

Swing and GUI Programming

Lecture 16
CGS 3416 Spring 2017

April 17, 2017

Graphics classes

- In the original version of Java, graphics components were in the AWT library (Abstract Windows Toolkit)
- When Java 2 was released, a library known as the **Swing components** were introduced with the idea of replacing the older AWT user-interface components (like Button, TextField, TextArea, etc).
 - less dependent on target platform
 - a more robust and flexible library
 - Primary package: `javax.swing`

Java Graphics API (Using Swing Components)

- **Component:** A base class for all non-menu graphic user interface classes.
- **Container:** a base class for container classes. A container is used to group smaller components. The most important containers are:
 - **JApplet** - for holding Applets
 - **JFrame** - for holding GUI components in applications. A window that is on the outer level (not inside another window)
 - **JPanel** - invisible container holding user-interface components. Can be nested, and can be used as canvases for drawing graphics.
 - **JDialog** - for creating dialog boxes (usually temporary popup messages or dialogs for receiving additional info.

JComponent

Base class for all of the lightweight Swing components, which are graphical items placed on the canvases or containers. Its subclasses are the basic elements for constructing GUIs. Here are just a few of the more common elements:

- JButton - for creating push buttons
- JCheckBox - for creating toggle checkboxes
- JMenu - for pop-up menus
- JRadioButton - for radio buttons (made into a group, only one can be selected)
- JLabel - a display area for a short string or image
- JList - a component allowing the user to select from a list
- JOptionPane - a component allowing the user to pop up an easy dialog box as an information message or for user input
- JTextField - component allowing an editable line of text
- JTextArea - multi-line area for displaying text

Helper classes

Helper classes - used by components and containers to control drawing and placing of objects. Some important helper classes (from package `java.awt`):

- `Graphics` - abstract class. Provides graphical context for drawing
- `Color` - used for specifying colors in components and drawings
- `Font` - specify fonts used in `Graphics` drawings
- `FontMetrics` - abstract class. Encapsulates information and properties about the rendering of a font on screen
- `Dimension` - encapsulates width and height of a component in an object

Events

- Event: A signal that something has happened in a program.
Examples: Button clicks, mouse movements, menu selections
- GUI programs generally driven by events, rather than a specific procedural order
- Events are handled with event objects. These are triggered by actions on source objects (components or objects on which the event is generated), and they must implement corresponding event listener interfaces. The listener listens for the event, and invokes an event handler when event occurs.
- `java.util.EventObject`: Base class for event classes in Java

Some examples of event types:

These are just a few examples, not a comprehensive list.

- `ActionEvent` - clicking a button, pressing return on a text field
- `ItemEvent` - clicking a check box, selecting an item
- `WindowEvent` - Closing a window, opening a window
- `ContainerEvent` - component added to a container
- `ComponentEvent` - resizing a component, hiding a component
- `TextEvent` - changing a text value
- `MouseEvent` - clicking the mouse, dragging the mouse
- `KeyEvent` - pressing a key on the keyboard

JOptionPane

- JOptionPane is a class library that makes it easy to pop up a simple dialog box that either provides an information message or asks for a simple input from the user

Simple Dialog Box types

- Message dialog box
 - This is a simple information dialog box
 - To use it, call one of the `showMessageDialog()` methods
 - These are void methods, so just call them by themselves to pop up a message box
- Input dialog box
 - This is a dialog box that allows a user to type in some input
 - To use it, call one of the `showInputDialog()` methods
 - These methods return the input as a `String`. Such calls should capture the returned value
- Confirm dialog box
 - This is a dialog box that presents the user with a Yes/No/Cancel option
 - To use it, call one of the `showConfirmDialog()` methods
 - These methods return the user's answer as an `int` value.
 - The returned integer value can be tested against pre-defined constants, shown in the Field Summary for the class

Using message dialogs

- There are three versions of `showMessageDialog()`
 - `void showMessageDialog(Component parentComponent, Object message)`
 - `void showMessageDialog(Component parentComponent, Object message, String title, int messageType)`
 - `void showMessageDialog(Component parentComponent, Object message, String title, int messageType, Icon icon)`

Using message dialogs

- The parameters:
 - `parentComponent`: indicates what component this dialog will pop up over. Use null to make the message not connected to any other component
 - `message`: This is the actual message being displayed.
 - `title`: The string that will appear in the pop-up window's title bar
 - `messageType`: Use one of these static constants from the class. Except for plain message, each one uses a special icon on the message box
 - `PLAIN_MESSAGE`
 - `ERROR_MESSAGE`
 - `INFORMATION_MESSAGE`
 - `WARNING_MESSAGE`
 - `QUESTION_MESSAGE`
 - `icon`: A custom icon to use

Note that all calls use the first two parameters. The others can be used optionally

Using input dialogs

- There are several versions of `showInputDialog`. Some of these have very similar options as the `showMessageDialog` methods
- However, the easiest form is simply to pass in a message prompt as a single parameter. Example:

```
String s1;  
s1 = JOptionPane.showInputDialog("Please enter an integer:");
```
- Note that in the above call, we captured the return value into a variable. The `showInputDialog` call will always return the entered value
- Also note that this value always comes back as a `String`
- If you want to save the entered data as a different type, you will need to convert it

```
int value = Integer.parseInt(s1); // converts s1 into an integer
```