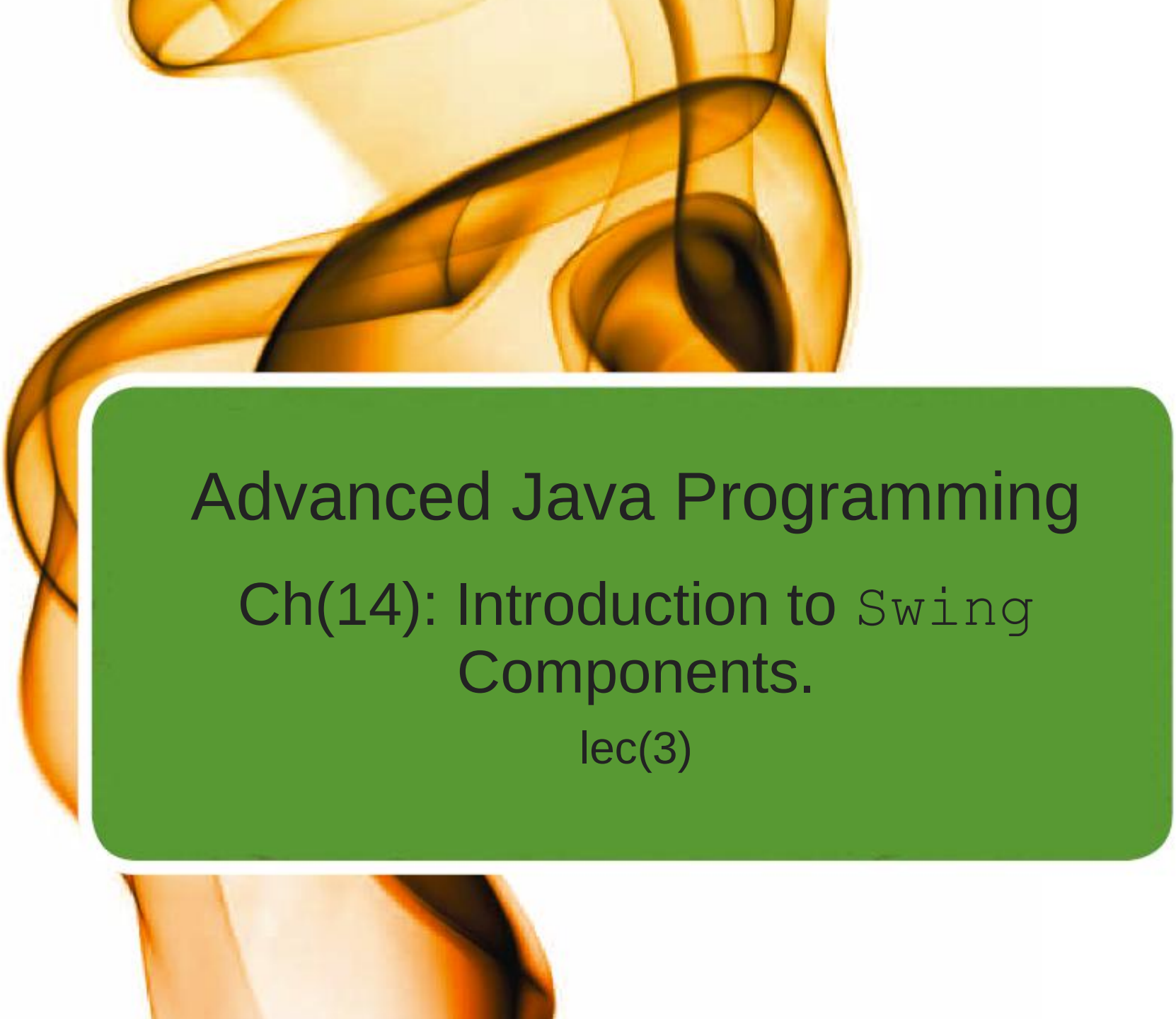

Advanced Java Programming

Ch(14): Introduction to `Swing` Components.

lec(3)

Extending the `JFrame` Class

- Create class that descends from `JFrame` class
- Advantage
  - Set `JFrame`'s properties within object's constructor
  - When `JFrame` child object created
    - Automatically endowed with specified features
- How?
  - Create child class using keyword `extends`
  - Call parent class's constructor method Using keyword `super`

