

Lecture 02

Creating Graphical User Interface

Part 1

Java GUI and the swing package

References:

- (1) Textbook, chapter 10, 11
- (2) Java-Tutorials/tutorial-2015/uiswing/index.html

Why should we study this lecture?



Objectives

- Do you want to create GUI applications (graphic user interface)?
- How do GUI applications execute?
- What do Java APIs support for developing GUI App?

Knowledge used in this lecture:

- **Event**: Some thing happens.
- **Event in applications**: In almost cases, when IO devices complete a task, an event is thrown → When user presses a key or clicks mouse, events are thrown.
- **Event handler**: Code executes when the appropriate event is thrown.
- **How does application know an event?**
When an IO device completes a task, a signal (called as interrupt) from the device will be sent to CPU and operating system (OS) knows it. OS collects states of keyboard, mouse position, mouse keys,..., called as event object then OS passes the event object to the application.

- Two Kinds of Applications
- Java Model for Event Management
- About Java Foundation Classes (JFC)
- A Strategy for Designing the GUI
- Using Ordinary Swing Components
- A Demonstration

1- Two Kinds of Apps



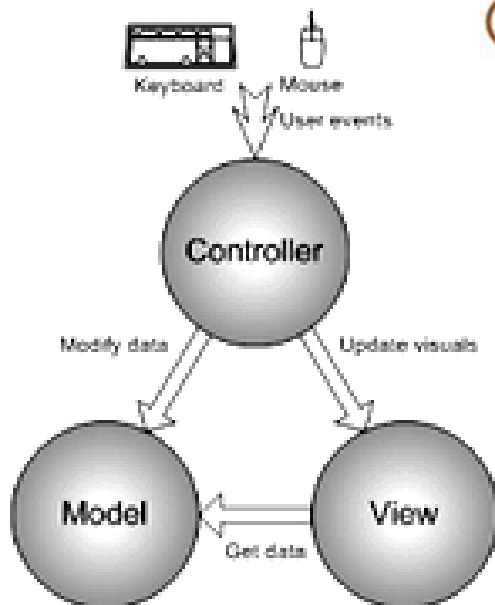
Console App.
Compute-centric Apps

```
C:\WINDOWS\system32\cmd.exe

G:\GiangDay\FU\CoreJava\Chapter01\build\classes>java myPackage.Ex01.8.3

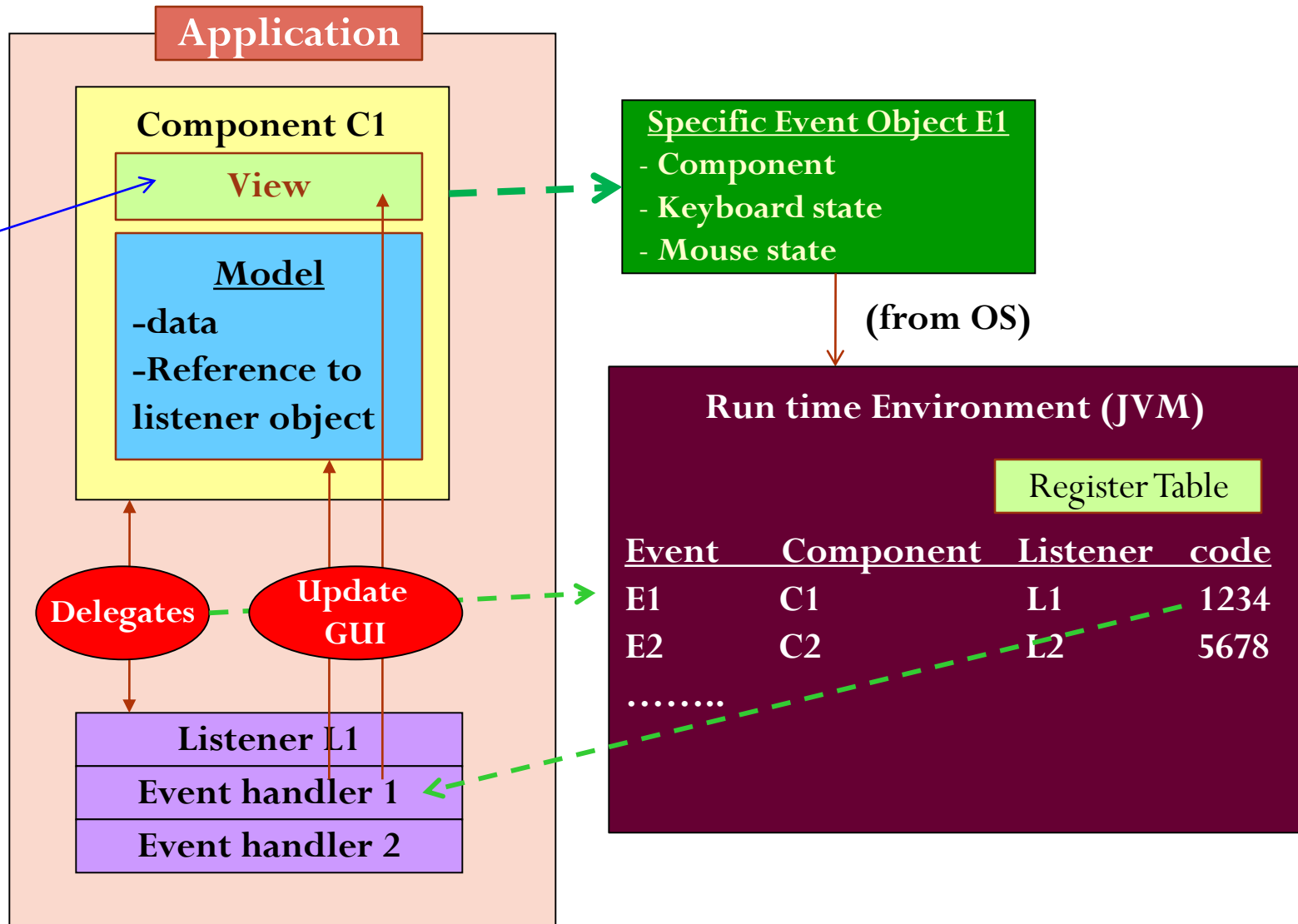
G:\GiangDay\FU\CoreJava\Chapter01\build\classes>pause
Press any key to continue . . .
```

