JOptionPane Dialogs

javax.swing.JOptionPane is a class for creating dialog boxes.

Has both static methods and instance methods for dialogs.

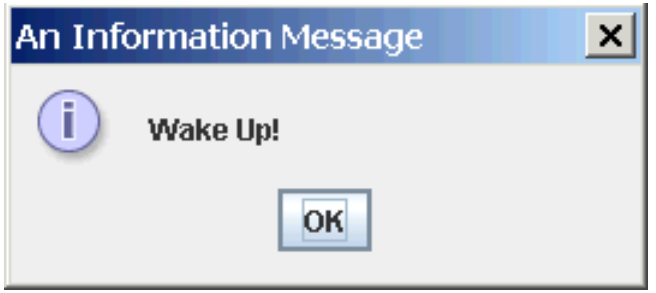
Easy to create 4 Common Dialogs:

- Message Dialog - display a message
- Confirm Dialog - Yes, No, or Cancel dialog
- Choice Dialog - click a button to make choice
- Input Dialog - request input, return a string

Dialog Examples

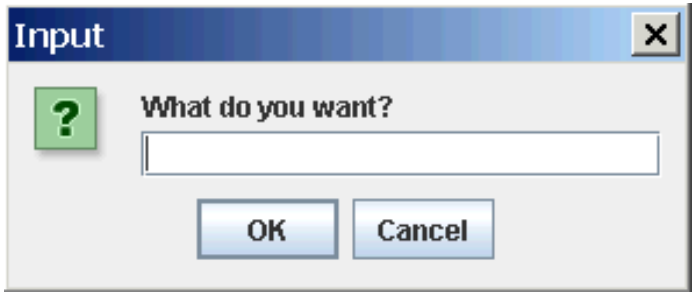
- JOptionPane is a class for showing dialog boxes.
- It can display dialogs like these:

```
showMessageDialog(...)
```

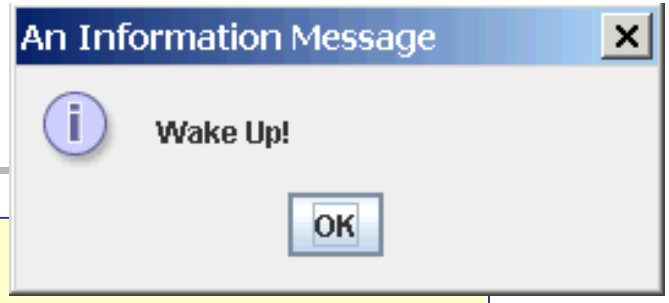



```
showConfirmDialog(...)
```

```
reply = showInputDialog(...)
```



Message Dialog




```
// basic message dialog
JOptionPane.showMessageDialog( null,
    "Wake Up!");

// message dialog with a title and message type.
JOptionPane.showMessageDialog( null, "Wake Up!",
    "An Information Message",
        JOptionPane.INFORMATION_MESSAGE );
```

Syntax:

static void showMessageDialog(Component parent, Object message)

static void showMessageDialog(Component parent ,
    Object message ,
    String title_line ,
    int messageType)

messageType is one of: INFORMATION_MESSAGE, QUESTION_MESSAGE, WARNING_MESSAGE, ERROR_MESSAGE

Modal or Non-Modal Dialog?

Graphics frameworks support 2 ways of using dialogs:

Modal Dialog - user must close the dialog (click OK, Cancel, etc) before he can work in the parent window. Example: "Open file" dialog.

Non-modal Dialog - user can go back and work in the parent window while the dialog is still open.

```
JOptionPane.showMessageDialog( parent, "Wake Up!" );
```

First parameter is a reference to **parent window**.

if parent == null, it displays a non-modal dialog.

if parent != null, it displays a modal dialog

Multi-line Message Dialog

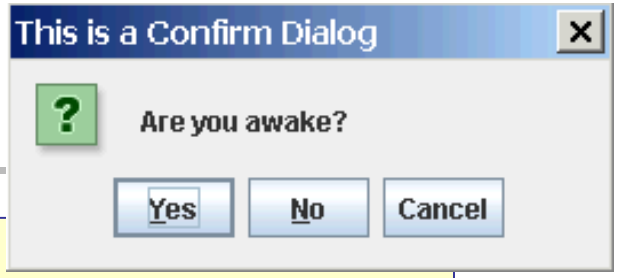
```
String message = "Please give the customer:\n"
    + "2 Rice balls\n"
    + "1 Green tea\n"
    + "5 Tofu sticks");

JOptionPane.showMessageDialog( null,
    message, "Order Details",
        JOptionPane.INFORMATION_MESSAGE );
```

