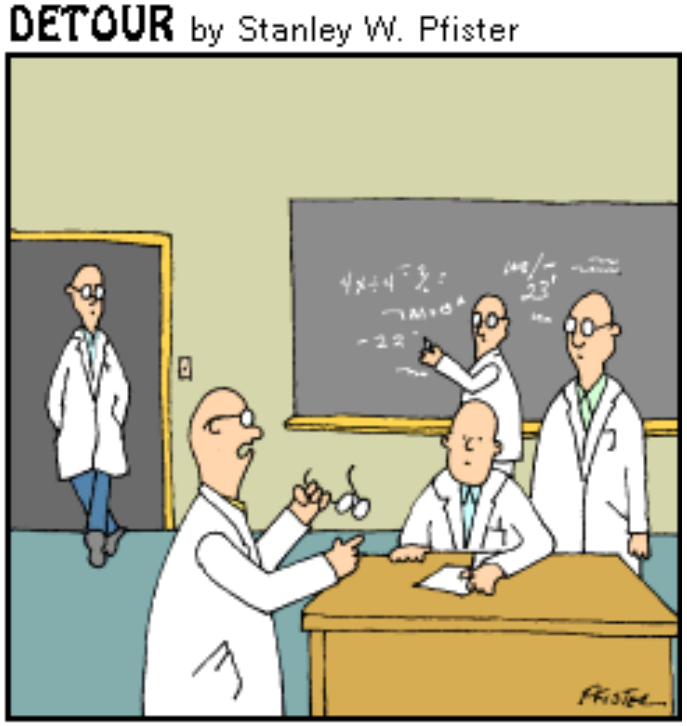
ORF 201

COMPUTER METHODS FOR PROBLEM SOLVING

Lecture 19

Static Vs. Non-static, Other Odds and Ends



"Hey! Why you're no scientist at all! These are nothing but window glass!"

Static Variables

```
class SomeStuff
{
    static int s;
    double value;
}
```

Static variable **`s`** belongs to the class **`SomeStuff`**. There is always exactly one "copy of this variable."

Variable **`value`** belongs to *instances* of class **`SomeStuff`**.

- Zero or more instances of this variable can exist at any time.
- They are created by calls to **`new SomeStuff()`**.

Terminology:

- Static variables are called *class variables*.
- Other variables are called *instance variables*.

A Demo Program

```
public class StaticDemo
{
    public static void main(String[] args)
    {
        SomeStuff x, y;

        x = new SomeStuff();
        y = new SomeStuff();

        x.s = 1;
        x.value = 3.14159;

        y.s = 2;
        y.value = 2.71828;

        System.out.println("x.s     = " + x.s);
        System.out.println("x.value = " + x.value);

        System.out.println("y.s     = " + y.s);
        System.out.println("y.value = " + y.value);

        SomeStuff.s = 3;

        System.out.println("x.s     = " + x.s);
    }
}
```

x.s and **y.s** are the same variable.

Referring to class variable **s** using **x.s** is legal but misleading.

It is better to use **SomeStuff.s**.

Writing **SomeStuff.value** is illegal - i.e. it produces a compile-time error.

What's the output?

```
x.s     = ____
x.value = ____
y.s     = ____
y.value = ____
x.s     = ____
```

Working with Checkboxes

- Create a column of checkboxes in a gridpanel.
- Each checkbox should know which row it is in.
- Class checkbox should remember the two most recently checked boxes.

```
public class CheckBoxDemo extends Applet
{
    private static final int n = 5;

    Label[] label;
    myCheckbox[] cb;

    public static void main(String[] args)
    {
        new AppletFrame(new CheckBoxDemo(), 300, 300);
    }

    public void init()
    { /* later */
    }

    public boolean handleEvent(Event evt)
    {
        if (evt.id == Event.WINDOW_DESTROY) System.exit(0);
        return super.handleEvent(evt);
    }

    public void setLabels(int row1, int row2)
    { /* later */
    }
}
```

Methods init() and setLabels()

```
public void init()
{
    int j;

    GridPanel gp = new GridPanel();

    cb = new myCheckbox[n+1];
    for (j=0; j<n; j++) { cb[j] = new myCheckbox(this, j); }
    for (j=0; j<n; j++( { gp.add(cb[j], 1, j); }

    label = new Label[n];
    for (j=0; j<n; j++) { label[j] = new Label("          ");
    for (j=0; j<n; j++) { gp.add(label[j], 2, j); }

    this.add(gp);
}

public void setLabels(int row1, int row2)
{
    for (int j=0; j<n; j++) {label[j].setText(""); }
    if (row1 >= 0) {label[row1].setText("previous"); }
    if (row2 >= 0) { label[row2].setText("current"); }
}
```

Class myCheckbox

```
class myCheckbox extends Checkbox
{
     int row;
     CheckBoxDemo cbd;

     static int curr=-1, prev=-1;

     public myCheckbox(CheckBoxDemo c, int r)
     {
          cbd = c;
          row = r;
     }

     public boolean action(Event evt, Object arg)
     {
          if (getState() == true) {
               if (prev>=0) cbd.cb[prev].setState(false);
               prev = curr;
               curr = row;
               cbd.setLabels(prev, curr);
          }
          return true;
     }
}
```

row must be an instance variable (i.e., not static).

curr and **prev** must be class variables (i.e., static).

The constructor records the calling class and the row.

getState() returns **true** if the checkbox is being clicked on.

setState() allows the programmer to set manually the checkbox's state.