Swing

Part 4: Dialog Boxes

Chapter 11, Core Java, Volume I

Contents

- Dialog Boxes
- Option Dialogs
- Creating Dialogs
- Example: Dialog Box
- Data Exchange
- Example: Data Exchange
- File Dialogs

Dialog Boxes

- Dialog box pops up from application window.

- Types of dialog boxes
  - *Modal dialog*: Must be completed before other windows become active again.
  - *Modeless dialog*: Stays in place for as long as needed.

- Using standard dialog boxes
  - Option panes (JOptionPane)– ready-made simple modal dialogs
  - File chooser(JFileChooser)/Color chooser(JColorChooser) etc.

- Creating general dialog boxes
  - To show message
  - To get user input

Option Dialogs

- JOptionPane class has ready-made dialogs for a single piece of information:
  - `showMessageDialog` shows a message.
  - `showConfirmDialog` gets a confirmation such as OK/Cancel.
  - `showOptionDialog` makes user select from a set of options.
  - `showInputDialog` gets an input string.

- Follow this recipe:
  1. Choose the dialog type (message, confirm, option, input).
  2. Choose an icon (error, information, warning, question, none, or custom).
  3. Choose a message (string, icon, custom component, or a stack of them).
  4. For a confirmation dialog, choose the option type (default, Yes/No, Yes/No/Cancel, or OK/Cancel).
  5. For an option dialog, choose the options (strings, icons, or custom components) and the default option.
  6. For an input dialog, choose between a text field and a combo box.

- Example:

```
int selection = JOptionPane.showConfirmDialog(parent, "Message", "Title",
   JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE);
```

Option Dialogs

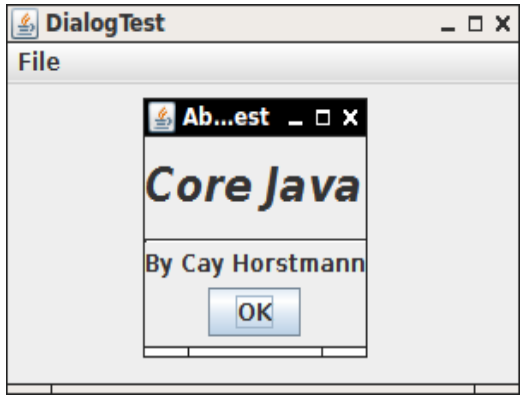
- Return values
  - showMessageDialog – None
  - showConfirmDialog – an integer representing the chosen option
    - OK_OPTION
    - CANCEL_OPTION
    - YES_OPTION
    - NO_OPTION
    - CLOSED_OPTION (user closed the dialog)
  - showOptionDialog – an integer representing the chosen option or the value CLOSED_OPTION
  - showInputDialog – the string that the user supplied or selected

Option Dialogs

```
frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        int result = JOptionPane.showConfirmDialog(frame,
            "Do you want to Exit ?", "Exit Confirmation : ",
            JOptionPane.YES_NO_OPTION);
        if (result == JOptionPane.YES_OPTION)
            System.exit(0);
    }
});
```

Creating Dialogs

- To implement a dialog box:
  1. Extends JDialog class
  2. In the constructor, call the constructor of the super class
  3. Add the user interface components
  4. Add the event handlers
  5. Set the size for the dialog box




```
public AboutDialog extends JDialog
{
   public AboutDialog(JFrame owner)
   {
      super(owner, "About DialogTest", true);
      add(new JLabel( "<html><h1><i>Core Java</i></h1><hr>By Cay Horstmann</html>"),
                        BorderLayout.CENTER);
      . . .
      setSize(250, 150);
   }
}
```

modality

Creating Dialogs

- To create and display the dialog:

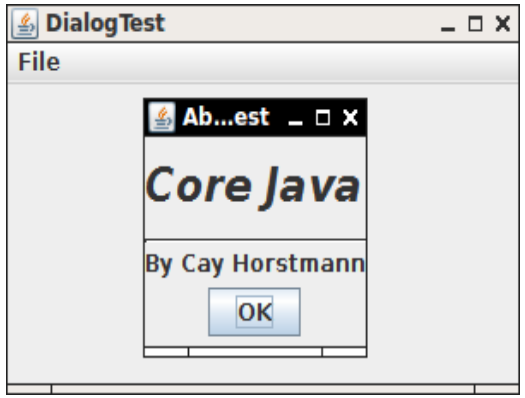
```
JDialog dialog = new AboutDialog(this);
dialog.setVisible(true);
```

- To create the dialog only once (singleton):

```
JDialog dialog;
...
if(dialog == null)
   dialog = new AboutDialog(this);
dialog.setVisible(true);
```

- When the user selects OK, hide the dialog:

```
JButton ok = new JButton("OK");
ok.addActionListener(event -> setVisible(false));
```



Example: Dialog Box (Listing 11.11-12)

```
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

public class DialogFrame extends JFrame
{
   private static final int DEFAULT_WIDTH = 300;
   private static final int DEFAULT_HEIGHT = 200;
   private AboutDialog dialog;

   public DialogFrame()
   {
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
```

```
      // Construct a File menu.
      JMenuBar menuBar = new JMenuBar();
      setJMenuBar(menuBar);
      JMenu fileMenu = new JMenu("File");
      menuBar.add(fileMenu);

      // Add About and Exit menu items.
      // The About item shows the About dialog.
      JMenuItem aboutItem = new JMenuItem("About");
      aboutItem.addActionListener(event -> {
         if (dialog == null) // first time
            dialog = new AboutDialog(DialogFrame.this);
         dialog.setVisible(true); // pop up dialog
      });
      fileMenu.add(aboutItem);
```

Example: Dialog Box (Listing 11.11-12)

```
    // The Exit item exits the program.
    JMenuItem exitItem = new JMenuItem("Exit");
    exitItem.addActionListener(event -> System.exit(0));
    fileMenu.add(exitItem);
  }
}
```

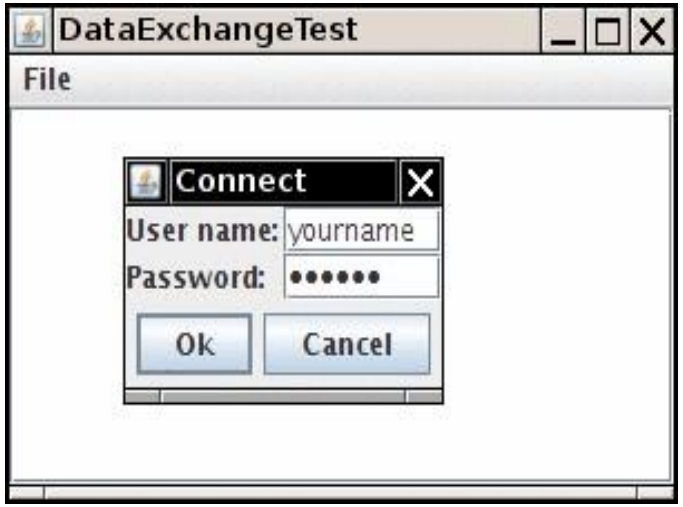
Example: Dialog Box (Listing 11.11-12)

```
public class AboutDialog extends JDialog
{
  public AboutDialog(JFrame owner)
   {
     super(owner, "About DialogTest", true);
     add( new JLabel(
              "<html><h1><i>Core Java</i></h1><hr>By Cay Horstmann</html>"), BorderLayout.CENTER);
     JButton ok = new JButton("OK");   // OK button closes the dialog
     ok.addActionListener(event -> setVisible(false));
     JPanel panel = new JPanel();
     panel.add(ok);
     add(panel, BorderLayout.SOUTH);  // by default, BorderLayout
     pack();
   }
}
```

