Week 5 - Wednesday

COMP 2000

Last time

- What did we talk about last time?
- Exam 1!

Questions?

Project 2

GUIs

GUIs

- A graphical user interface (GUI) is a way of interacting with a program using a mouse, buttons, menus, windows, etc.
- Before COMP 1600, that was probably the only way you interacted with programs
- The programs in COMP 1600 were command line interface (CLI) programs
- We put off GUIs until now…
  - Because they're much more complicated!

AWT

- The oldest Java GUI library is the AWT (abstract windowing toolkit)
- AWT objects are connected to OS GUI objects (called widgets)
- For this reason, AWT objects
  - Look and act differently on different OSes
  - Are slow
  - Can't be styled with different looks and feels
- Even though AWT isn't used much, later libraries used components of it, requiring the **`java.awt`** package in many cases

Swing

- A newer GUI library is called Swing
- Swing objects are independent from OS widgets
- For that reason, Swing objects
  - Look and behave consistently across different OSes
  - Are relatively fast
  - Can be styled with different looks and feels (which can make them *look* like they're natural parts of the OS)
- To differentiate between AWT and Swing objects, Swing objects always have a J at the beginnings of their names
  - **`Button`** (AWT) vs. **`JButton`** (Swing)

JavaFX

- JavaFX was intended as a replacement for Swing
- It uses a completely different system for layout that relies on XML files
  - But it still references AWT and Swing libraries
- In spite of some cool features, it is not included in standard Java
- So we're not going to cover it

Full GUIs

- Chapter 15 explains how to construct full GUIs:
  - Windows
  - Buttons
  - Fields
  - Labels
  - Layout
- Today, we're only talking about easy additions to our existing programs
- These can:
  - Display a message
  - Ask a question

JOptionPane

JOptionPane

- **`JOptionPane`** class provides static methods for:
  - Displaying a message
  - Asking a question
- Although it is possible to create a **`JOptionPane`** object, you almost never do
- Just call the static methods
  - Which means typing a lot of **`JOptionPane.`**

showMessageDialog()

- Perhaps the simplest thing you can do with **JOptionPane** is have it display a message in a dialog window
- By default these dialogs are **modal**, meaning that you have to deal with it before the program continues
- To display a message, the method signature is:

```
void showMessageDialog(Component parent,
    Object message)
```

- The **parent** is the window that waits for you to finish with the dialog, but you can pass in **null** if you don't have a parent window
- Even though the parameter is an **Object**, you usually pass in a **String**

showMessageDialog() example

- To display **"This is a message."** you could call the following:

```
JOptionPane.showMessageDialog(null,
  "This is a message.");
```

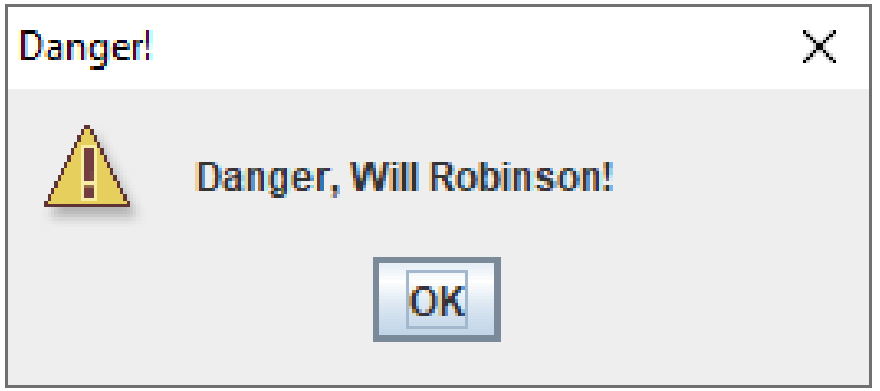
Adding a title

- Most **`JOptionPane`** methods have many overloads
- If you want to put a title on the window, you can pass it in as the third parameter
- But this overloaded method also requires an **`int`** parameter that says what kind of message you want
- To add the title **`"Window Title"`**, you might call the following method:


```
JOptionPane.showMessageDialog(null,
  "This is a message.", "Window Title",
  JOptionPane.PLAIN_MESSAGE);
```

