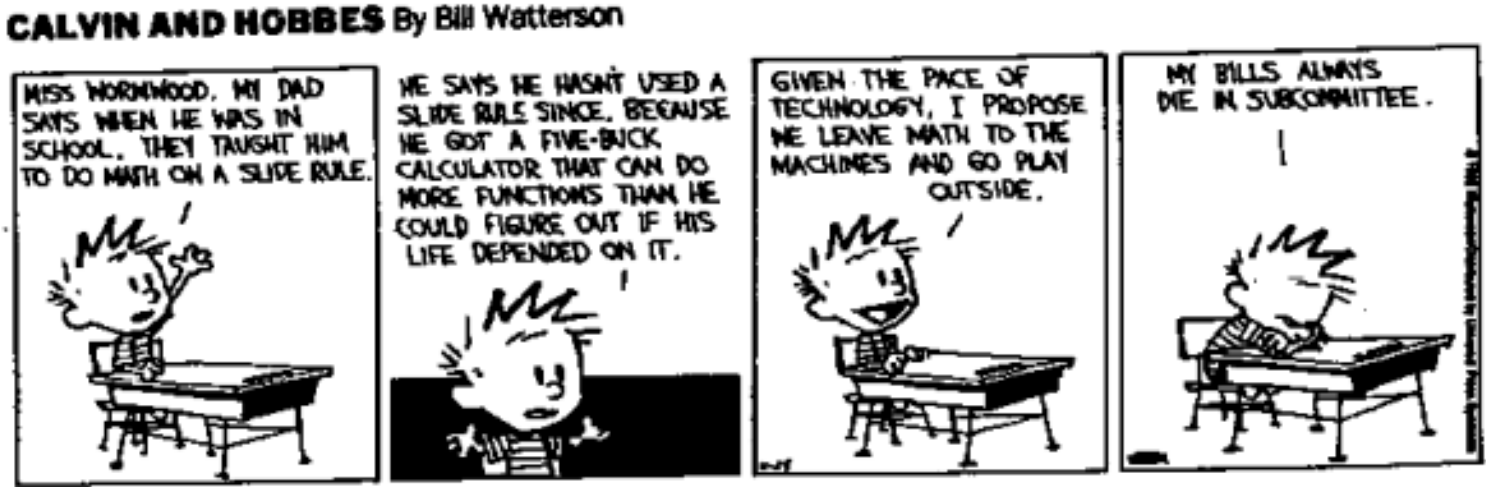
ORF 201

COMPUTER METHODS FOR PROBLEM SOLVING

Lecture 14

Graphical User Interface (GUI)



Class Maze3 Extends Applet

```
public class Maze3 extends Applet
        implements ActionListener
{
    MazeFrame mf;
    Label Nlabel;
    TextField Ntext;
    Button OpenButton;

    public static void main (String[] args) {
        new AppletFrame(new Maze3(), 150, 100);

    }

    public void init()
    { /* more here */
    }

    public void actionPerformed(ActionEvent evt)
    { /* more here */
    }
}
```

Label - A class for displaying text in a window.

Textfield - A class for displaying/inputting text in a window

Button- A class for buttons that can be pushed.

- If run as an application, starts at **`main()`**
- If run as an applet, starts at **`init()`**

AppletFrame is defined in **`myutil`** - details are uninteresting.

Pushing buttons create **`ActionEvent`**s, which are passed to **`actionPerformed()`** for processing.

Init - Where it all begins

`new Label("blah")` makes an instance of a label containing the text blah.

`new Button("My Butt")` makes a button labeled My Butt.

`new TextField("xyz", 8)` makes a text field 8 characters wide initially containing the text xyz. Once placed on a frame, the text can be edited.

```
public void init()
{
     setBackground(Color.white);

     Nlabel = new Label("n: ");
     Ntext = new TextField("10",5);
     OpenButton = new Button("Open Maze Frame");
     OpenButton.addActionListener(this);

     GridPanel gp = new GridPanel();
     gp.add(Nlabel,1,1); gp.add(Ntext, 2, 1);
     gp.add(OpenButton, 1, 2, 3, 1);
     add(gp);
}
```

Placement

- Buttons, textfields, labels, and other objects can be placed directly on a **`Frame`** (using **`add(OpenButton)`** for example) or they can be placed on a **`Panel`** which is placed on a **`Frame`**.
- Placement on a **`Panel`** provides greater flexibility.
- **`GridPanel`** is defined in **`myutil`**. It extends **`Panel`** making it easier to use. Think of the panel consisting of a grid. Objects can be placed in particular cells of the grid.
- **`gp.add(NLabel, x, y)`** puts the label **`NLabel`** in the grid cell that is x cells over from the left edge and y cells down from the top.
- **`gp.add(NLabel, x,y)`** puts the label **`NLabel`** in the grid cell that is **`x`** cells over from the left edge and **`y`** cells down from the top.
- **`gp.add(OpenButton, x, y, w, h)`** puts the button **`OpenButton`** on a rectangular array of grid cells **`w`** wide and **`h`** high with **`(x,y)`** being the upper-left cell.

