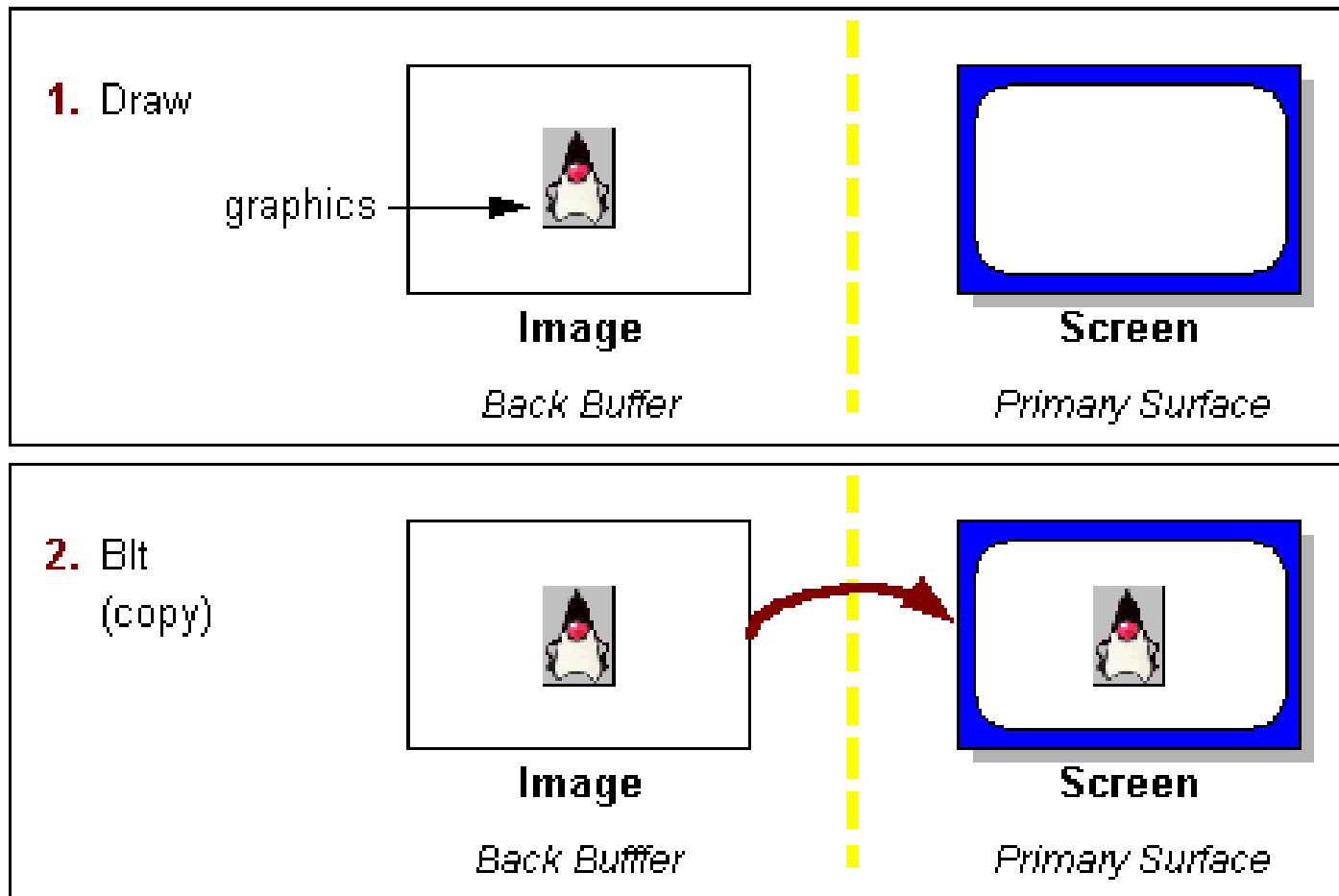


Graphical User Interface (GUI), Part 2

- Double Buffering
- Various components
 - Menu Bar, Menu, and Menu Items
 - Combo Box
 - Table
- Java Beans
 - For visual programming

Double Buffering

Double Buffering

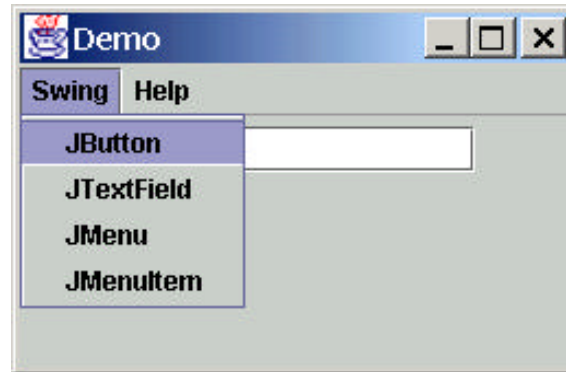


[Source: java.sun.com]

Double Buffering, cont.

