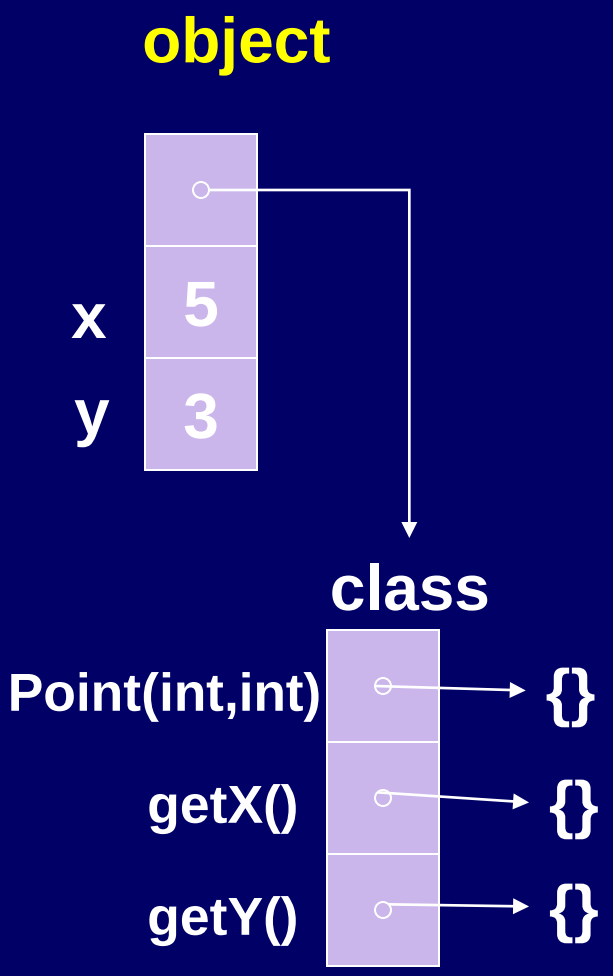
CSC324: Principles of Programming Languages
Lecture 8
Java's Reflection

Wael Aboulsaadat

Internal Representation of .class (Java's bytecode)

```
public class Point {
  public Point(int x, int y) {
    x = 5; Y = 10;
  }
  public int getX() {
        return x;
  }
  public int getY() {
        return y;
    }
  protected int x, y;
}
```



The class Class

- Instances of the class Class store information about classes
  - Class name
  - Inheritance
  - Interfaces implemented
  - Methods, members, etc.

- Can look up instances:
  - By name
  - From an object

Discovering the type of a Class

Discovering the Methods of a class

Calling a Method using Reflection

1. Look up a method based on its signature: the name and list of parameter types

2. Specify signature as a comma-separated list of Class objects
   - Specifies the types of arguments
   - Special values for types like int and boolean

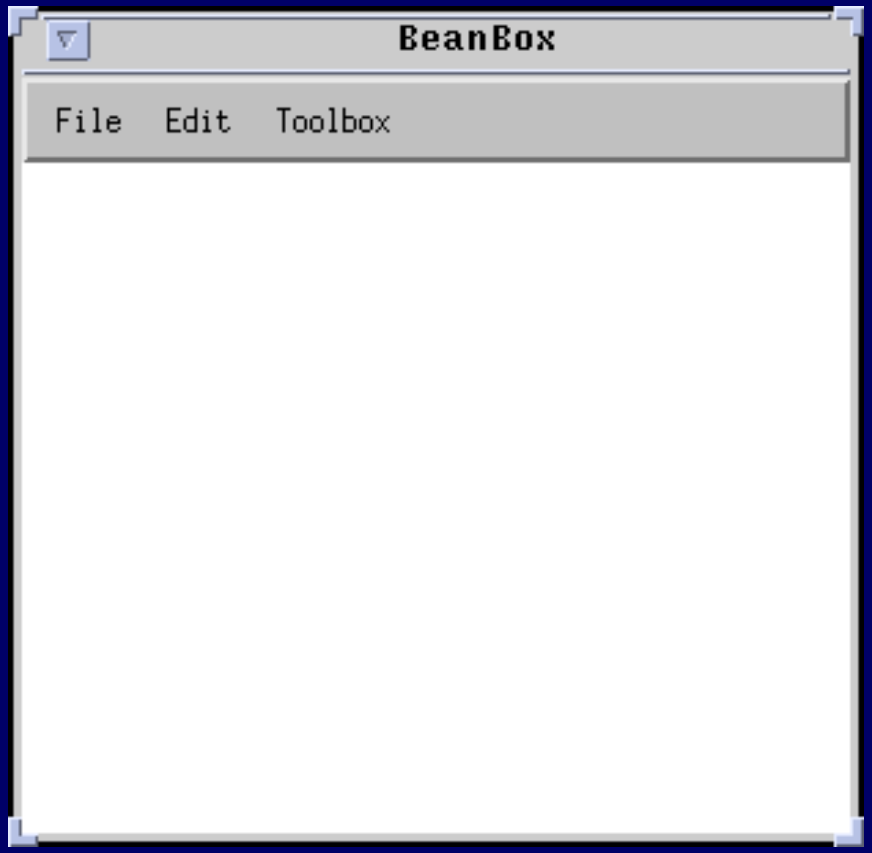
3. Call the method, passing in parameters and capturing return value

```
Class mysClass = Class.forName("Mystery");
Object o = mysClass.newInstance();

Method m = mysClass.getMethod("euclidean",
                              Double.TYPE,
                              Double.TYPE);

double result = (Double) m.invoke(o, new Double(5.0), 12.0);

Method m2 = mysClass.getMethod("play",
                               Class.forName("java.lang.String"),
                               String.class);

m2.invoke(o, "Che", "Karl");
```