Event-based App.
User-centric Apps(GUI)



Model-View Controller Architecture for GUI component.
Model: Object contains data.
View: Object users can see it on the screen
Controller: Object manages events

2-Java model for event management



Some Common Events



<i>Object</i>	<i>Event</i>	<i>Interface</i>	<i>Method</i>
JButton	ActionEvent	ActionListener	actionPerformed()
JCheckBox	ActionEvent ItemEvent	ActionListener ItemListener	actionPerformed() itemStateChanged()
JRadioButton	ActionEvent ItemEvent	ActionListener ItemListener	actionPerformed() itemStateChanged()
JTextField	ActionEvent	ActionListener	actionPerformed()
JTextArea	FocusEvent	FocusListener	focusGained(), focusLost()
JPasswordField	ActionEvent	ActionListener	actionPerformed()

Mouse ▶
 MouseMotion ▶
 MouseWheel ▶

mouseClicked
 mouseEntered
 mouseExited
 mousePressed
 mouseReleased

MouseMotion ▶
 MouseWheel ▶
 MouseWheel ▶

mouseDragged
 mouseMoved
 mouseWheelMoved

More Details about events will be introduced in the next session.

Các phương thức của interface `MouseListener`:

- **`public void mousePressed(MouseEvent event)`**: được gọi khi một nút chuột được nhấn và con trỏ chuột ở trên component.
- **`public void mouseClicked(MouseEvent event)`**: được gọi khi một nút chuột được nhấn và nhả trên component mà không di chuyển chuột.
- **`public void mouseReleased(MouseEvent event)`**: được gọi khi một nút chuột nhả ra khi kéo rê.
- **`public void mouseEntered(MouseEvent event)`**: được gọi khi con trỏ chuột vào trong đường biên của một component.
- **`public void mouseExited(MouseEvent event)`**: được gọi khi con trỏ chuột ra khỏi đường biên của một component.