Action and HandleEvent

Here we indicate that pushing button **OpenButton** will trigger an **ActionEvent** that is passed to actionPerformed in **this** class.

```
public void init()
{
     .
     .
     OpenButton.addActionListener(this);
     .
     .
}

public void actionPerformed(ActionEvent e)
{
     if (e.getSource() == OpenButton) {
          int n = SL.atoi(Ntext.getText());
          mf = new MazeFrame(n);
          mf.show();
     }
}
```

A button push produces an **ActionEvent** object **e** and a call to **actionPerformed()**.

Since many events can produce calls to **actionPerfomed()**, it must first figure out which event it has.

If the event was the pushing of a button **MyButt**, then **e.getSource()** will return a pointer to **MyButt**.

SL.atoi() (defined in **myutil**) converts a string to its integer value (assuming the string represents an integer).

Ntext.getText() (defined in class **TextField**) returns a string representing what currently is showing in the textfield.

Class MazeFrame Extends Frame

```
class MazeFrame extends Frame
{
    int n;
    MazePanel mp;
    Button GenButton, SolveButton, ExitButton;

    public MazeFrame(int n)
    {
        /* more here */
    }

    class ActionEventHandler implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            /* more here */
        }
    }
}
```

MazeFrame is the constructor method. Here, we make the buttons, say where to put them and which event handler will handle them.

Applets have **ActionListener**s built in so all we needed to say was "**implements ActionListener**" on the declaration line.

Frames don't have them built in and so they must be explicitly provided as shown here.

Constructor MazeFrame

```
public MazeFrame(int n)
{
    this.n = n;

    mp = new MazePanel(n);

    ActionEventHandler handler = new ActionEventHandler();

    GenButton = new Button("Generate");
    SolveButton = new Button("Solve");
    ExitButton = new Button("Exit");

    GenButton.addActionListener( handler );
    SolveButton.addActionListener( handler );
    ExitButton.addActionListener( handler );

    Panel bp = new Panel();
    bp.setFont(new Font("Helvetica", Font.BOLD,10));
    bp.add(GenButton);
    bp.add(SolveButton);
    bp.add(ExitButton);

    setLayout(new BorderLayout(10,10));
    add("North", bp);
    add("Center", mp);

    setSize(30*n, 30*n);
}
```

Mazeframe consists of two panels: **bp** and **mp**.

Instantiate an event handler called **handler** and have each button use it

The two panels are arranged on the frame using **BorderLayout**.

setSize(x,y) resets the frame's size to be **x** pixels wide by **y** pixels high.

Borderlayouts provide "North", "South", "east", "West", and "Center" as first arguments to **add()**.

ActionEventHandler's actionPerformed

As before, **`actionPerformed()`** must figure out what the event is.

```
public boolean actionPerformed(ActionEvent e)
{
     if (e.getSource() == ExitButton) {
         dispose();
     } else if (e.getSource() == GenButton) {
         mp.genMaze();
     } else if (e.getSource() == SolveButton) {
         mp.solveMaze();
     }
 }
```

To close a window, call **`dispose()`**.

Class MazePanel Extends GridPanel Extends Panel

```
class MazePanel extends GridPanel
{
    public MazePanel(int n)
    {   /* more here (perhaps)*/
    }

    public void genMaze()
    {
        GL.color(Color.black);
        GL.drawString("genMaze", 2,2,"CENTER","CENTER");
        try {Thread.sleep(1000);} catch(InterruptedException ie);
        paint(getGraphics());
    }

    public void solveMaze()
    {
        GL.color(Color.black);
        GL.drawString("solveMaze now",1,1,"LEFT","CENTER");
        try {Thread.sleep(1000);} catch(InterruptedException ie);
        paint(get Graphics());
    }

    public void paint (Graphics g)
    {
        GL.ginit(getGraphics(), size().width, size().height,this,
        GL.ortho2(0,4,0,4);

        GL.color(Color.gray);
        GL.clear();
    }
}
```