COMP241
Software Engineering Development
Lecture 16: Swing Widgets & Tools 2 (CVS)

Mark Hall

- Swing widgets
  - Radio buttons
  - Check boxes
  - Combo boxes
  - Lists
  - Menus
  - Text Areas
- CVS

THE UNIVERSITY OF WAIKATO *DEPARTMENT OF COMPUTER SCIENCE*
TARI RO ROHIKO

Radio Buttons

- One way to present a finite set of choices to the user—radio buttons present a *mutually exclusive* set of choices

```
JRadioButton smallB = new JRadioButton("small");
JRadioButton largeB = new JRadioButton("large");
```

- To create a set of radio buttons add them to a `ButtonGroup`
  - Ensures that only one radio button can be selected at any one time

```
ButtonGroup group = new ButtonGroup();
group.add(smallB); group.add(largeB);
```

- Use `setSelected/isSelected` to set, or see if a button is selected

Check Boxes

- Another way to present a finite set of choices—check boxes are *not mutually exclusive*

```
JCheckBox iCheckBox = new JCheckBox("Italic");
```

- Do not place check boxes inside a button group
- Like radio buttons, use `setSelected/isSelected`
- Both `JCheckBox` and `JRadioButton` produce `ActionEvents` and `ItemEvents`

Combo Boxes

- Combo Boxes (drop down box)
  - Can present a large number of choices
  - Combination of a list and a text field

```
JComboBox fnCombo = new JComboBox();
fmCombo.addItem("Serif");
fnCombo.addItem("SansSerif");
```

  - You get the item that the user has selected by calling the `getSelectedItem` method
    - Returns `Object` because `JComboBox` can store arbitrary objects!
    - Uses the `toString` method of stored objects to display text in the drop down box

List

- Constructor

```
String [] listEntries = {"alpha", "beta", "gamma"};
JList list = new JList(listEntries);
```

- Usage
  - Vertical scrollbar

```
JScrollPane scroller = new JScrollPane(list);
scroller.setVerticalScrollBarPolicy(ScrollPaneConstraints.
  VERTICAL_SCROLLBAR_ALWAYS);
```

  - Set number of lines before scrolling

```
list.setVisibleScrollCount(4);
```

  - Restrict user to selecting only **one** thing at a time

```
list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

  - Register for *list selection* events

```
list.addListSelectionListener(this);
```


Menus

- Are easy to create in Java
- First construct a *menu bar* to hold menus and attach it to a `JFrame:`

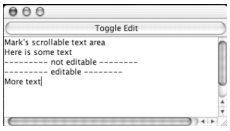
```
JMenuBar mBar = new JMenuBar();
JFrame myApp = new JFrame();
myApp.setJMenuBar(mBar);
```

Menus

- Menus
  - Then you add menus to the menu bar:

```
JMenu fileMenu = new JMenu("file");
mBar.add(fileMenu);
```

  - A menu is a collection of *menu items* and more menus (submenus)
  - `JMenuItem fileNewMenuItem = new JMenuItem("New");`
  - `fileMenu.add(fileNewMenuItem);`
- A menu item has no further submenus
- When the user selects a menu item, the menu item sends an action event
  - So, attach an action listener to each menu item



Text Areas

- `JTextField` can hold a single line of text
- To display multiple lines of text we can use a `JTextArea` object
  - Can specify the number of rows and columns in the constructor
  - Use `setText` to set the text and `append` to add text to the end of the text area
  - Can use `JTextArea` for display only by calling `setEditable(false)`


```
public class TextAreaExample extends JPanel {
  private JTextArea mMyText = new JTextArea(10, 30);
  private JButton mEditableToggle =
    new JButton("Toggle Edit");
  public TextAreaExample() {
    super();
    setLayout(new BorderLayout());
    add(mEditableToggle, BorderLayout.NORTH);
    JScrollPane scrollPane = new JScrollPane(mMyText);
    add(scrollPane, BorderLayout.CENTER);
    mMyText.setText("Mark's scrollable text area");

    mEditableToggle.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        boolean editable = !mMyText.isEditable();
        mMyText.setEditable(editable);
        String message = ((editable == true)
                ? " editable "
                : " not editable ");
       mMyText.append("\n---------"+message+"--------");
      }
    });
  }
}
```

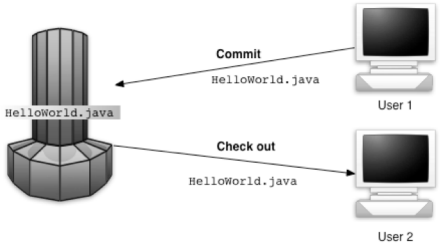
Further readings on Swing

- Horstmann Chapters 4 and 5
- Sun's tutorial "trail" on Swing UI programming: http://java.sun.com/docs/books/tutorial/uiswing

CVS, what is it?

- CVS—Concurrent Versions System
- Collaboration
  - Supports editing of source file collections by several developers
  - Merges changes made by different developers at the same time
- Record keeping
  - Keeps history of changes made to your source files

The CVS Repository



Set Up

- Put the following into your .profile

```
export CVSROOT =
:ext:<user>@cvs.scms.waikato.ac.nz:/usr/local/global-cvs/comp241a
export CVS_RSH=ssh
```

- This may only take effect after logging in
- Then you can check out your group's work directory

  `cvs co` <*username*>

  – I have created subdirectories using your usernames

Usage

- Add files to repository

  `cvs add` <*file1*> <*file2*> …
- Commit your changes

  `cvs commit` <*file1*> <*file2*> …

  `cvs commit -m` *"<log entry>"* <*file1*> <*file2*> …
- Retrieve files from repository

  `cvs update` <*file1*> <*file2*> …
- View differences

  `cvs diff` <*file1*> <*file2*> …
- Setting a tag

  `cvs tag` <*name*>

Documentation

- CVS Book
  – http://cvsbook.red-bean.com
- CVS Tutorial
  – http://www.cvshome.org/docs/blandy.html
- CVS Manual
  – http://www.cs.waikato.ac.nz/~mhall/241/resources/cedergvist-1.11.14.pdf