3- About Java Foundation Classes

- JFC encompass a group of features for building graphical user interfaces (GUIs) and adding rich graphics functionality and interactivity to Java applications. It is defined as containing the features shown in the table below:

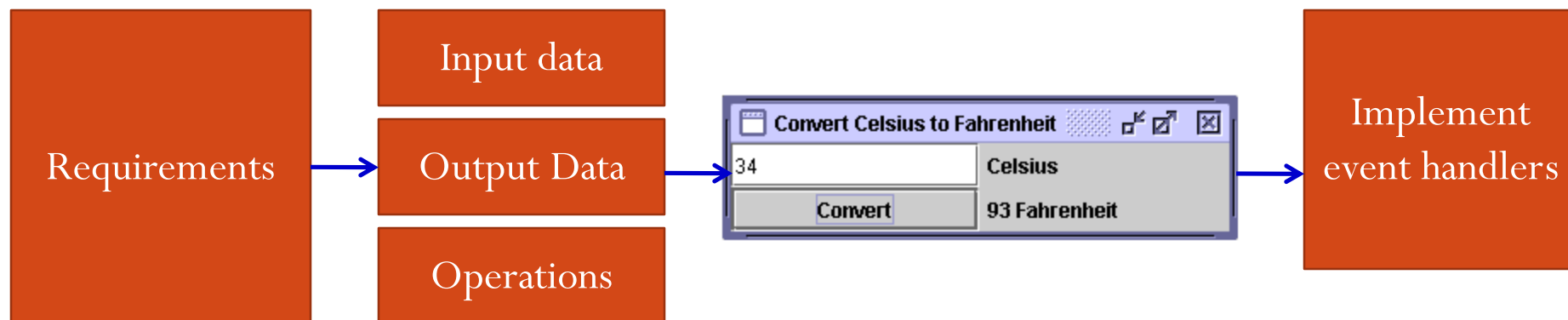
Feature	Description
Swing GUI Components	Includes everything from buttons to split panes to tables. Many components are capable of sorting, printing, and drag and drop, to name a few of the supported features.
Pluggable Look-and-Feel Support	Allowing a choice of look and feel. For example, the same program can use either the Java or the Windows look and feel. Additionally, the Java platform supports the GTK+ look and feel, which makes hundreds of existing look and feels available to Swing programs. Many more look-and-feel packages are available from various sources.
Accessibility API	Enables assistive technologies, such as screen readers and Braille displays, to get information from the user interface.
Java 2D API	Enables developers to easily incorporate high-quality 2D graphics, text, and images in applications and applets. Java 2D includes extensive APIs for generating and sending high-quality output to printing devices.
Internationalization	Allows developers to build applications that can interact with users worldwide in their own languages and cultural conventions. With the input method framework developers can build applications that accept text in languages that use thousands of different characters, such as Japanese, Chinese, or Korean.

- **Which Swing Packages Should I Use?**

The Swing API is powerful, flexible — and immense. The Swing API has 18 public packages:

<code>javax.accessibility</code>	<code>javax.swing.plaf</code>	<code>javax.swing.text</code>
<code>javax.swing</code>	<code>javax.swing.plaf.basic</code>	<code>javax.swing.text.html</code>
<code>javax.swing.border</code>	<code>javax.swing.plaf.metal</code>	<code>javax.swing.text.html.parser</code>
<code>javax.swing.colorchooser</code>	<code>javax.swing.plaf.multi</code>	<code>javax.swing.text.rtf</code>
<code>javax.swing.event</code>	<code>javax.swing.plaf.synth</code>	<code>javax.swing.tree</code>
<code>javax.swing.filechooser</code>	<code>javax.swing.table</code>	<code>javax.swing.undo</code>

4- A Strategy for Designing the GUI



- Identify needed components.
- Isolate regions and behaviors.
- Sketch (phác hoạ) the GUI.
- Choose Layout managers.

5- Ordinary Swing Component

Chapter11 - NetBeans IDE 6.7.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config> Search (Ctrl+I)

Projects

- Chapter11
 - Source Packages
 - <default package>
 - Demo_1.java
 - Test Packages
 - Libraries
 - Test Libraries

Component Properties:
Name, size, location,
background color,
foreground color, font,
enable, editable

Set/get

Palette

Swing Controls

- OK Button label Label
- ON Toggle Button ☐ Check Box
- Radio Button ☐ Button Group
- Combo Box List

txt1 [JTextField] - Properties

Properties Binding Events Code

Properties

background	[0,0,204]
columns	0
document	<default>
editable	<input checked="" type="checkbox"/>
font	The quick brown fox
foreground	[255,255,0]
horizontalAlignment	LEADING
text	Textfield