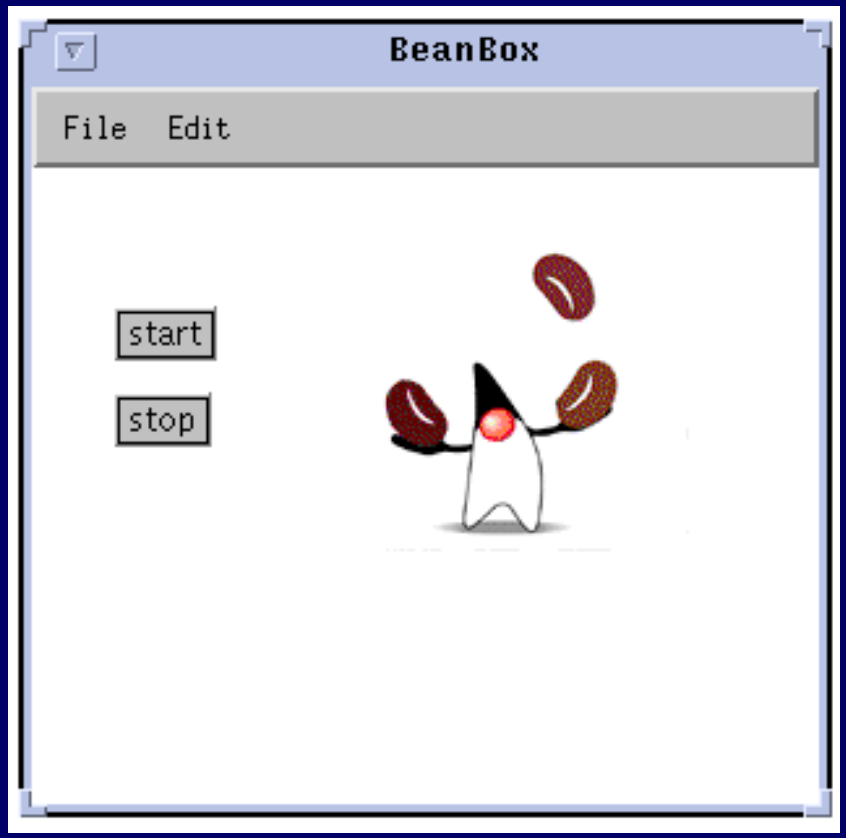
Advantages of Reflection?

- Supports component based development
- Decreases control coupling by eliminating direct references to class names.
- Supports software automation

