Tell, Don't Ask

- **Tell objects what you want them to do, do not ask questions about state, make a decision, then tell them what to do** *(Pragmatic Programmers, LLC)*
  - **Think declaratively, not procedurally**
  - **Don't ask for a map, then walk through the map**
  - **Instead of iteration, apply to all**
    - **Breaks when we don't want to apply to all**

- **Rules are made to be broken**
  - **Reduce coupling, better code**

Law of Demeter

- Don't talk to objects, don't call methods. The more you talk, the more you rely on something that will break later
  - Call your own methods
  - Call methods of parameter objects
  - Call methods if you create the object

- Do *NOT* call methods on objects returned by calls

```
List all = obj.getList();
all.addSpecial(key,getValue());
obj.addToList(key,getValue());   // ok here
```

Toward a GUI-programming model

- We want to adhere to language-independent ideals
  - Concepts move from GUIs in Java to …
  - javax.swing and java.awt offer thousands of choices
    - Too many to have to understand/find comfort in, but …
- But, write reasonable, robust , GUI applications
  - Actually write code, not simply adhere to lofty ideals
  - Show me the code!

One GUI Conceptual Framework

- Create a JPanel for the GUI contentPane
  - Provide a BorderLayout, organize hierarchically
  - Ok to use GridLayout, FlowLayout, … nested

- Create Buttons, Menu-items, and other widgets
  - Bind each event-generator to a listener
  - Do not dispatch within a listener on event source
    - No "if event-generator is button A do this"

- Use anonymous inner classes, or named inner classes
  - Process events, created and attached close-to-source
  - Make a button, make a button-listener

Click on a button, display the click

```
ActionListener textDisplayer = new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
        showText(e.getActionCommand());
    }
};
```

- What does an ActionListener do?
  - Listens for an event, e.g., from Button, Menu, …
  - Processes the command/event

- How do anonymous classes work?
  - Note: ActionListener is an *interface*, but object created!
  - See what Eclipse refactoring will do with this

Making a Move: View and Controller

```
ActionListener moveMaker = new ActionListener(){
  public void actionPerformed(ActionEvent e)
  {
    int val = Integer.parseInt(e.getActionCommand());
    myControl.makeMove(new GameMove(val));
  }
};
```

- We know this will be bound to a specific type of button
  - Not generic, completely application specific
  - Turns swing/GUI event into application event: Move
- Controllers should be programmed abstractly
  - Don't base code on a GUI toolkit, separate concerns

Alan Kay, winner of 2004 Turing Award

- Alto, Smalltalk, Squeak, Object-Oriented Programming

"Don't worry about what anybody else is going to do… The best way to predict the future is to invent it. Really smart people with reasonable funding can do just about anything that doesn't violate too many of Newton's Laws!"

Alan Kay on Education and OO

- "By the time I got to school, I had already read a couple hundred books. I knew in the first grade that they were lying to me because I had already been exposed to other points of view. School is basically about one point of view -- the one the teacher has or the textbooks have. They don't like the idea of having different points of view, so it was a battle. Of course I would pipe up with my five-year-old voice."

- Java and C++ make you think that the new ideas are like the old ones. Java is the most distressing thing to hit computing since MS-DOS.

- I invented the term "Object-Oriented", and I can tell you I did not have C++ in mind.