Extending the `JFrame` Class (cont'd.)

```
import javax.swing.*;
public class JMyFrame extends JFrame
{
   final int WIDTH = 200;
   final int HEIGHT = 120;
   public JMyFrame()
   {
      super("My frame");
      setSize(WIDTH, HEIGHT);
      setVisible(true);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

Figure 14-15 The JMyFrame class

```
public class CreateTwoJMyFrameObjects
{
    public static void main(String[] args)
    {
        JMyFrame myFrame = new JMyFrame();
        JMyFrame mySecondFrame = new JMyFrame();
    }
}
```

Figure 14-16 The CreateTwoJMyFram

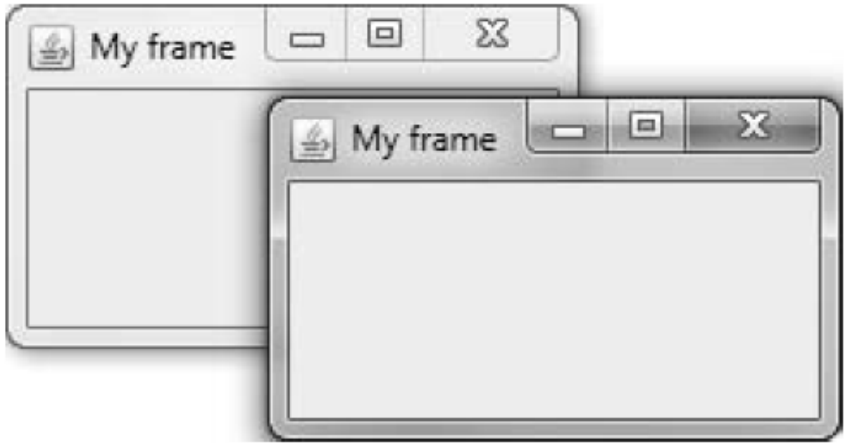

Figure 14-17 Output of the CreateTwoJMyFrameObjects application after dragging the top frame

Using the JCheckBox, and JComboBox Classes

- Besides JButtons and JTextFields
  - Several other Java components allow a user to make selections in a UI environment

The `JCheckBox` Class

- `JCheckBox`
  - Consists of a label positioned beside a square
  - Click square to display or remove check mark
  - Use to allow user to turn option on or off
- Constructors

`JCheckBox()` // No label, unselected

`JCheckBox("Check here")` // Label, unselected

`JCheckBox("Check here",false)`//Label,unselected

`JCheckBox("Check here",true)`// Label, selected