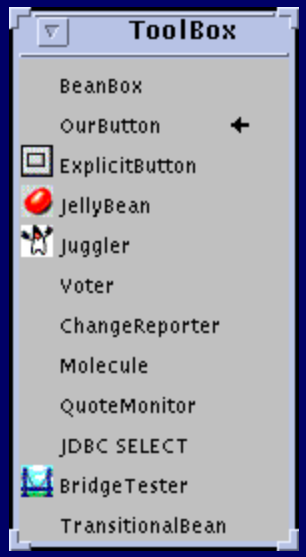
Bean Box

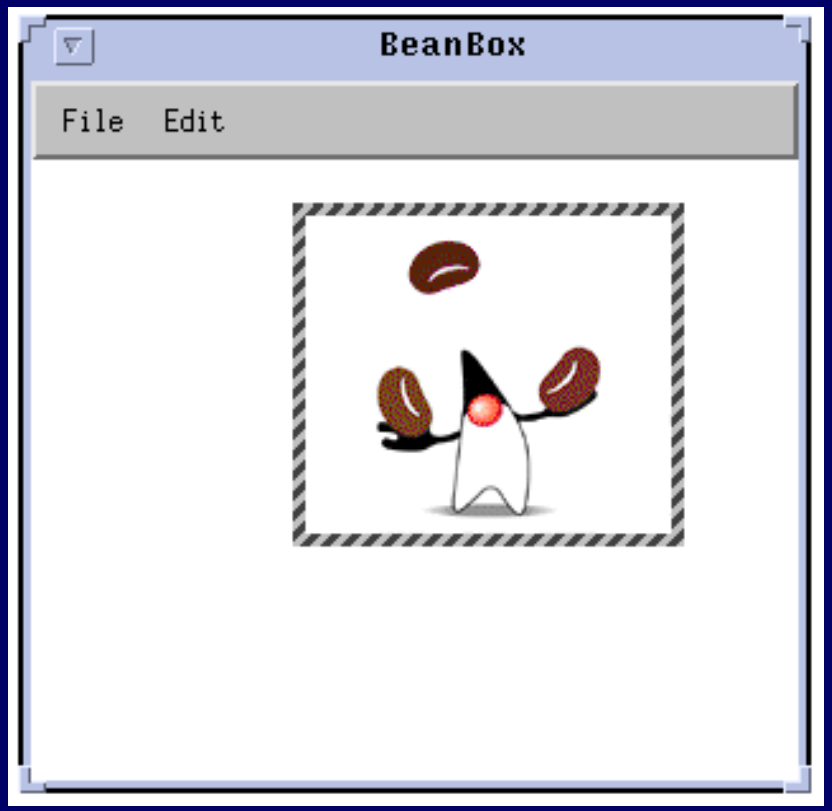


Bean Box

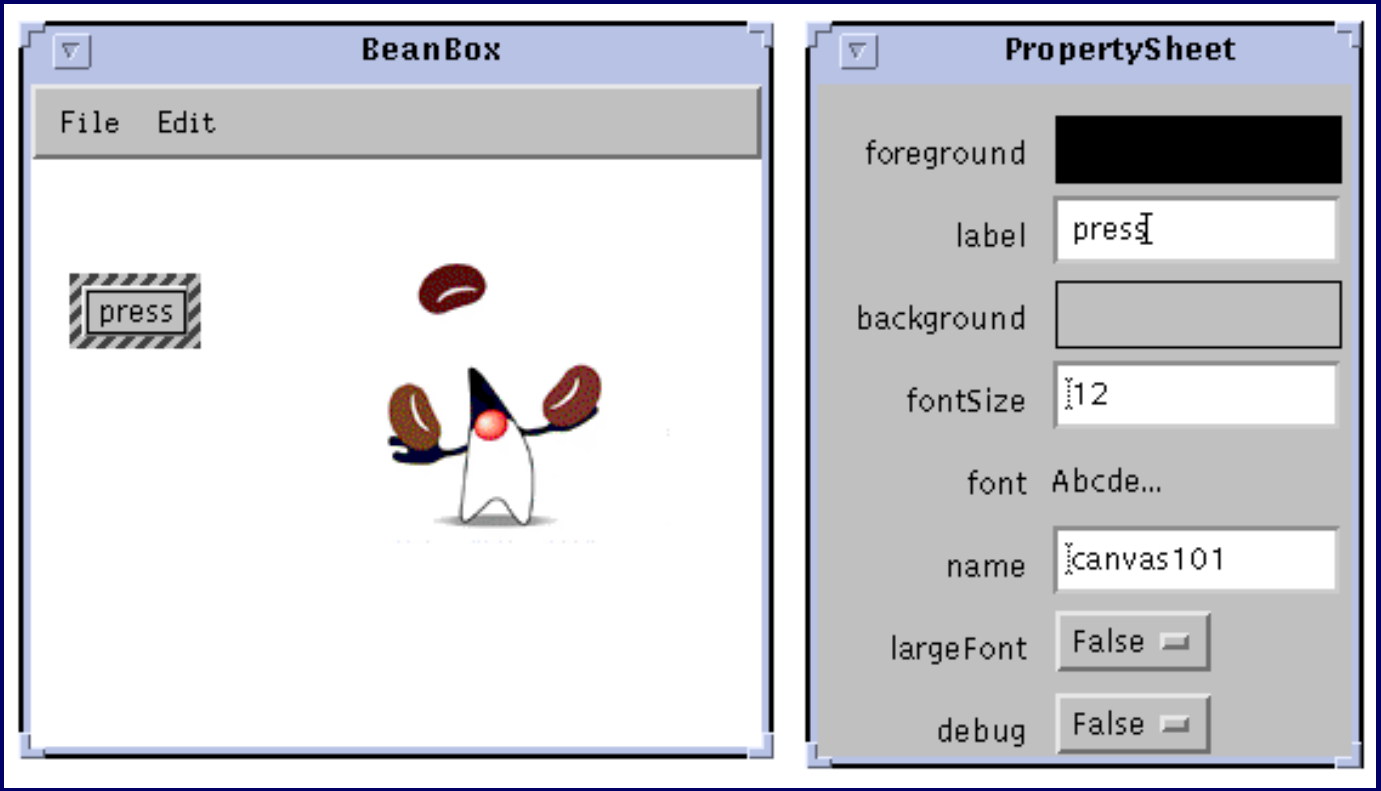


Bean Box