Data Exchange

- Normally, dialog gets complex user input.
- Data needs to be transmitted back to the application.
- Make a class for the data:

```
public class User { . . .}
```



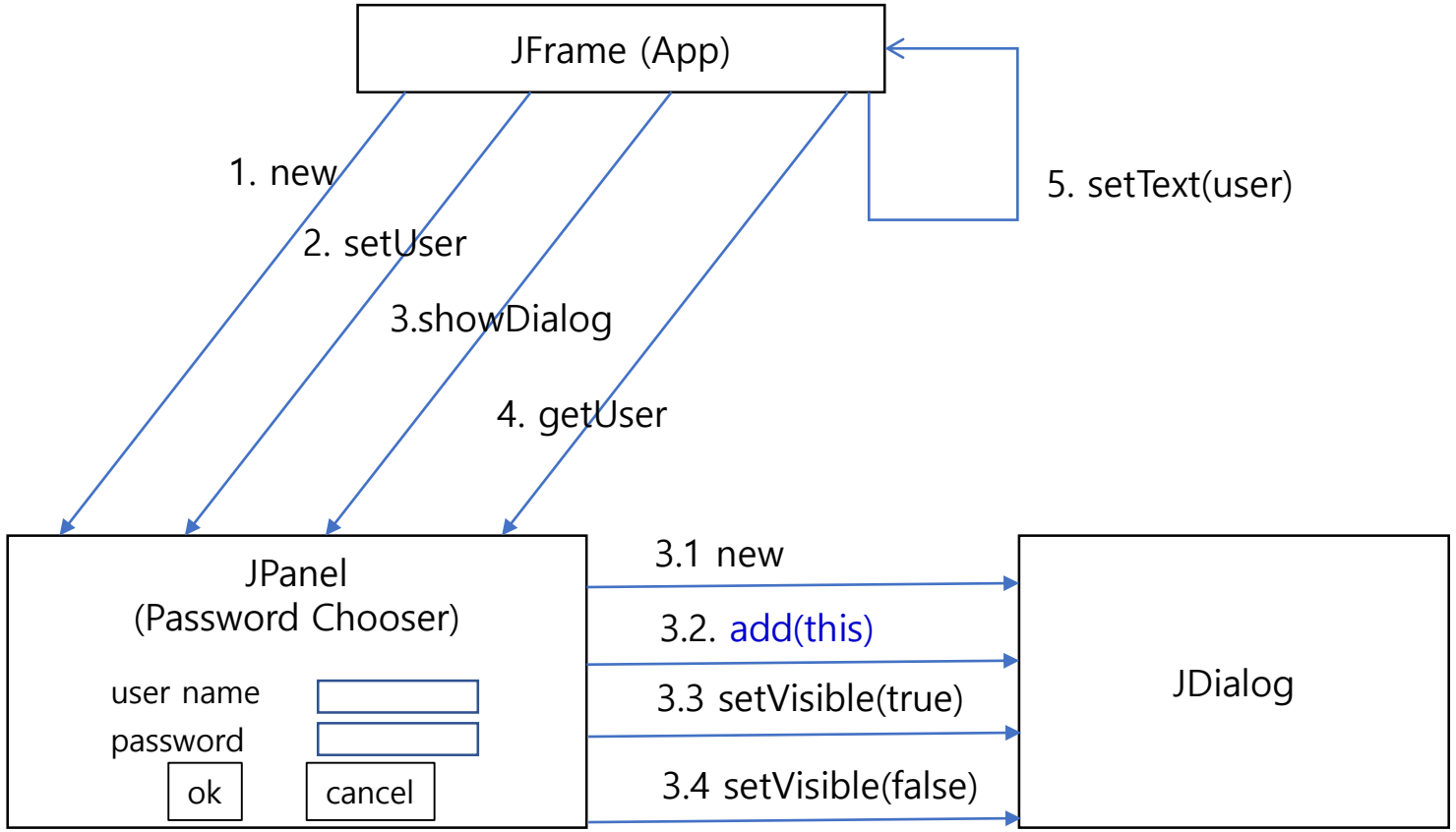
- Make methods for setting, getting data:

```
public class PasswordChooser extends JPanel
{
        public void setUser(User u) { . . . }
        public User getUser() { . . . }
        . . .
}
```