- Double buffering is used to eliminate visual draws.
- Used extensively in Swing use the method **setDoubleBuffered** in **javax.swing.JComponent**.
- An alternative technique is called *page flipping*.
- Page flipping is used to avoid tearing, a splitting effect that occurs when drawing to the screen happens faster than the monitor's refresh rate.

Menu and Menu Items



- The class **JMenuBar**, **JMenu**, and **JMenuItem** are used for this purpose.

Menu and Menu Items, cont.

```
public class DemoApplet extends JApplet {
    JTextField t = new JTextField(15);
    Container cp;

    ActionListener al = new ActionListener() {
        public void actionPerformed(ActionEvent e){
            t.setText(((JMenuItem)e.getSource()).getText());
        }
    };

    JMenu[] menus = { new JMenu("Swing"),
                      new JMenu("Help") };

    JMenuItem[] swingItems = { new JMenuItem("JButton"),
                                new JMenuItem("JTextField"),
                                new JMenuItem("JMenu"),
                                new JMenuItem("JMenuItem") };

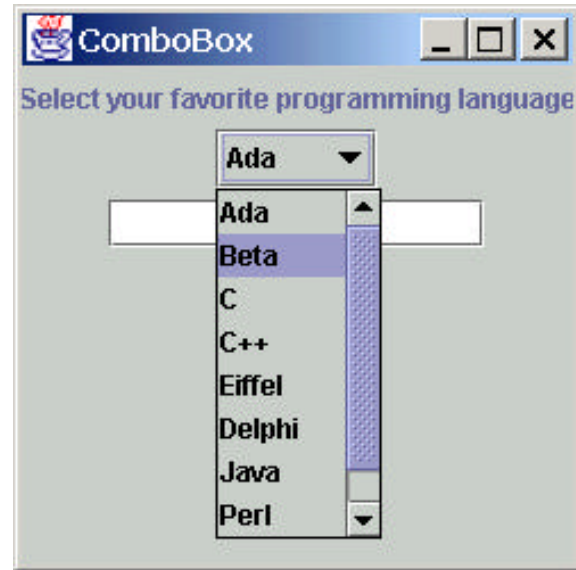
    JMenuItem[] helpItems = { new JMenuItem("Topics"),
                               new JMenuItem("About") };
```

Menu and Menu Items, cont.

```
public void init() {
    // the swing menu
    for(int i = 0; i < swingItems.length; i++) {
        swingItems[i].addActionListener(a1);
        menus[0].add(swingItems[i]);
    }
    // the help menu
    for(int i = 0; i < helpItems.length; i++) {
        helpItems[i].addActionListener(a2);
        menus[1].add(helpItems[i]);
    }

    // create the menu bar
    JMenuBar mb = new JMenuBar();
    for(int i = 0; i < menus.length; i++) {
        mb.add(menus[i]);
    }
    // set up the menu bar
    setJMenuBar(mb);
    cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(t);
}
```

Combo Box



- The class **JComboBox** is used for this purpose.
- One and only one element from the list can be selected.

Combo Box, cont.

```
public class ComboBox extends JApplet {
    JTextField t = new JTextField(15);
    JLabel      l =
        new JLabel ("Select your favorite programming language");
    Container cp;

    ActionListener al = new ActionListener() {
        public void actionPerformed(ActionEvent e){
            t.setText(
                (String)((JComboBox)e.getSource()).getSelectedItem());
        }
    };

    String[] languages = { "Ada", "Beta", "C", "C++",
                           "Eiffel", "Delphi", "Java",
                           "Perl", "Python"};

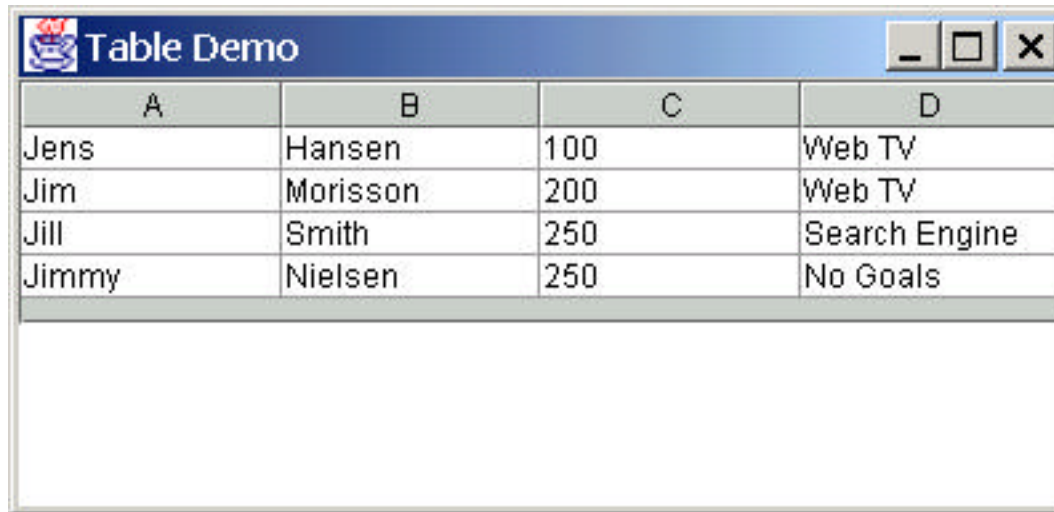
    JComboBox cb = new JComboBox();
}
```