Use correct singular / plural

Avoid errors like this:

`You have 1 new messages`

Confirm Dialog




```
int choice;
choice = JOptionPane.showConfirmDialog( null,
        "Are you awake?",
        "This is a Confirm Dialog",
        JOptionPane.YES_NO_CANCEL_OPTION );

if ( choice == JOptionPane.NO_OPTION )
    JOptionPane.showMessageDialog( null, "Liar!" );
```

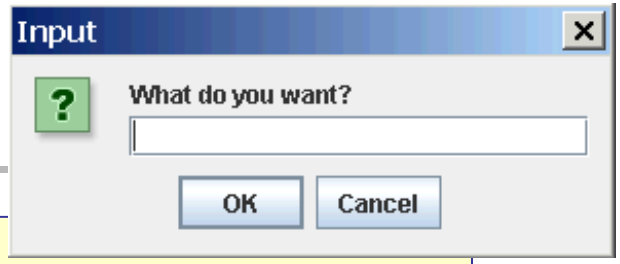
Syntax:

```
static int showConfirmDialog(
      Component parent,
      Object message ,
      String title_line,
      int optionType )
```

optionType is: YES_NO_OPTION or YES_NO_CANCEL_OPTION

return value is: YES_OPTION, NO_OPTION, or CANCEL_OPTION

Input Dialog



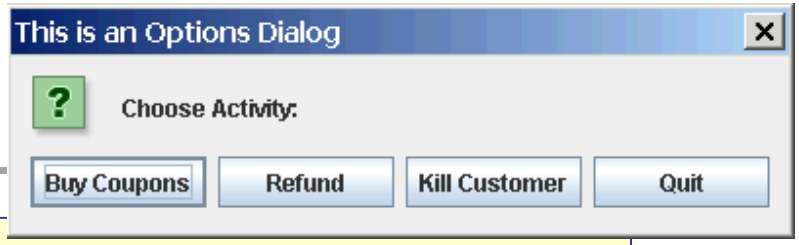

```
String reply;
reply = JOptionPane.showInputDialog( null,
        "What do you want?",
        "This is an Input Dialog",
        JOptionPane.QUESTION_MESSAGE );

if ( reply == null ) /* user pressed CANCEL */ ;
else doSomething( reply );
```

Syntax:

```
static String showInputDialog(
      Component parent,
      Object message,
      String title_line,
      int messageType )
```

return value is `null` if "Cancel" pressed; else string typed in textbox.

Option Dialog




```
String [] choices = { "Buy Coupons",
    "Refund", "Kill Customer", "Quit"};
int reply =
     JOptionPane.showOptionDialog(
     null,                              // parent
        "Choose Action:",                 // message
        "This is an Options Dialog",  // title string
        JOptionPane.YES_NO_OPTION,       // useless
        JOptionPane.QUESTION_MESSAGE, // msg type
        null,                             // no icon
        choices,                  // array of choices
        choices[0]                // default choice
     );
switch( reply ) {
    case 0:  couponDialog( ) ; break;
    case 1:  refundDialog( ) ; break;
    case 2:  killCustomer( ) ; break;
    default:  confirmQuit( ) ;
}
```

JOptionPane Usage

- These 4 dialogs are static methods: you don't need to create an *instance* of JOptionPane.
- JOptionPane also has *instance methods* for more control over dialogs. See the Java API.
- JOptionPane is in javax.swing:

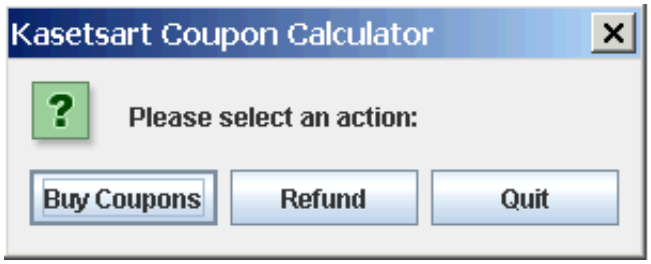
```
import javax.swing.JOptionPane;
```

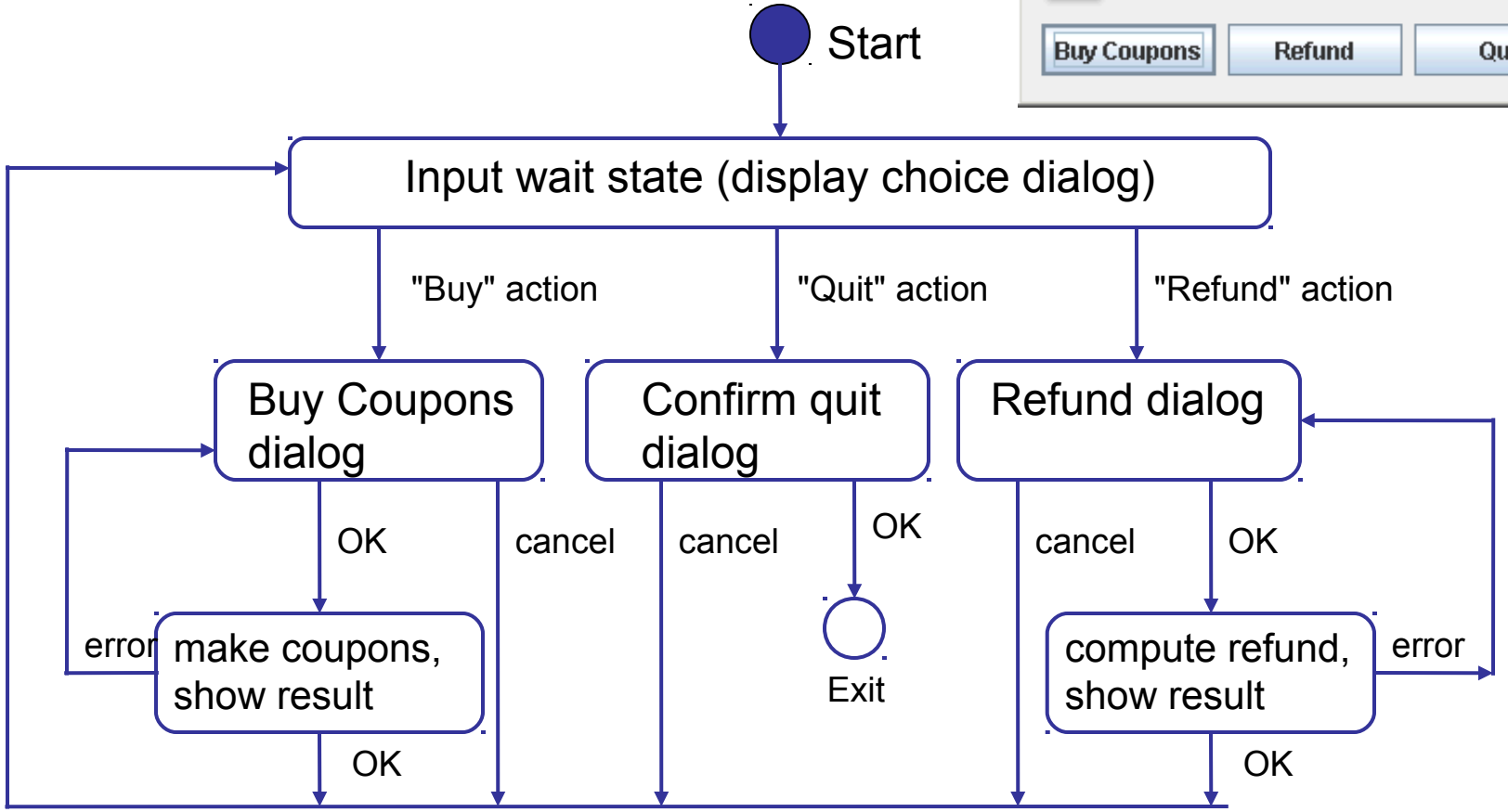
or:

```
import javax.swing.*;
```

KU Coupons Design using Dialogs

- Draw a UML state chart diagram.