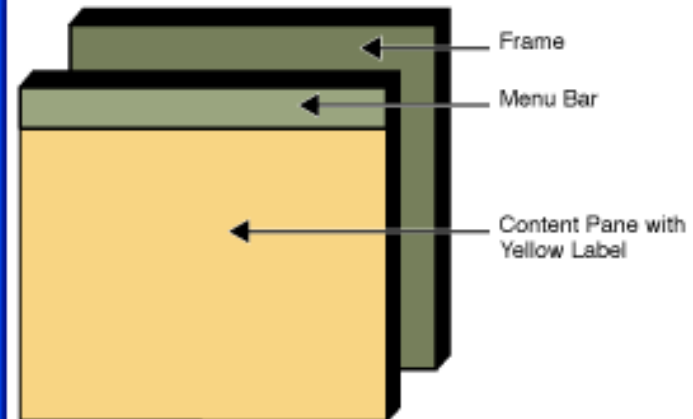
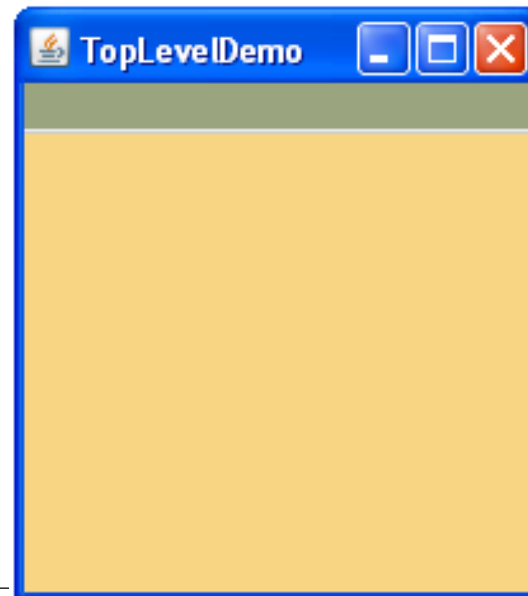
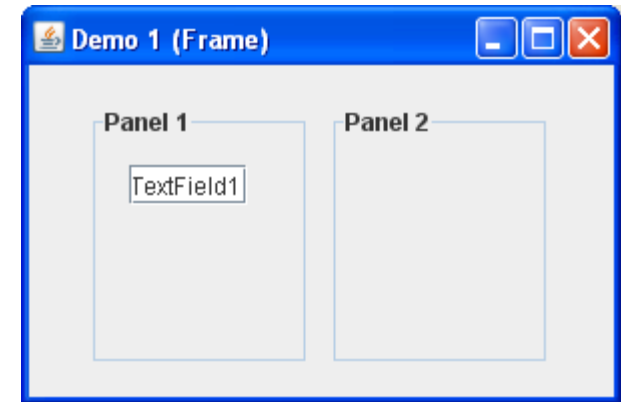
txt1 [JTextField]

- **javax.swing** package contains APIs for developing GUI Apps Refer to Java documentation for more details of this package.
- Almost component class names begin with 'J'
- **Three categories:**
 - **Container components**
 - **Ordinary components:**
 - **Menu components.**

Ordinary Swing Components...

Containers: JFrame, JPanel

- They contain other components.
- They use layout managers to determine the size and position of their child components.
- JFrame is a top-level container.
- JPanel is called general-purpose container.



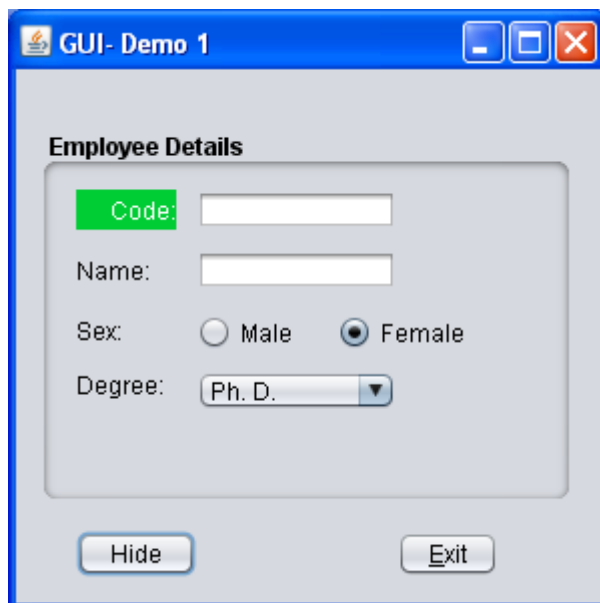
- JLabel
- JTextField
- JRadioButton
(ButtonGroup)
- JComboBox
- JTextArea
- JScrollBar
- JCheckBox
- JButton