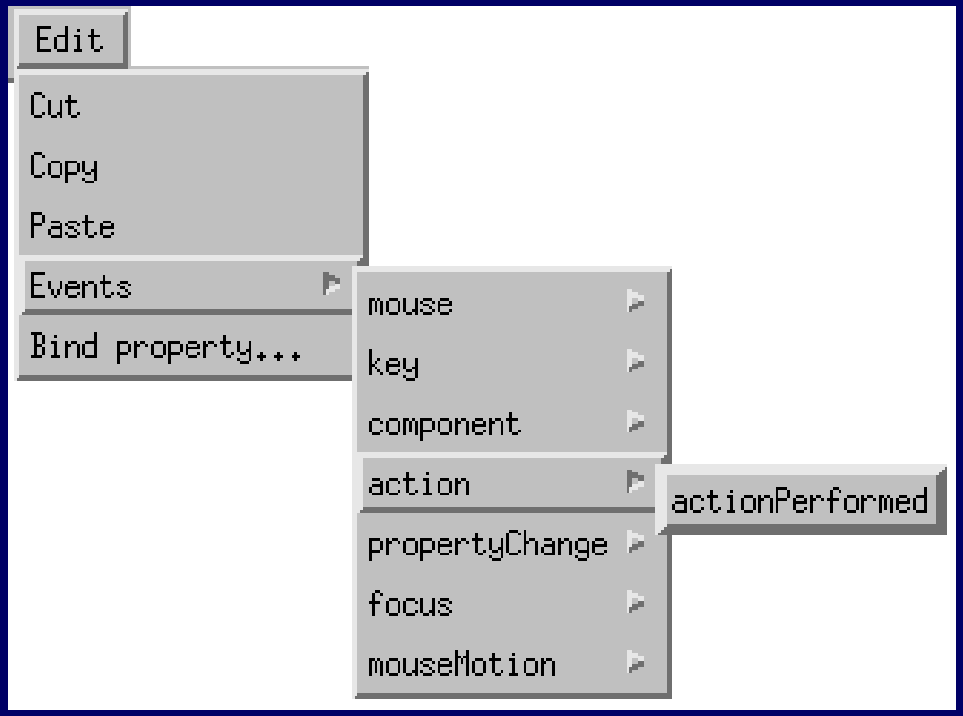

ToolBox
BeanBox
OurButton
ExplicitButton
JellyBean
Juggler
Voter
ChangeReporter
Molecule
QuoteMonitor
JDBC SELECT
BridgeTester
TransitionalBean