- Pop up the dialog:

```
public boolean showDialog()
{
   JDialog dialog = new JDialog(frame, true); // modal
   dialog.add(panel);
   dialog.pack();
   dialog.setVisible(true);  // wait until the dialog is no longer visible
}
```

Data Exchange



3.3.1 user types nane & password
3.3.2 user enters OK

Example: Data Exchange (Listing 11.13-14)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * A frame with a menu whose File->Connect action shows a password dialog.
 */
public class DataExchangeFrame extends JFrame
{
   public static final int TEXT_ROWS = 20;
   public static final int TEXT_COLUMNS = 40;
   private PasswordChooser dialog = null;
   private JTextArea textArea;
```

Example: Data Exchange (Listing 11.13-14)

```
public DataExchangeFrame()
{
  JMenuBar mbar = new JMenuBar();
   setJMenuBar(mbar);
   JMenu fileMenu = new JMenu("File");          // construct a File menu
   mbar.add(fileMenu);
  JMenuItem connectItem = new JMenuItem("Connect");
   connectItem.addActionListener(new ConnectAction());
   fileMenu.add(connectItem);
  JMenuItem exitItem = new JMenuItem("Exit");
   exitItem.addActionListener(event -> System.exit(0));
   fileMenu.add(exitItem);
  textArea = new JTextArea(TEXT_ROWS, TEXT_COLUMNS);
   add(new JScrollPane(textArea), BorderLayout.CENTER);
   pack();
}
```

Example: Data Exchange (Listing 11.13-14)

```
   private class ConnectAction implements ActionListener
   {
      public void actionPerformed(ActionEvent event)
      {
        if (dialog == null) dialog = new PasswordChooser();      // if first time, construct dialog
        dialog.setUser(new User("yourname", null));               // set default values
        if (dialog.showDialog(DataExchangeFrame.this, "Connect")) // pop up dialog and if OK is pressed
        {
           // if accepted, retrieve user input
           User u = dialog.getUser();
           textArea.append("user name = " + u.getName() + ", password = "
                 + (new String(u.getPassword())) + "\n");
        }
      }
   }
}
```

Example: Data Exchange (Listing 11.13-14)

```
/**
 * A password chooser that is shown inside a dialog
 */
public class PasswordChooser extends JPanel
{
   private JTextField username;
   private JPasswordField password;
   private JButton okButton;
   private boolean ok;
   private JDialog dialog;

   public PasswordChooser()
   {
      setLayout(new BorderLayout());
```

Example: Data Exchange (Listing 11.13-14)

```
// construct a panel with user name and password fields
JPanel panel = new JPanel();
panel.setLayout(new GridLayout(2, 2));
panel.add(new JLabel("User name:"));
panel.add(username = new JTextField(""));
panel.add(new JLabel("Password:"));
panel.add(password = new JPasswordField(""));
add(panel, BorderLayout.CENTER);

// create Ok and Cancel buttons that terminate the dialog
okButton = new JButton("Ok");
okButton.addActionListener(event -> {
   ok = true;
   dialog.setVisible(false);
});
```

Example: Data Exchange (Listing 11.13-14)

```
    JButton cancelButton = new JButton("Cancel");
    cancelButton.addActionListener(event -> dialog.setVisible(false));

    // add buttons to southern border
    JPanel buttonPanel = new JPanel();
    buttonPanel.add(okButton);
    buttonPanel.add(cancelButton);
    add(buttonPanel, BorderLayout.SOUTH);
}
public void setUser(User u)
{    username.setText(u.getName());
}
public User getUser()
{
    return new User(username.getText(), password.getPassword());
}
```

Example: Data Exchange (Listing 11.13-14)