```
import java.awt.*;
import javax.swing.*;
public class CheckBoxDemonstration extends JFrame
{
FlowLayout flow = new FlowLayout();
JLabel label = new JLabel("What would you like to drink?");
JCheckBox coffee = new JCheckBox("Coffee", false);
JCheckBox cola = new JCheckBox("Cola", false);
JCheckBox milk = new JCheckBox("Milk", false);
JCheckBox water = new JCheckBox("Water", false);
public CheckBoxDemonstration()
{
super("CheckBox Demonstration");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(new FlowLayout());
label.setFont(new Font("Arial", Font.ITALIC, 22));
add(label);
add(coffee);
add(cola);
add(milk);
add(water);
}
public static void main(String[] arguments)
{
final int FRAME_WIDTH = 350;
final int FRAME_HEIGHT = 120;
CheckBoxDemonstration frame = new CheckBoxDemonstration();
frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
frame.setVisible(true);
}
}
```

The `JCheckBox` Class (cont'd.)

Figure 14-31 Output of the `CheckBoxDemonstration` class

The JCheckBox Class (cont'd.)

Method	Purpose
`void setText(String)`	Sets the text for the `JCheckBox`
`String getText()`	Returns the `JCheckBox` text
`void setSelected(boolean)`	Sets the state of the `JCheckBox` to `true` for selected or `false` for unselected
`boolean isSelected()`	Gets the current state (checked or unchecked) of the `JCheckBox`

Table 14-5 Frequently used `JCheckBox` methods

Using the `JComboBox` Class

- `JComboBox`
  - Component that combines two features
    - Display area showing one option
    - List box containing additional options
  - When user clicks `JComboBox`, list of alternative items drops down
    - User selects one to replace box's displayed item

Using the `JComboBox` Class (cont'd.)

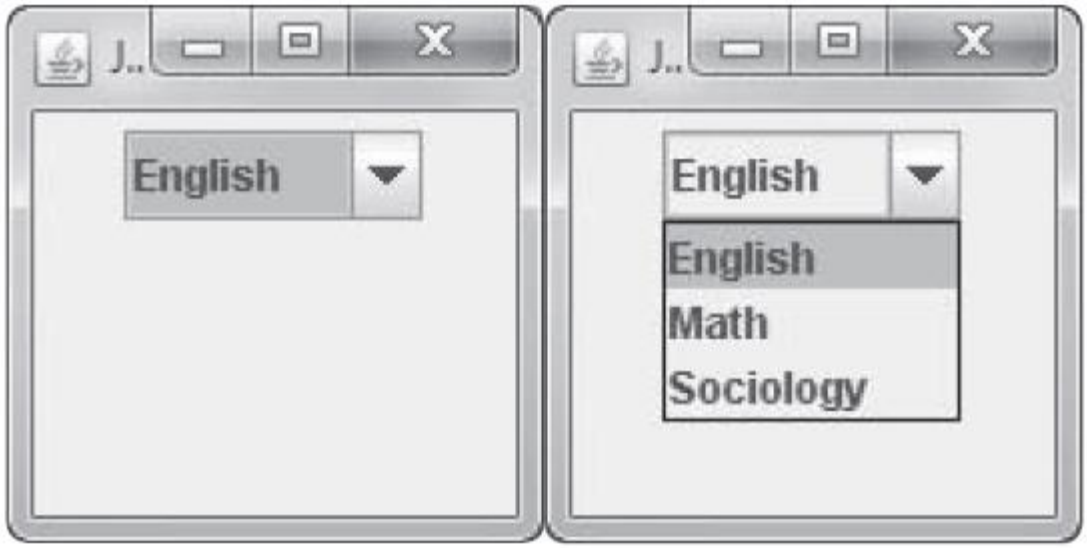

Figure 14-35 A JComboBox before and after the user clicks it

Using the `JComboBox` Class (cont'd.)

- Build `JComboBox`
  - Use constructor with no arguments
  - Add items with `addItem()` method
  - Alternatively, construct by using array of `Objects` as constructor argument
  - Example :

```
String[] majorArray ={"English","Math","Sociology"};
JComboBox majorChoice = new JComboBox(majorArray);
```

Method	Purpose
`void addItem(Object)`	Adds an item to the list
`void removeItem(Object)`	Removes an item from the list
`void removeAllItems()`	Removes all items from the list
`Object getItemAt(int)`	Returns the list item at the index position specified by the integer argument
`int getItemCount()`	Returns the number of items in the list
`int getMaximumRowCount()`	Returns the maximum number of items the combo box can display without a scroll bar
`int getSelectedIndex()`	Returns the position of the currently selected item
`Object getSelectedItem()`	Returns the currently selected item
`Object[] getSelectedObjects()`	Returns an array containing selected `Objects`
`void setEditable(boolean)`	Sets the field to be editable or not editable
`void setMaximumRowCount(int)`	Sets the number of rows in the combo box that can be displayed at one time
`void setSelectedIndex(int)`	Sets the index at the position indicated by the argument
`void setSelectedItem(Object)`	Sets the selected item in the combo box display area to be the `Object` argument

Table 14-6 Some `JComboBox` class methods

```
import java.awt.*;
import javax.swing.*;
public class CoboBoxDemonstration extends JFrame
{
FlowLayout flow = new FlowLayout();
String[] majorArray = {"English", "Math", "Sociology"};
JComboBox majorChoice = new JComboBox(majorArray);
public CoboBoxDemonstration()
{
super("CheckBox Demonstration");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(new FlowLayout());
add(majorChoice);
}
public static void main(String[] arguments)
{
final int FRAME_WIDTH = 350;
final int FRAME_HEIGHT = 120;
CoboBoxDemonstration frame = new CoboBoxDemonstration();
frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
frame.setVisible(true);
}
}
```