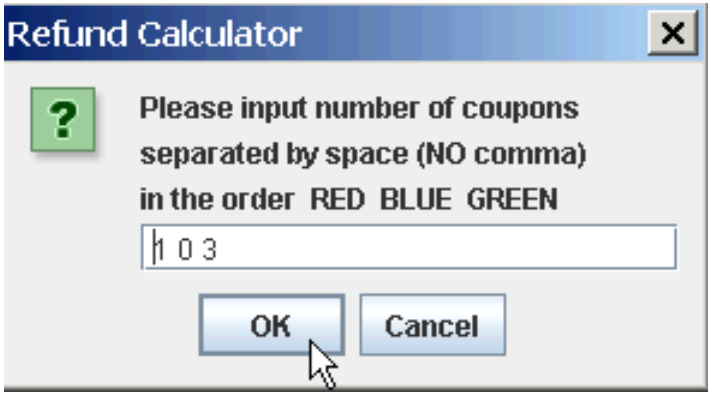


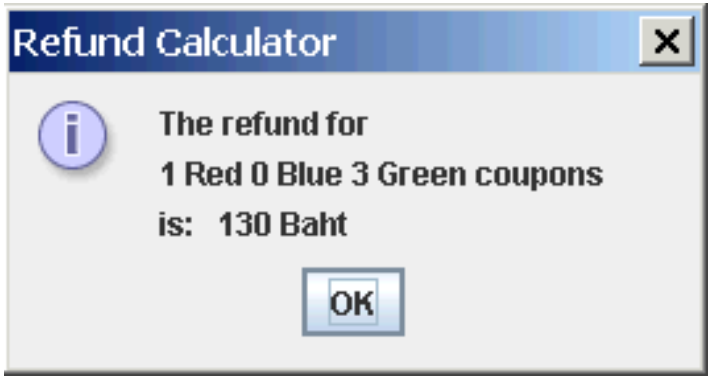
Bad Input Dialog

```
String red = JOptionPane.showInputDialog( null,
          "How many red?",
       "Red Dialog",
       JOptionPane.QUESTION_MESSAGE );
String blue = JOptionPane.showInputDialog( null,
          "How many blue?",
       "Blue Dialog",
       JOptionPane.QUESTION_MESSAGE );
String green = JOptionPane.showInputDialog( null,
          "How many green?",
       "Green Dialog",
       JOptionPane.QUESTION_MESSAGE );
// now parse the Strings
```

- Too many dialogs!
- No feedback from previous dialog (how many red and blue did he enter?).

Better Input Dialog





Display input data for verification and feedback.

Hints

- How to convert a **String** containing a number like "123" to a number?

  Use results of Homework 2, or use Scanner.

- How to process a String containing several numbers, such as "12 25 7.5 4"?

  Look at the API for the "Scanner" class.

  There is a *constructor* that accepts a String argument

  `Scanner( String source );`

```
String reply = /* string containing numbers */ ;
Scanner scan = new Scanner( reply );
/* now use Scanner methods to read the values */
```

Crashable Input Method

```
private void withdrawDialog( ) {
   String reply =
       JOptionPane.showInputDialog( null,
           "Please input amount to withdraw",
        "Withdraw Calculator",
        JOptionPane.QUESTION_MESSAGE );

   Scanner scan = new Scanner( reply );
   // if user presses CANCEL this will crash!

   amount = scan.nextDouble( );
      // crash! if no nextDouble

   // what if user inputs TOO MUCH data?
```

Writing a Crash-proof Method

```
private void withdrawDialog( ) {
    String reply = JOptionPane.showInputDialog( null,
        . . . , // same as previous slide
    JOptionPane.QUESTION_MESSAGE );

    if ( reply == null ) return; // dialog cancelled
    Scanner scan = new Scanner( reply );

    boolean inputOK = true;
    // test before reading
    if ( scan.hasNextDouble() ) red = scan.nextInt( );
    else inputOK = false;

    // test for too much data
    inputOK = inputOK && ( ! scan.hasNext() );
    if ( inputOK ) { /* compute withdraw */ }
    else { /* display error message */ }
```

Making Your Program Crash-proof

- Don't assume input exists or is of expected type.
- Ask someone else to test your program!
- Tester should be malicious.