BeanBox
File
Edit

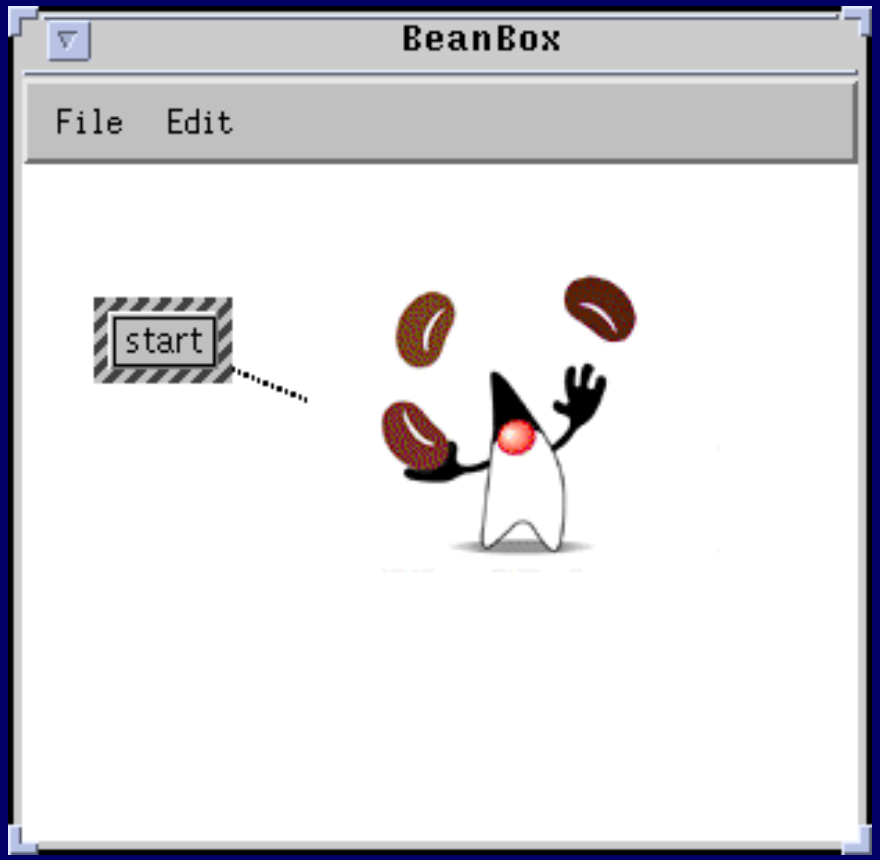
Bean Box


BeanBox
File Edit
press
PropertySheet
foreground
label
press
background
fontSize
12
font Abcde...
name
canvas101
largeFont
False
debug
False

Bean Box


Edit
Cut
Copy
Paste
Events
Bind property...
mouse
key
component
action
propertyChange
focus
mouseMotion
actionPerformed

