Homework Hint!

**UI Class**
( imaginary)
*All user interaction goes here—and nothing else!*

**Other Classes**
(real)
*No user interaction here—all game and player state here*

Only make calls this way!
GUIDemo Example

- Shows how to construct a basic UI
- Illustrates an interesting UI responsiveness issue
The GUI Threading Architecture

- **main() thread**
  - Create window
  - Set up callbacks
  - Show window (thread ends)

- **GUI Thread**
  - Loop forever:
    - Get system event
    - Invoke callback

  - Callback code:
    - Compute fibonacci
    - (UI is unresponsive)
    - Show result
GUIDemo Example

- A fix: SwingWorker
The GUI Threading Architecture

**main() thread**
- Create window
- Set up callbacks
- Show window (thread ends)

**GUI thread**
- Loop forever:
  - Get system event
  - Invoke callback

**Callback code:**
- create SwingWorker
- start it executing

**Worker thread**
- Worker thread execution:
  - invoke doInBackground()
  - compute fibonacci
  - store result in SwingWorker
  - signal to UI that we are done

**SwingWorker**
- result : Long
The GUI Threading Architecture

The diagram illustrates the GUI Threading Architecture with two main threads:

1. **main() thread**
   - Create window
   - Set up callbacks
   - Show window (thread ends)

2. **GUI thread**
   - Loop forever:
     - Get system event
     - Invoke callback

3. **Worker thread**
   - Worker thread execution:
     - invoke doInBackground()
     - compute fibonacci
     - store result in SwingWorker
     - signal to UI that we are done

- **Invoke SwingWorker.done()**
- **get() result from SwingWorker**
- **show result in the UI**

**SwingWorker**
- result : Long
Organizational Tips

• Declare references to components you’ll be manipulating as instance variables

• Put the code that performs the actions in private “helper” methods. (Keeps things neat)
GUI design issues

• Interfaces vs. inheritance
  – Inherit from JPanel with custom drawing functionality
  – Implement the ActionListener interface, register with button
  – Why this difference?

• Models and views
GUI design issues

• Interfaces vs. inheritance
  – Inherit from JPanel with custom drawing functionality
    • Subclass “is a” special kind of Panel
    • The subclass interacts closely with the JPanel – e.g. the subclass calls back with super()
    • The way you draw the subclass doesn’t change as the program executes
  – Implement the ActionListener interface, register with button
    • The action to perform isn’t really a special kind of button; it’s just a way of reacting to the button. So it makes sense to be a separate object.
    • The ActionListener is decoupled from the button. Once the listener is invoked, it doesn’t call anything on the Button anymore.
    • We may want to change the action performed on a button press—so once again it makes sense for it to be a separate object

• Models and views
Model-View-Controller (MVC)

Manage inputs from user: mouse, keyboard, menu, etc.

Manage display of information on the screen

Manage data related to the application domain

Model-View-Controller (MVC)

Passive model

Active model

Example: RabbitWorld GUI

- ...hw2.lib.ui.WorldImpl
  - The Model class
  - Model is passive: does not have a reference to the view
- ...hw2.lib.ui.WorldUI
  - The Controller class
  - Listener callbacks in constructor react to events
    - Delegating to the view (is this design ideal?)
- ...hw2.lib.ui.WorldPanel
  - The View class
  - Gets data from Model to find out where to draw rabbits, foxes, etc.
  - Implements stepping (in step())
    - Invokes model to update world
    - Invokes repaint() on self to update UI
Find That Pattern!

- What pattern is BorderLayout a part of?

- What pattern is JPanel a part of?

- What pattern are the ActionListeners part of?

- There are classes representing the AI’s decision to Eat, Breed, or Move. What pattern are these representing?

- Look at the documentation for JComponent.paint(). What pattern is used?
For More Information

• Oracle’s Swing tutorials
  – http://download.oracle.com/javase/tutorial/uiswing/
• Introduction to Programming Using Java, Ch. 6
Questions?