

## Computer/Human Interaction

### Lecture 16

#### Overview:

- More input elements, listeners
- Message dialogs
- Menus
- File I/O

#### References:

- JSWI, JTUT, JAPI

## Purple Pizza Parlor, v2

- Download from web or copy files  
`/home/hwang/cs350/lecture16/*.*`
- All Linux boxen have updated Java compiler and runtimes, so don't need to change path
- To use Eclipse: create directory for files of a project. Import directory as project.
- Writes to file "`output/orders.txt`", need to create `output` subdirectory first.

## Combo Boxes

- JComboBox – construct with array of string items
- Uses ActionListener interface, dynamically cast the source of the event object into a JComboBox and get the selected item

## List Boxes

- JList – construct with array of string items
- Uses ListSelectionListener interface
  - Defined in `javax.swing.event.*`
  - Handler is `valueChanged`
- `MULTIPLE_INTERVAL_SELECTION` mode is default, also `SINGLE_SELECTION` and `SINGLE_INTERVAL_SELECTION` modes
- `getSelectedIndices` returns int array of indices of selected items

## Message Dialogs

- Simple OK message dialogs can be created using `JOptionPane.showMessageDialog`
- Arguments are the top-level frame, dialog message, window title, and message type (determines the icon); can also add icon image argument
  - Defined message types are `INFORMATION_MESSAGE`, `WARNING_MESSAGE`, `ERROR_MESSAGE`, `QUESTION_MESSAGE`, `PLAIN_MESSAGE` (no icon)

## MenuBars and Menus

- JMenuBar is added specially to top-level frame
- Default placement is platform-dependent
- Add JMenuItem objects to menubar
- Add JMenuItem, JMenuItem objects to menus
- Uses ActionListener interface
  - PPP Action menu uses same listener as the buttons, so define `ButtonListener`, a named inner class that both can use

## Miscellaneous

- Size and toppings are not in same panel; their panels have labels
- Added keyboard mnemonics to buttons
  - `submit.setMnemonic (KeyEvent.VK_S);`
  - Underlines the letter in the label
- Menubar creation is a call to application object method, so the application object is created first, then added to outer frame.

## File I/O

- File streams defined in `java.io.*`
- General paradigm is to wrap a buffered stream around an unbuffered stream around a file.
- `PrintWriter` implements `print` and `println`

```
PrintWriter outputStream = null;
try {
    outputStream = new PrintWriter(
        new FileWriter ("output/orders.txt"));
    outputStream.println ("Text with newline");
}
catch (IOException ex) { // error code
} finally { // clean up code }
```

## File I/O 2

- `BufferedReader` implements `readLine` that returns a line as a `String`, or null at end of file

```
BufferedReader inStream = null;
try {
    inStream = new BufferedReader(
        new FileReader ("input.txt"));
    String line;
    while ((line = inStream.readLine())
        != null)
    { // do something with line }
} catch ...
```

## Exercise

- Try the following
  - Add a phone field, add error checking to make sure it's filled in, add code to write the data to the output file
  - Separate the toppings into meats and vegetables. Create a list box for each. Create a named inner class that implements the `ListSelectionListener` interface so they can share the same listener.