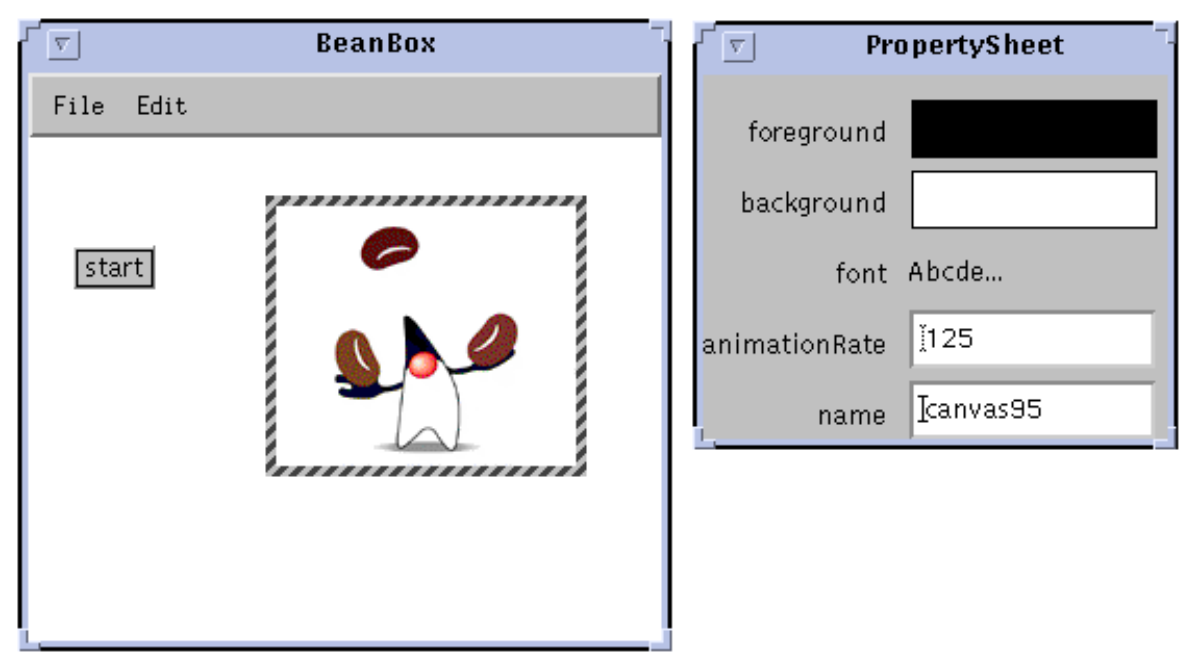
Bean Box



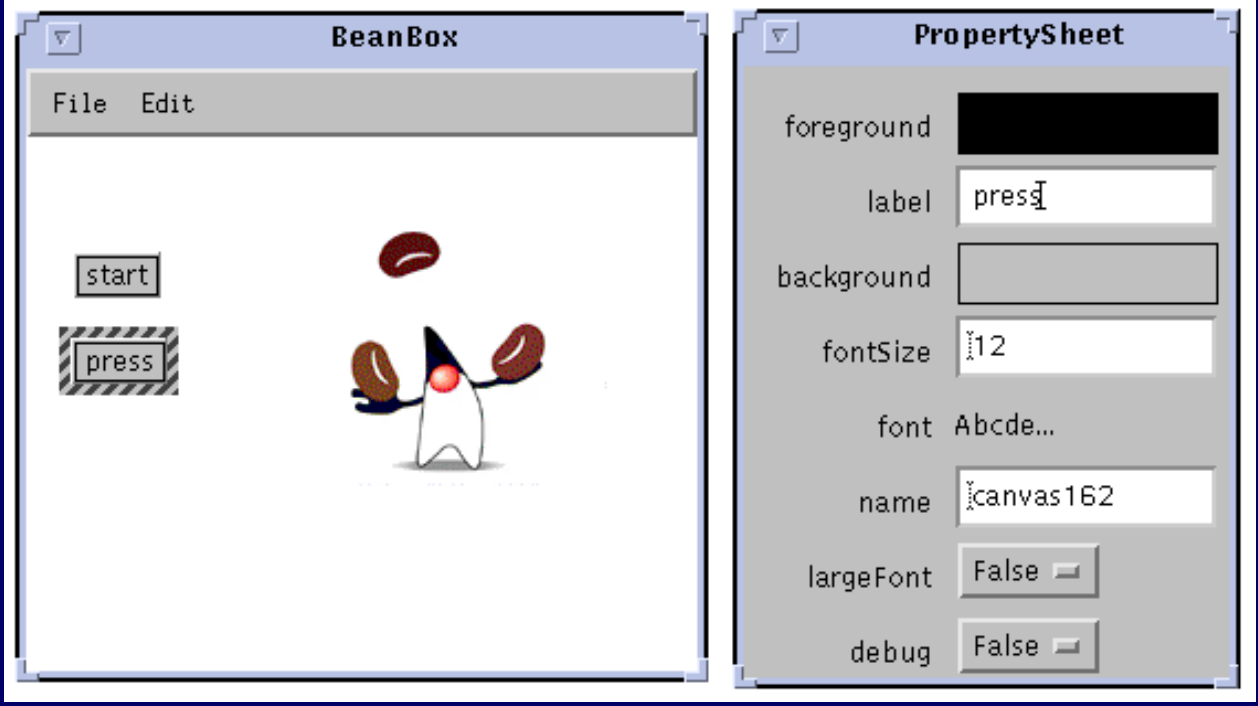
Bean Box


BeanBox
File Edit
start
EventTargetDialog
Please chose a target method:
stop
start
Cancel
OK