Combo Box, cont.

```
public void init() {
    // populate the combo box
    for(int i = 0; i < languages.length; i++) {
        cb.addItem(languages[i]);
    }
    // connect the action listener
    cb.addActionListener (al);
    cp = getContentPane();
    cp.setLayout(new FlowLayout());
    cp.add(l);
    cp.add(cb);
    cp.add(t);
}

public static void main(String[] args) {
    ComboBox applet = new ComboBox();
    JFrame frame = new JFrame("ComboBox");
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().add(applet);
    frame.setSize(250,250);
    applet.init();
    applet.start();
    frame.setVisible(true);
}
```

Tables



A	B	C	D
Jens	Hansen	100	Web TV
Jim	Morisson	200	Web TV
Jill	Smith	250	Search Engine
Jimmy	Nielsen	250	No Goals

- The classes **JTable** and **AbstractTableModel** are used.
 - The latter controls the data

Tables, cont.

```
public class Table extends JApplet {
    JTextArea text = new JTextArea(4, 24);

    // AbstractTableModel controls all data
    class TModel extends AbstractTableModel {
        Object[][] table_data = {
            {"Jens", "Hansen", "100", "Web TV"},
            {"Jim", "Morisson", "200", "Web TV"},
            {"Jill", "Smith", "250", "Search Engine"},
            {"Jimmy", "Nielsen", "250", "No Goals"}};

        // reprint table data when changes
        class TMList implements TableModelListener {
            public void tableChanged(TableModelEvent e){
                text.setText(""); // clear screen
                for(int i = 0; i < table_data.length; i++) {
                    for(int j = 0; j < table_data[i].length; j++){
                        text.append(table_data[i][j] + " ");
                    }
                    text.append("\n");
                }
            }
        }
    }
}
```

OOP: Exception Handling

Tables, cont.

```
public TModel() {
    addTableModelListener(new TMList());
}
public int getColumnCount() {
    return table_data[0].length;
}
public int getRowCount() {
    return table_data.length;
}

public Object getValueAt(int row, int col) {
    return table_data[row][col];
}
}
public void init() {
    Container cp = getContentPane();
    JTable the_table = new JTable(new TModel());
    cp.add(the_table);
    cp.add(BorderLayout.CENTER, text);
}
```

Java Beans

- Component programming model
- Core JDK1.1 capability
- Must be able to instantiate, query and configure objects at design time
- Java *reflection* provides method and field information on a "live" object.
 - Methods, arguments, return values
- Beans specifies a naming convention.
 - Identifies design-time fields, event handlers
- For information see <http://java.sun.com/products/javabeans/>

Java Beans, cont.

- Simply a Java class (or classes)
- Supports three concepts
 - Properties
 - Events
 - Methods
- Follows naming convention to identify the concepts.

Java Beans Properties

- For a property named weight create two methods
 - `getWeight()` and
 - `setWeight()`. (First letter automatically to lowercase).
- For boolean property possible to use “is” instead of “get.”
- "Ordinary" methods are public.
- Events use the same “Listeners” with add- and remove-methods.
 - You can create your own events.

A Simple Java Bean

```
import java.awt.*; // [Source: java.sun.com]
import java.io.Serializable;
public class SimpleBean extends Canvas
                                implements Serializable{
    private Color color = Color.green;

    //property getter method public
    Color getColor(){ return color; }

    //property setter method. Sets color and repaints.
    public void setColor(Color newColor){
        color = newColor; repaint();
    }
    public void paint(Graphics g){
        g.setColor(color); g.fillRect(20, 5, 20, 30);
    }
    //Constructor sets inherited properties
    public SimpleBean(){
        setSize(60,40);
        setBackground(Color.red);
    }
}
```


Summary

- This should get you started programming GUIs
- Listener event model and Beans are huge steps forward.
- Swing is a good UI library.
- All Swing components are Java Beans.
- Numerous application builders use Java Beans.
- Java Beans enable RAD environments.
- Java UI library has gone through a lot of design changes.
- Use a GUI builder for your project.