Lecture 14
Graphical User Interface (GUI)
Class Maze Extends Applet

```java
public class Maze extends Applet {
    MazeFrame mf;
    Label Nlabel;
    TextField Ntext;
    Button OpenButton;

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

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

    public boolean action(Event evt, Object arg) {
        /* more here */
    }

    public boolean handleEvent(Event evt) {
        /* more here */
    }
}
```

**Label** - A class for displaying text in a window.

**TextField** - A class for displaying/inputting text in a window.

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

Pushing buttons create "action" `Event` s, where are passed to `action()` for processing.

Closing the window creates an `Event` that gets passed to `handleEvent()`.
Init - Where it all begins

```java
public void init()
{
    setBackground(Color.white);
    Nlabel = new Label("n: ");
    Ntext = new TextField("10", 5);
    OpenButton = new Button("Open Maze Frame");

    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.

- `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.

- `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.
A button push produces an `Event` object `evt` and a call to `action()`. Since many events can produce calls to `action()`, it must first figure out which event it has. If the event was the pushing of a button labeled *My Butt*, then `arg.equals("My Butt")` will evaluate to a `true`. `Ntext.getText()` (defined in class `TextField`) returns a string representing what currently is showing in the textfield.

Events that aren’t handled must be passed up so that they can be handled by a higher authority.

Using the mouse to close the window produces an `Event.WINDOW_DESTROY` which is passed to `handleEvent`. The program just explicitly request to die with `System.exit(0)`.
Class MazeFrame Extends Frame

class MazeFrame extends Frame
{
    int n;
    MazePanel mp;

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

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

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

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

        resize(30*n, 30*n);
    }

    public boolean action(Event evt, Object arg)
    {
        /* more here */
    }
}

MazeFrame consists of two panels: ButtonPanel and mp.

The two panels are arranged on the frame using BorderLayout.

resize(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().
MazeFrame’s Action Method

```java
public boolean action(Event evt, Object arg)
{
    if (arg.equals("Exit")) {
        dispose();
    } else if (arg.equals("Generate")) {
        mp.genMaze();
    } else if (arg.equals("Solve")) {
        mp.solveMaze();
    } else {
        return super.action(evt, arg);
    }
    return true;
}
```

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

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

If the event is handled here, then `true` must be returned.
class MazePanel extends GridPanel
{
    public MazePanel(int n)
    {  /* more here */
        
        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(getGraphics());
        }
        
        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();
        }
    }