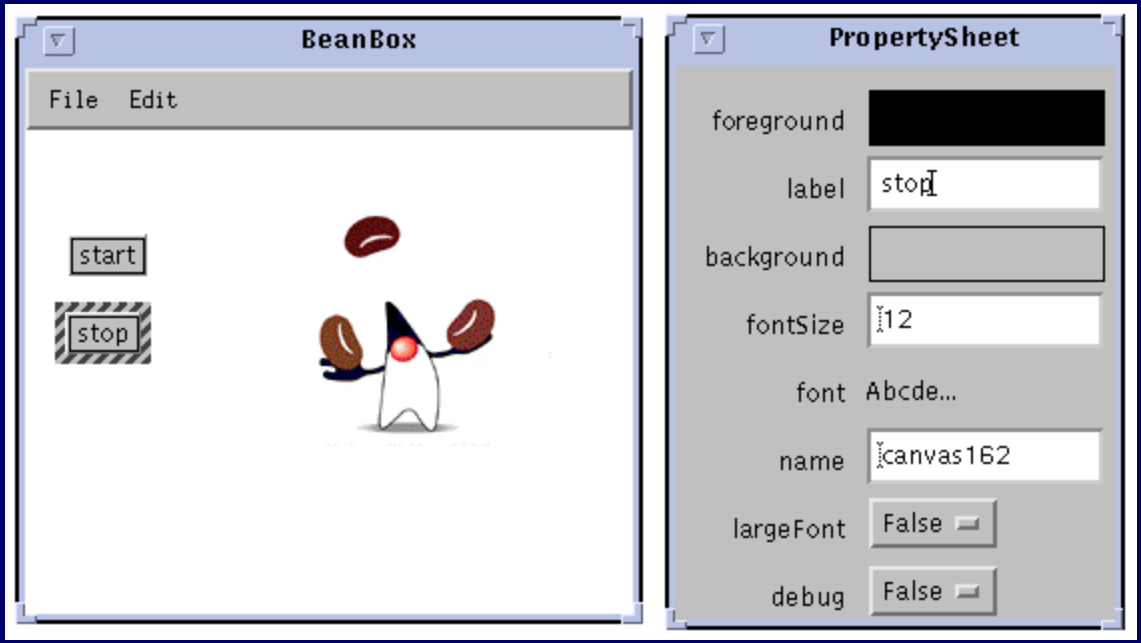
Bean Box


BeanBox
File Edit
start
PropertySheet
foreground
background
font Abcde...
animationRate 125
name canvas95

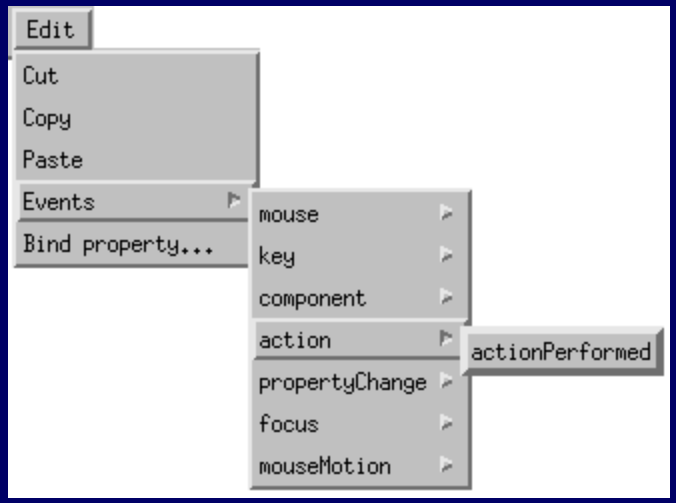
Bean Box

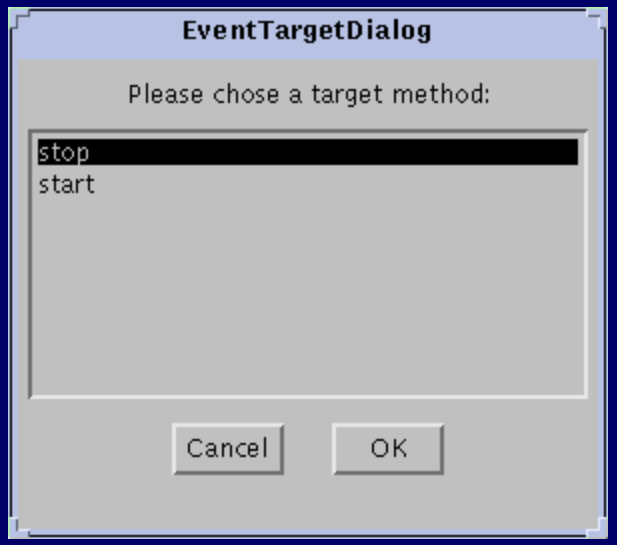

BeanBox
File Edit
start
press
PropertySheet
foreground
label
press
background
fontSize
12
font Abcde...
name
canvas162
largeFont
False
debug
False

