# More GUI

# Recap

- Create a class that is both a window and a listener
  - extends JFrame
  - implements ActionListener
- Define instance variables for all the widgets.
- In the Constructor
  - Set the window properties
  - Create the widgets.
  - Position the widgets on the window.
  - Set this as their listener.
  - Make the window visible.

# Recap (2)

- Define an actionPerformed method.
  - Use the ActionEvent parameter to find out which widget generated the event.
  - Program the code to be done when an event arrives.
- Define a main program.
  - Create a window object, which will start things off.

# Disabling Widgets

- Widgets such as buttons can be disabled.
  - The are greyed out
  - They do not accept any input.
  - `button1.setEnabled(false);`
- They can then be enabled again.
  - `button1.setEnabled(true);`
- This allows us to control which input is allowed.
- Widgets are enabled when they are created.

# Terminating The Program

- The user can stop the program by clicking on a button labled “Quit”.
  - The cleanest way to exit from a program is
  - `System.exit(0);`
  - We can use a number other than 0 to indicate an error.
  - This can be the `actionPerformed` code for a quit button.

- The user can also close the window.
  - We can tell the Frame to exit by writing
  - `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
  - We put this line in the constructor when setting up the wndow basic properties.

# Java Utility Dialog Boxes

# JOptionPane

- Java has a series of utilities windows that we can use in our program.
- They are displayed as additional windows that require immediate attention.
- They disappear once we have dealt with them.
- They are used for.
  - Error and informative messages
  - Diagnostics.
  - Getting a yes/no decision from the user.
- We call a method of the static class `JOptionPane`.

# showMessageDialog

- This method displays a message.
- The message stays until we press the OK button.
- Example:

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

# showMessageDialog (2)

- The first parameter is always `null`.
- The last parameter determines the icon shown. Alternatives are:
  - `INFORMATION_MESSAGE`
  - `WARNING_MESSAGE`
  - `QUESTION_MESSAGE`



# showConfirmDialog

- Lets the user chose yes, no, cancel or close the window.

```
int result =JOptionPane.showConfirmDialog(null,
"Message");
```

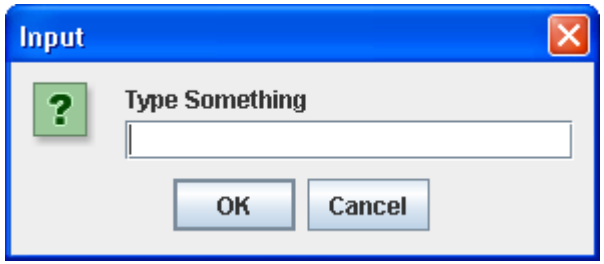
- Possible results are:
  - `JOptionPane.YES_OPTION`
  - `JOptionPane.NO_OPTION`
  - `JOptionPane.CANCEL_OPTION`
  - `JOptionPane.CLOSED_OPTION`

# showInputDialog

- The lets the user input a string.

```
String s = JOptionPane.showInputDialog(null,
   "Type Something");
```

- A null string is returned if Cancel is pressed or the dialog box is closed.



# Panels

# Windows Within Windows

- We can divide our window into sub-windows.
- A sub-window can be placed anywhere that a widget can be placed.
  - We can put a sub-window in the “North” position of our main window.
  - We can then place several widgets in this sub-window, rather than just one.
- This way we can get more than 5 widgets in our main window.
- The default layout for a sub-window is Flow Layout.
  - Widgets are displayed side by side.
- We can divide a sub-window into sub-sub-windows.
  - And so on.

# JPanel

- The Java class for a sub-window is called `JPanel`.
- A sub-sub-window is also a JPanel
- A JFrame is only used for the main window.
- The following code adds to our example by
  - creating 2 new buttons, `button3` and `button4`
  - a `JPanel pan`
  - which is added to the `“North”` position of the `JFrame`.
- We need to import `java.awt.*`
- We also need to add code to `actionPerformed` so that it recognises the two new buttons.

# The Code

```
button3 = new JButton("Three");
button4 = new JButton("Four");
button3.addActionListener(this);
button4.addActionListener(this);
JPanel pan = new JPanel();
pan.add(button3);
pan.add(button4);
add(pan, "North");
```

# Running The Program

- This is the window produced.


- And this is how it looks if we add to the "`West`" position.

# Layout Managers

# Default Layout Managers

- We have already met two layout managers.
  - BorderLayout with JFrame.
  - FlowLayout with JPanel.
- We can replace these defaults with other more appropriate ways of positioning widgets.
- We will look at
  - GridLayout
- But first, lets recap the defaults

# Border Layout / Flow Layout

- A `BorderLayout` is the default for a `JFrame`.
- It has 5 regions, Center, North, South, East and West.
- We position a widget by
  - `add(widget, String position);`

- A `FlowLayout` is the default for a `JPanel`.
- It fits widgets horizontally side by side and centres them.
- They are positions in the order we add them.
  - `add(widget);`

# Grid Layout

- A `GridLayout` divides the window or sub-window into a grid of equally sized positions.
- We can put one widget in each position.
- We must specify how many rows and columns the grid has.
- The widgets are added left to right and top to bottom in the order they are added.
  - `add(widget);`
- We create a `GridLayout` object first.
- We then pass it as a parameter when we create the `JPanel`.

```
GridLayout grid = new GridLayout(2, 3);
JPanel pan = new JPanel(grid);
```

# Running The Program

- This is what it looks like after we add buttons Three to Eight.

# Changing The Default for JFrame

- We can change the default layout manager for a JPanel by creating the layout manager we want and passing it as a parameter in the constructor.
- A JFrame is automatically created with a BorderLayout.
- We override it by creating the new layout object and then passing it as a parameter to setLayout.
- The following code is in the set basic sections of the window section.

```
GridLayout grid2 = new GridLayout(2,2);
setLayout(grid2);
```

# Running The Program

- The add calls now ignore “Center” etc, and place button1, button2 and pan in a 2 x 2 grid in the order the add methods were called.

# Summary

- `widget.setEnabled(boolean)` enable / disable
- `JOptionPane.showMessageDialog` shows a message
- `JOptionPane.showConfirmDialog` Yes / No options
- `JOptionPane.showInputDialog` can enter a string
- `JPanel` is a sub-window
- `BorderLayout` Center, North, South, East, West
- `FlowLayout` side by side.
- `GridLayout` in a grid
- `setLayout` changes the default layout manager.