Different icons

- You can choose an icon associated with one of the following constants:
  - **ERROR_MESSAGE**
  - **INFORMATION_MESSAGE**
  - **WARNING_MESSAGE**
  - **QUESTION_MESSAGE**
  - **PLAIN_MESSAGE**




```
JOptionPane.showMessageDialog(null,
   "Danger, Will Robinson!", "Danger!",
   JOptionPane.WARNING_MESSAGE);
```

showConfirmDialog()

- What if you don't just want to display a message?
- You could also display a prompt with yes, no, and cancel buttons, using the following method signature

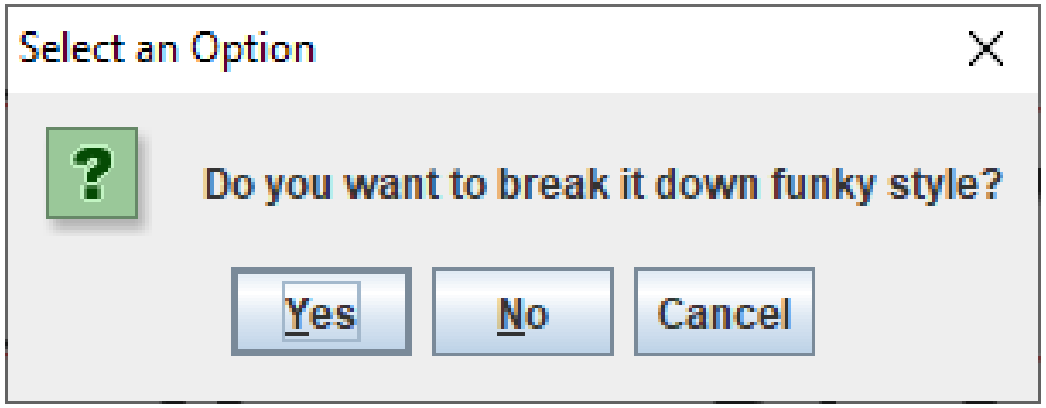
```
int showConfirmDialog(Component parent,
    Object message)
```

- This method will display **message** and return one of the following **int** constants inside **JOptionPane**:
  - **YES_OPTION**
  - **NO_OPTION**
  - **CANCEL_OPTION**

showConfirmDialog() example

```
int answer = JOptionPane.showConfirmDialog(null,
    "Do you want to break it down funky style?");
if(answer == JOptionPane.YES_OPTION)
    JOptionPane.showMessageDialog(null, "Dope!");
else
    JOptionPane.showMessageDialog(null, "Weak!");
```

- Hitting the X in the corner is the same as Cancel



showOptionDialog()

- What if you want options other than yes, no, and cancel?
- No problem, you can supply an array of **`Object`** values to **`showOptionDialog()`**
  - It returns the index in the array of options
- Generally, these values will be **`String`** objects, but they could also be Swing widgets
- The signature is…long:

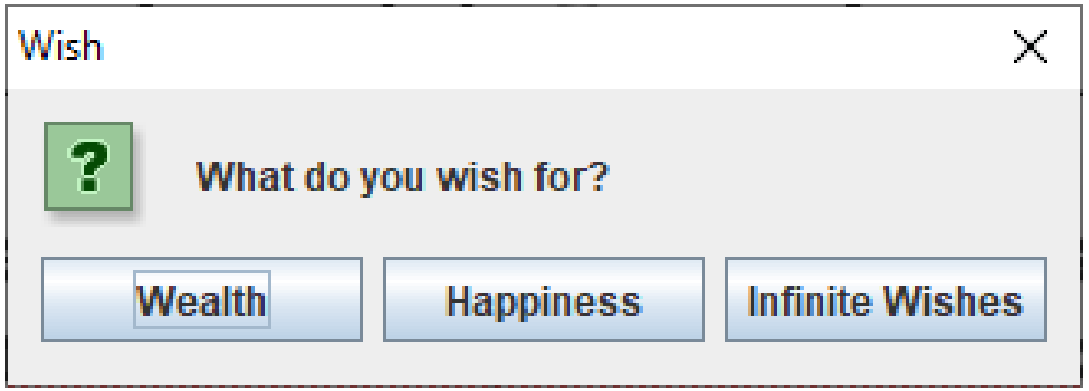
```
int showOptionDialog(Component parent,
    Object message,  String title,
    int optionType, int messageType, Icon icon,
    Object[] options, Object initialValue)
```

showOptionDialog() example

- Here's an example showing a dialog that allows a user to choose between Wealth, Happiness, and Infinite Wishes

```
String[] options = {"Wealth", "Happiness", "Infinite Wishes"};
int answer = JOptionPane.showOptionDialog(null,
  "What do you wish for?",  "Wish", JOptionPane.DEFAULT_OPTION,
  JOptionPane.QUESTION_MESSAGE, null, options, null);
```

- Note that many parameters can be **`null`**: parent, icon, default option



showInputDialog()

- A more flexible option for input is **showInputDialog()**
- It allows the user to type arbitrary text into a box
- Be careful with the return value
  - If they cancel, it's **null**
  - They might put crazy spaces at the beginning and end (use **trim()**)
  - Numbers have to be converted from **String** values
- Signature:

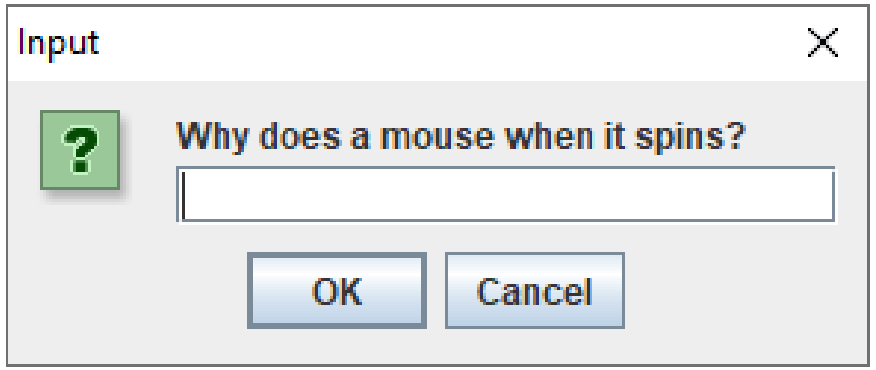
```
String showInputDialog(Component parent,
    Object message)
```

showInputDialog() example

- This input dialog asks a pressing question

```
String answer =
  JOptionPane.showInputDialog(null,
  "Why does a mouse when it spins?");
```

- As with other methods, there are overloaded versions that allow for titles, icons, and other options



Coding example

- Let's write a program that makes the user guess a number between 1 and 100
- Each time they guess a number, we use a dialog to display:
  - You got it! (if they were correct)
  - Too high! (if they were too high)
  - Too low! (if they were too low)
  - Not a number! (if they didn't enter a correctly formatted integer)
- The program quits when they guess the answer

Upcoming

Next time…

- Making windows with **`JFrame`**
- Other Swing widgets:
  - **`JTextField`**
  - **`JButton`**
  - **`JLabel`**

Reminders

- **Amazon Alexa Meetup**
  - Discussion of Alexa technology
  - Free pizza!
  - Tomorrow from 6-8 p.m., here at The Point (in the large meeting area)
  - RSVP here: https://www.meetup.com/Columbus-Amazon-Alexa-Meetup/events/bnlzznybcdbrb/
- Reading Chapter 15
- Keep working on Project 2