```
public boolean showDialog(Component parent, String title)
{
   ok = false;

   // locate the owner frame

   Frame owner = null;
   if (parent instanceof Frame)
      owner = (Frame) parent;
   else
      owner = (Frame) SwingUtilities.getAncestorOfClass(Frame.class, parent);
```

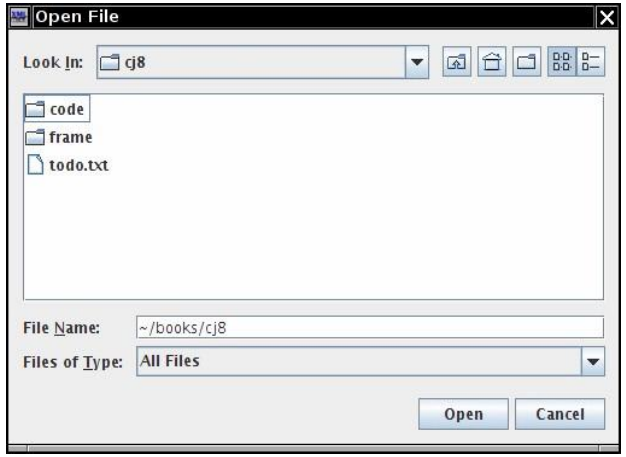
Example: Data Exchange (Listing 11.13-14)

```
    // if first time, or if owner has changed, make new dialog
    if (dialog == null || dialog.getOwner() != owner)
    {
      dialog = new JDialog(owner, true);
      dialog.add(this);
      dialog.getRootPane().setDefaultButton(okButton);
      dialog.pack();
    }

    // set title and show dialog
    dialog.setTitle(title);
    dialog.setVisible(true); // wait until the dialog is no longer visible (OK or Cancel)
    return ok;  // ok is true if OK button, otherwise false
  }
}
```

File Dialogs

- Often want to ask user for a file name to open or save.



- Make a JFileChooser object:
  ```
  JFileChooser chooser = new JFileChooser();
  ```
- Set the directory:
  ```
  chooser.setCurrentDirectory(new File("."));
  ```
- If you have a default file name:
  ```
  chooser.setSelectedFile(new File(filename));
  ```
- To enable the user to selects multiple files:
  ```
  chooser.setMultiSelectionEnabled(true);
  ```
- If you want to restrict the files, set the file filter:
  ```
  chooser.setFileFilter(new FileNameExtensionFilter("Image files", "gif", "jpg"));
  ```
- By default, a user can select only files. If you want a user to select directories:
  ```
  chooser.setFileSelectionMode(JFileChooser.FILE_AND_DIRECTORY);
  ```

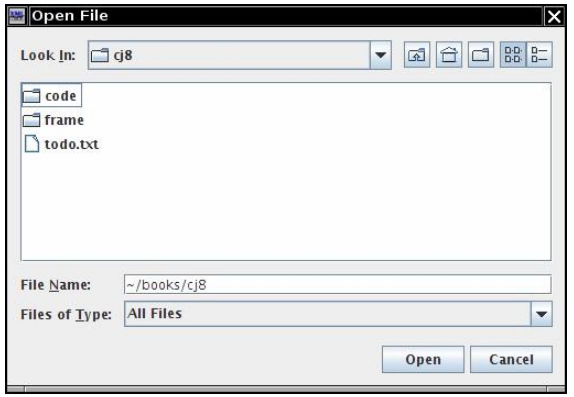
File Dialogs

- Pop up the dialog:

  int result = chooser.showOpenDialog(parent);

  or

  int result = chooser.showSaveDialog(parent);



- Return values:

  JFileChooser.APPROVE_OPTION

  JFileChooser.CANCEL_OPTION

  JFileChooser.ERROR_OPTION

- If the result is JFileChooser.APPROVE_OPTION, get the file:

  File file = chooser.getSelectedFile();

  File[] files = chooser.getSelectedFiles();