The image shows a Java Swing window titled "Employee Details". It contains the following components:

- JLabel:** The title "Employee Details" and labels for "Code:", "Name:", "Sex:", "E-mail:", "Degree:", "Experiences:", and "Hobbies:".
- JTextField:** Input fields for "Code:", "Name:", and "E-mail:".
- JRadioButton (ButtonGroup):** Radio buttons for "Male" (selected) and "Female" under the "Sex:" label.
- JComboBox:** A dropdown menu for "Degree:" with the text "----Select----".
- JTextArea:** A text area for "Experiences:".
- JScrollBar:** A vertical scrollbar on the right side of the "Experiences:" text area.
- JCheckBox:** Checkboxes for "Music" and "Sport" under the "Hobbies:" label.
- JButton:** Buttons labeled "New", "OK", and "Exit" at the bottom.

Objectives

- Use a panel to group some components
- Hide and show those as a whole through this panel
- Disable the exit button on the frame
- The **exit** button allows user close the application
- GUI:



GUI-Demo 1

Employee Details

Code:

Name:

Sex: ☐ Male ☒ Female

Degree: ▼

Hide Exit



GUI-Demo 1

Show Exit

Design

The screenshot displays the NetBeans IDE in Design mode for a project named 'DJA_P1'. The main window is titled 'Employee Details' and contains the following components:

- Code:** A text field with a green label.
- Name:** A text field with a label.
- Sex:** Two radio buttons labeled 'Male' (selected) and 'Female'.
- Degree:** A dropdown menu currently showing 'Ph. D.'.
- Buttons:** 'Hide' and 'Exit' buttons at the bottom.

The [JFrame] - Navigator on the left lists the components in the hierarchy:

- Form GUI_01
 - Other Components
 - bgrpSex [ButtonGroup]
 - [JFrame] (selected)
 - null
 - pEmp [JPanel]
 - null
 - label lbCode [JLabel] (connected to Code label)
 - label jLabel1 [JLabel] (connected to Name label)
 - label jLabel3 [JLabel] (connected to Sex label)
 - label jLabel4 [JLabel] (connected to Degree label)
 - txtCode [JTextField] (connected to Code field)
 - txtName [JTextField] (connected to Name field)
 - rdMale [JRadioButton] (connected to Male radio button)
 - rdFemale [JRadioButton] (connected to Female radio button)
 - cbDegree [JComboBox] (connected to Degree dropdown)
 - btnShowHide [JButton] (connected to Hide button)
 - btnExit [JButton] (connected to Exit button)

A lightbulb icon in the top right of the design view indicates a tip: "The Tools>Palette>Swing/AWT Components menu item all".

Setup Properties

Class	Object name	Properties
JFrame	GUI_01	Layout: null, defaultCloseOperation: DO_NOTHING Title: GUI-Demo 1
JPanel	pEmp	Layout: null, Border: tittleBorder
JLable	lbCode	Opaque: true, background: green, foreground: white horizontalAlignment: RIGHT, text: Code
JLabel	others	Text: Name, Sex, Degree
ButtonGroup	bgrpSex	
JRadioButton	rdMale	buttonGroup: bgrpSex, Text: Male, Selected: true,
JRadioButton	rdFemale	buttonGroup: bgrpSex, Text: Female, Selected: false
JComboBox	cbDegree	Model: Ph. D., Master, Engineer, Bachelor, Technician
Button	btnHideShow	Text: Hide
JButton	btnExit	Text: Exit, mnemonic: E, TooltipText: Exit program

```
public GUI_01() {
    initComponents();
    this.setSize(300,300);
}
```

Constructor

Code:

```
private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(0);
}
```

Code for handling the event on the button **Exit**

```
private void btnShowHideActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (this.btnShowHide.getText().equalsIgnoreCase("hide")){
        this.pEmp.setVisible(false);
        this.btnShowHide.setText("Show");
    }
    else {
        this.pEmp.setVisible(true);
        this.btnShowHide.setText("Hide");
    }
}
```

Code for handling the event on the button **ShowHide**

- View code at the end of the source file, inserted by NetBeans → Comment
- View the contents of the method **initComponents()**, inserted by NetBeans, to explore anonymous classes, then verify them using Window Explorer

- Two Kinds of Applications
- Java Model for Event Management
- About Java Foundation Classes (JFC)
- A Strategy for Designing the GUI
- Using Basic Swing Components

Thank You