Graphical User Interface Programming (Part 1)

Copyright © 2020 by
Robert M. Dondero, Ph.D.
Princeton University
Objectives

• You will learn about:
  – Graphical user interface (GUI) programming
  – Specifically...
  – For Java: The *AWT/Swing* GUI library
  – For Python: The *PyQt5* GUI library
Objectives

• More specifically…

• You will learn about:
  – “High-level” GUI programming
    • Programming with buttons, menus, text fields, …

• You will **not** learn about:
  – “Low-level” GUI programming
    • Drawing lines, circles, rectangles, etc.
    • See COS 126
Agenda

• Introduction
• Components/widgets
• Layout managers
Aside: Firewalls

• Wikipedia:
  – “In computing, a **firewall** is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a **trusted internal network** and **untrusted external network**, such as the Internet.”
Option 1: Run on coursela

User 1
Local Computer

X Window Server

User 2
Local Computer

X Window Server

coursela

App (User 1)

App (User 2)

Firewall

X Windows

X Windows
GUI Programming in COS 333

• Option 1: Run on courselab
  – Must install X Window System Server on local computer
  – See instructions in the A COS 333 Computing Environment document from 1st lecture
GUI Programming in COS 333

Option 2: Run on your computer

User 1
Local Computer
App

User 2
Local Computer
App
GUI Programming in COS 333

• Option 2: Run on your computer
  – Must install Java (with AWT/Swing) or Python (with PyQt) on your computer
  – Caveats:
    • Make sure you install proper version
      – Java SE 8 or higher
      – Python 3.6 or higher
    • Assignments must work on courselab
Java AWT/Swing Brief History

- **Abstract Window Toolkit (AWT)**
  - JDK 1.0
  - Delegates painting/behavior of each GUI component to the *native GUI toolkit*, e.g.:
    - Mac: Cocoa
    - MS Windows: Win64 API
    - Linux: Xlib
  - Library constrained to greatest common denominator approach
  - Components could behave (somewhat) differently on different computers
Java AWT/Swing Brief History

- **Swing**
  - J2SE 1.2
  - Built on top of AWT architecture
  - Relies on underlying native GUI toolkit to paint windows
  - Paints components itself onto blank windows
  - Library **not** constrained to greatest common denominator
  - Components **do** behave the same on any computer
  - Bonus: Application/user can choose look-and-feel
AWT/Swing Window Structure
AWT/Swing Pgm Structure

• See **HelloSwing.java**
  – Define class to implement **Runnable**
  – Define **run()** method within class
  – Instantiate **JLabel** object
  – Instantiate **JPanel** object
    • Add **JLabel** object to it
  – Instantiate **JFrame** object
    • Add **JPanel** object to it
AWT/Swing Pgm Structure

- **main() method:**
  - Instantiate **Runnable** object
  - Pass to `EventQueue.invokeLater()`

- `EventQueue.invokeLater()` method starts event loop in a new thread

- Suggestion: accept as a pattern

- More details in *Concurrent Programming* lecture
Python PyQt5 Brief History

• **Qt**
  – “Cute”
  – Haavard Nord and Eirik Chambe-Eng
  – Trollteck, Nokia, The Qt Company
  – Very successful cross-platform GUI library
  – Written in C++

• **PyQt5**
  – Riverbank Computing
  – A “Python binding” of Qt
  – 440 classes; > 6000 functions/methods
Python PyQt5 Brief History

• Note:
  – PyQt5 ≈ PySide2 ≈ Qt for Python
  – Write PyQt5 code
  – Read PyQt5, PySide2, or Qt for Python documentation
PyQt5 Window Structure

QMainWindow

QFrame (centralWidget)

QLabel

Hello, world
PyQt5 Program Structure

• See `hellogyqt.py`
  – Instantiate `QLabel` object
  – Instantiate `QFrame` object
    • Add `QLabel` object to it
  – Instantiate `QMainWindow` object
    • Add `QFrame` object to it
  – `app.exec_()` method call starts event handling loop

• Suggestion: accept as a pattern
Agenda

• Introduction
• **Components/widgets**
• Layout managers
Swing Components Example

- See `ComponentTest.java`
  - Note...
# Swing Components Example

<table>
<thead>
<tr>
<th>Conceptual Component</th>
<th>Relevant Swing Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>JFrame, JPanel</td>
</tr>
<tr>
<td>Button</td>
<td>JButton</td>
</tr>
<tr>
<td>Label</td>
<td>JLabel</td>
</tr>
<tr>
<td>Text field</td>
<td>JTextField</td>
</tr>
<tr>
<td>Text area</td>
<td>JTextArea</td>
</tr>
<tr>
<td>Scrolling text area</td>
<td>JTextArea, JScrollPane</td>
</tr>
<tr>
<td>Slider</td>
<td>JSlider</td>
</tr>
<tr>
<td>Check box</td>
<td>JCheckbox</td>
</tr>
<tr>
<td>Radio button</td>
<td>JRadioButton, ButtonGroup</td>
</tr>
</tbody>
</table>
## Swing Components Example

<table>
<thead>
<tr>
<th>Conceptual Component</th>
<th>Relevant Swing Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>List box</td>
<td>JList</td>
</tr>
<tr>
<td>Scrolling list box</td>
<td>JList</td>
</tr>
<tr>
<td></td>
<td>JScrollView</td>
</tr>
<tr>
<td>Menu</td>
<td>JMenuBar</td>
</tr>
<tr>
<td></td>
<td>JMenu</td>
</tr>
<tr>
<td></td>
<td>JMenuItem</td>
</tr>
</tbody>
</table>
Swing Components