Bean Box

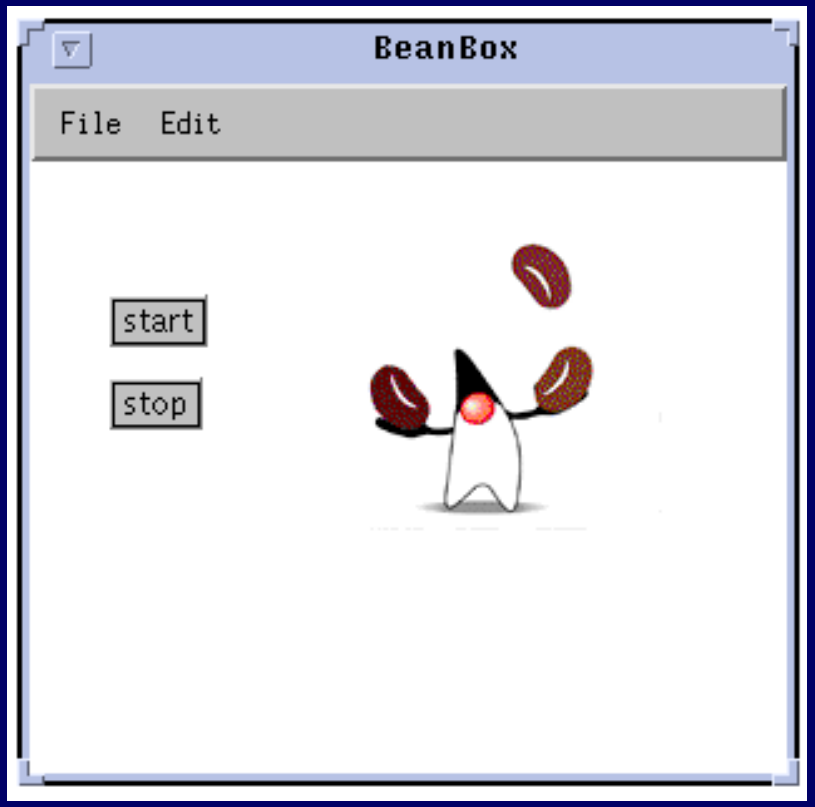

BeanBox
File Edit
start
stop
PropertySheet
foreground
label
stop
background
fontSize
12
font
Abcde...
name
canvas162
largeFont
False
debug
False

Bean Box


Edit
Cut
Copy
Paste
Events
Bind property...
mouse
key
component
action
propertyChange
focus
mouseMotion
actionPerformed


EventTargetDialog
Please chose a target method:
stop
start
Cancel
OK

Bean Box



What's happening in the background

- Reflection is used to expose the internals of an object
- Methods are written in a standard agreed-upon style
  - setTarget(int newValue);
  - int getTarget( );
- Events are known from the interface(s) implemented by the class

Advantages of Reflection?

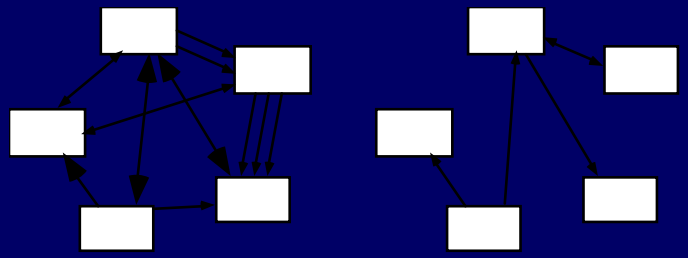
- Supports component based development

➔ Decreases Coupling by eliminating direct references to class names.

- Supports software automation

Goal: Reduce coupling where possible

- *Coupling* occurs when there are *interdependencies* between one module and another
  - When interdependencies exist, changes in one place will require changes somewhere else.
  - A network of interdependencies makes it hard to see at a glance how some component works.
  - Type of coupling:
    - Content, Common, Control, Stamp, Data, Routine Call, Type use, Inclusion/Import, External



Control coupling

- Occurs when one procedure calls another using *a 'flag' or 'command'* that explicitly controls what the second procedure does
  - To make a change you have to change both the calling and called method
  - The use of polymorphic operations is normally the best way to avoid control coupling
  - One way to reduce the control coupling could be to have a *look-up table*
    - commands are then mapped to a method that should be called when that command is issued

Example of control coupling

```
public routineX(String command)
{
   if (command.equals("drawCircle")
   {
      drawCircle();
   }
   else
   {
      drawRectangle();
   }
}
```

Advantages of Reflection?

- Supports component based development
- Decreases Coupling by eliminating direct references to class names.

➔ Supports software automation

What is Software Automation

- If a program consists of components….
- Then we would like to configure those modules anyway we like and be able to change that configuration of components later if we decide to do so…!