.org/community/design/golden_rules.asp
User Interface Design Principles
Follow the Human* (1/2)

- Know your end-user
  - Skill levels
  - Workflows/Tasks
  - Domain-language/notations

- Human Behavior Patterns
  - Safe Exploration
  - Instant Gratification
  - Satisficing
  - Changes in Midstream
  - Deferred Choices

*adapted from: Jennifer Tidwell: Designing Interfaces, O’Reilly Media, Inc., 2005
Follow the Human* (1/2)

• Human Behavior Patterns (continued)
  – Incremental Construction
  – Habituation
  – Spatial Memory
  – Prospective Memory
  – Streamlined Repetition
  – Keyboard Only
  – Other People's Advice

• Create Mockups/Wireframes

*adapted from: Jennifer Tidwell: Designing Interfaces, O’Reilly Media, Inc., 2005
Organization Principles for Applications*

- Lists of objects
  - email messages in inbox
  - songs in playlist
- Lists of actions or tasks
  - Create, Read, Update, Delete (CRUD)
  - browse
  - select
- Lists of subject categories
  - “stores” metaphor
  - song categories: classic rock, hard rock, soft rock, ...
- Lists of tools
  - editor
  - calendar
  - batch processor...

*adapted from: Jennifer Tidwell: Designing Interfaces, O’Reilly Media, Inc., 2005
GUI Development Processes*

- Functionality requirements gathering
- User analysis:
  - What would the user want the system to do?
  - How would the system fit in with the user's normal workflow or daily activities?
  - How technically savvy is the user and what similar systems does the user already use?
  - What interface look & feel styles appeal to the user?
- Information architecture
- Prototyping
- Usability testing
- Graphic Interface design

Java Swing Basics
Java Swing

- MVC-based Java API for UI development
  - actually: M(VC)
  - customizable and extensible

- Platform-independent but supports native Look-and-Feel (LAF)
Swing MVC Structure

BaseSource

- attach(Observer)
- detach(Observer)
- update()

Model

- state
- getState()
- setState()
- service()

IObservable

- notify()

View/Controller

- notify()
- display()
- handleInput()

1 model

observers *

© Ingolf Krueger, 2010
MVC: Behavior

- Controller: `handleInput()`, `getState()`, `notify()`
- Model: `service()`, `update()`, `notify()`, `getState()`
- View: `display()`, `notify()`
Java Swing

- Rich set of GUI elements:
  - Containers
    - JFrame: basic windows w/title and border
    - JDialog: request/response dialogs
    - JPanel: container for other GUI elements
    - JScrollPane: wrap scroll bars around contained element
    - JFileChooser
    - JSplitPane: allow user to adjust the relative size of two containers
    - JTabbedPane: introduce tabbed interface
    - JToolBar: provide textual/graphical toolbar
  - Basic
    - JLabel: text/graphics labels
    - JButton: buttons
    - JTextArea: editable/non-editable for input/output
    - JList: selectable list of elements
    - JTable: tables
    - ...
Complex Swing UI (java.sun.com)
SwingChatClient

JFrame
public class SwingChatClient extends JFrame ... {
...  
    public SwingChatClient() {
        super("My Swing Chat Client");
        // this makes the window close, and System.exit() gets invoked
        // when the "window close" action is issued
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }  
...  
    public static void main(String[] args) {

        SwingChatClient client = new SwingChatClient();
        client.setLocation(100, 100);
        client.setSize(500, 500);
        client.setVisible(true);

    }
...  
}
JFrame

- Has multiple layers

- GUI Elements are added to the “Content Pane” layer.
  - Find it via getContentPane()

- Other layers useful for graphics and customization
Containers Have Layout Managers

- Flow layout
- Box layout
- Border layout
- Grid layout

adapted from java.sun.com
public class SwingChatClient extends JFrame implements ActionListener, DocumentListener {
    JPanel main_pane;
    List<JPanel> row_panes;

    ... public SwingChatClient() {
    ... 
        JPanel first_row = createRowPanel(contents);
        row_panes.add(first_row);

    ... 
        JPanel second_row = createRowPanel(contents);
        row_panes.add(second_row);

    ... 
        JPanel sixth_row = createRowPanel(contents);
        row_panes.add(sixth_row);
        main_pane = new JPanel();

    ...
// the main panel uses a vertical box layout
main_pane.setLayout(new BoxLayout(main_pane, BoxLayout.Y_AXIS));

// and we add all rows we have initialized above
for(JPanel pane : row_panes) {
    main_pane.add(pane);
}

// add the main panel to the window
getContentPane().add(main_pane);

// layout and render
pack();
SwingChatClient

![Swing Chat Client GUI]

- **Servers**: Server 2
- **Rooms**: Room 3
- **Users**: Ingolf, Max, Barry

Chat logs:
- Action performed: Server 2
- Action performed: Room 3
- You say: Hello
- You say: What's going on?

**JPanel**
SwingChatClient

![Swing Chat Client Diagram]

- JLabel

**Servers**: Server 2

**Rooms**: Room 3

**Chat Log**:
- action performed: Server 2
- action performed: Room 3
- you say: Hello
- you say: What's going on?
SwingChatClient

JComboBox
Box
(not editable)
SwingChatClient

![Swing Chat Client Interface]

**JComboBox Box (editable)**

Servers: Server 2

Rooms: Room 3

Users:
- Ingolf
- Max
- Barry

Chat History:
- action performed: Server 2
- action performed: Room 3
- you say: Hello
- you say: What's going on?
SwingChatClient

Servers: Server 2
Rooms: Room 3

JList

Ingolf
Max
Barry

action performed: Server 2
action performed: Room 3
you say: Hello
you say: What's going on?
SwingChatClient

My Swing Chat Client

Servers: Server 2
Rooms: Room 3

Ingolf
Max
Barry

action performed: Server 2
action performed: Room 3
you say: Hello
you say: What's going on?

JButton
Leave Room
SwingChatClient

My Swing Chat Client

Servers: Server 2
Rooms: Room 3

Ingolf
Max
Barry

action performed: Server 2
action performed: Room 3
you say: Hello
you say: What's going on?

JText Area (editable)
GUI Activities Create Events

• Follows Observer-Part of the MVC Pattern
• Have to implement the corresponding interfaces & register for Events you are interested in:
  – ActionListener
  – DocumentListener
  – ...
• Then process the events just like you do in the Chat Team Project:
  – find out what was the action/notification
  – identify the source
  – get any payload data (change in document text, for instance)
  – process data
Example: Listeners

public interface ActionListener extends EventListener {
    public void actionPerformed(ActionEvent e);
}

public interface DocumentListener extends EventListener {
    public void insertUpdate(DocumentEvent e);
    public void removeUpdate(DocumentEvent e);
    public void changedUpdate(DocumentEvent e);
}

public class SwingChatClient extends JFrame implements ActionListener {

        JButton leave_room;

        public SwingChatClient() {

            leave_room = new JButton("Leave Room");
            leave_room.addActionListener(this);

        }

}
What have you learned today?
Learning Goals

• Be able to name key User Interface “Anti-Patterns”

• Be able to identify key User Interface Principles

• Understand structure and behavior of basic Java Swing UI Elements

• Be able to read and understand basic Java Swing UI code