Generalizing:

```
java.lang.Object
 java.awt.Component
  java.awt.Container
   java.awt.Window
    java.awt.Frame
     javax.swing.JFrame
  java.awt.Dialog
    javax.swing.JDialog
     javax.swing.JComponent
 (see next slide)
```
Swing Components

javax.swing.JComponent
    java.awt(AbstractButton
        javax.swing.JToggleButton
            javax.swing.JCheckBox
            javax.swing.JRadioButton
        javax.swing.JButton
    javax.swing.JMenuItem
        javax.swing.JMenu
    javax.swing.JInternalFrame
    javax.swing.JComboBox
    javax.swing.JLabel
    javax.swing.JLayeredPane
    javax.swing.JList
    javax.swing.JMenuBar
    javax.swing.JPanel
    javax.swing.JPopupMenu
    javax.swing.JProgressBar
Swing Components

javax.swing.JComponent (cont.)
  javax.swing.JRootPane
  javax.swing.JScrollBar
  javax.swing.JScrollPane
  javax.swing.JSlider
  javax.swing.JSplitPane
  javax.swing.JTabbedPane
  javax.swing.JTable
  javax.swing.JTextComponent
    javax.swing.JTextField
    javax.swing.JPasswordField
    javax.swing.JFormattedTextField
  javax.swing.JTextArea
    javax.swing.JTextEditorPane
    javax.swing.JTextPane
  javax.swing.JToolBar
  javax.swing.JTree
PyQt5 Widgets Example

- See `widgettest.py`
  - Note...
PyQt5 Widgets Example

<table>
<thead>
<tr>
<th>Conceptual Widget</th>
<th>Relevant PyQt Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>QMainWindow</td>
</tr>
<tr>
<td></td>
<td>QFrame</td>
</tr>
<tr>
<td>Button</td>
<td>QPushButton</td>
</tr>
<tr>
<td>Label</td>
<td>QLabel</td>
</tr>
<tr>
<td>Text field</td>
<td>QLineEdit</td>
</tr>
<tr>
<td>Text area</td>
<td>QTextEdit</td>
</tr>
<tr>
<td>Scrolling text area</td>
<td>QTextEdit</td>
</tr>
<tr>
<td>Slider</td>
<td>QSlider</td>
</tr>
<tr>
<td>Check box</td>
<td>QCheckBox</td>
</tr>
<tr>
<td>Radio button</td>
<td>QRadioButton</td>
</tr>
</tbody>
</table>
# PyQt5 Widgets Example

<table>
<thead>
<tr>
<th>Conceptual Widget</th>
<th>Relevant PyQt Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>List box</td>
<td>QListWidget</td>
</tr>
<tr>
<td>Scrolling list box</td>
<td>QListWidget</td>
</tr>
<tr>
<td>Menu</td>
<td>QMenuBar, QMenu</td>
</tr>
</tbody>
</table>
PyQt5 Widgets

Some PyQt5 widget classes…

QObject & QPaintDevice
QWidget
    QMainWindow
    QAbstractButton
        QCheckBox
        QPushButton
        QRadioButton
    QAbstractSlider
        QSlider
    QFrame
        QAbstractScrollArea
            QTextEdit
        QAbstractItemView
            QListView
                QListWidget
    QLabel
    QLineEdit
    QMenu
    QMenuBar

And many many more
Agenda

• Introduction
• Components/widgets
• Layout managers
AWT Layout Manager Examples

• See `LayoutFlow.java`
  – FlowLayout
  – Flow is maintained as window resizes
AWT Layout Manager Examples

• See `LayoutBorder.java`
  – `BorderLayout`
  – **North** and **south** border: sizes are fixed vertically, expand horizontally
  – **East** and **west** border: sizes are fixed horizontally, expand vertically
  – **Center** expands both horizontally and vertically
AWT Layout Manager Examples

• See *LayoutGrid.java*
  – GridLayout
  – Components arranged in 2-D grid
AWT Layout Managers

```java
java.lang.Object
    java.awt.FlowLayout
    java.awt.BorderLayout
    java.awt.GridLayout
    java.awt.GridBagLayout
    java.awt.BoxLayout
    java.awt.CardLayout
    java.awt GroupLayout
    java.awt.OverlayLayout
    java.awt.ScrollBarsLayout
    java.awt.SpringLayout
    java.awt.ViewportLayout
```
PyQt5 Layout Mgr Examples

• See `layoutvbox.py`
  – `QVBoxLayout` class
  – `addWidget()` method

• See `layout hbox.py`
  – `QHBoxLayout` class
  – `addWidget()` method
PyQt5 Layout Mgr Examples

• See `layoutgrid.py`
  – `QGridLayout` class
  – `addWidget(row, col)` method
  – `setSpacing(i)` method

• See `layoutgrid2.py`
  – `addWidget(row, col, rowSpan, colSpan)` method
  – `setRowStretch(row, stretch)` method
  – `setColumnStretch(col, stretch)` method
PyQt5 Layout Managers

- **QGridLayout layout manager**
  - Same functionality as AWT GridLayout
    - Components arranged in 2-D grid
  - Same functionality as AWT BorderLayout
    - Can specify row/col spans
    - Can specify row/col stretches
PyQt5 Layout Managers

QObject & QLayoutItem
QLayout
QBoxLayout
QFormLayout
QGridLayout
QHBoxLayout
QStackedLayout
QVBoxLayout
Summary

• We have covered:
  – Graphical user interface (GUI) programming
  – Specifically...
  – For Java: The **AWT/Swing** GUI library
  – For Python: The **PyQt5** GUI library
Summary

• For each:
  – Introduction
  – Components/widgets
  – Layout managers