CS1004: Intro to CS in Java, Spring 2005

Lecture #8: GUIs, logic design

Janak J Parekh
janak@cs.columbia.edu

Administrivia

- HW#2 out
- New TAs, changed office hours

How to create an Applet

- Your class must *extend* the `Applet` class
  - This makes use of *inheritance* (Chapter 8)
  - You don't need to know how this works in order to write applets
- Next, embed the applet into an HTML file using a tag that references the class file of the applet
- View the HTML file using a web browser or appletviewer
  - The web browser can automatically download the .class file like an image

HelloWorldApplet.java

```
import javax.swing.JApplet;
import java.awt.*;

public class HelloWorldApplet extends JApplet {
  public void paint(Graphics page) {
    page.drawString(“Hello world”, 100, 100);
  }
}
```

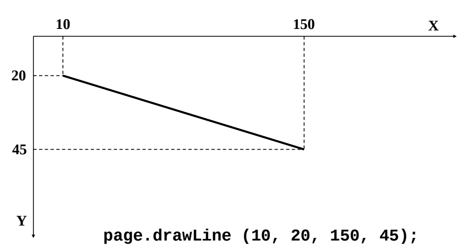
HelloWorldApplet.html

```
<html>
   <head><title>Hello World!</title></head>
   <body>
      <applet code=“HelloWorldApplet.class"
             width=600 height=400>
      </applet>
   </body>
</html>
```

Drawing Shapes

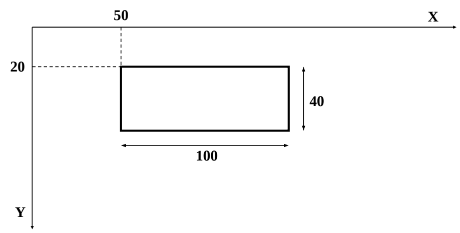
- The `Graphics` class has lots more primitives, including shape drawing
  - Let’s look at the Java API again
  - http://java.sun.com/j2se/1.5.0/docs/api/java/awt/Graphics.html
- Many shapes can be filled or unfilled
- The method parameters usually specify coordinates and sizes
- Shapes with curves, like an oval, are usually drawn by specifying the shape’s *bounding rectangle*
- An arc can be thought of as a section of an oval

Drawing a Line



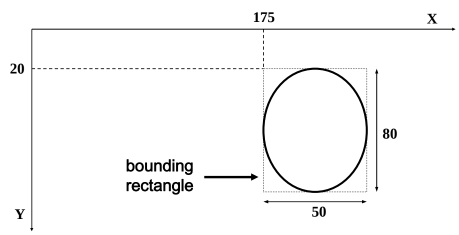

```
page.drawLine (10, 20, 150, 45);
or
page.drawLine (150, 45, 10, 20);
```

Drawing a Rectangle




```
page.drawRect (50, 20, 100, 40);
```

Drawing an Oval




```
page.drawOval (175, 20, 50, 80);
```

Drawing Shapes

- Every drawing surface has a *background color*
  - Your applet is one surface; for multiple backgrounds, use filled rectangles
- Every graphics context has a current *foreground color*
  - Which you can change as the program goes on; like picking up a different crayon
- setBackground(…) and page.setColor(…)
- Let's look at the book's applet (page 103)

Segue

- Back to computer hardware basics
- We'll pick up with more Java next time
- The stuff we covered up until now is what you need for HW#2

Boolean Logic

- Apart from storage, what does a computer do?
- Low-level manipulations consist of **boolean logic** – i.e., operations on true/false values
  - True/false maps easily onto bistable environment
- Boolean logic operations on electronic signals may be built out of transistors and other electronic devices
  - Goal: build computing logic out of these
  - Imagine a simple "elevator controller"

Boolean operations

- a AND b
  - True only when a is true and b is true
- a OR b
  - True when either a is true or b is true, or both are true
  - English "or" is *not* OR (it's XOR)
- NOT a
  - True when a is false, and vice versa
- And every more complex operation is built out of these three

Boolean Logic (continued)

- Boolean expressions
  - Constructed by combining together Boolean operations
  - (a AND b) OR ((NOT b) AND (NOT a))
- *Truth tables* capture the output/value of a Boolean expression
  - A column for each input plus the output
  - A row for each combination of input values

Boolean Logic (continued)

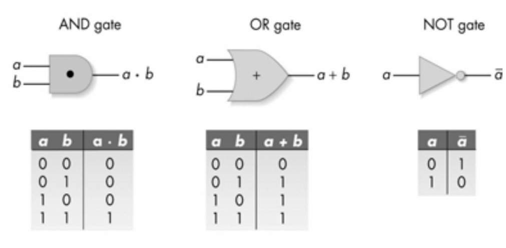
- Example:

  (a AND b) OR ((NOT b) and (NOT a))

a	*b*	*Value*
0	0	1
0	1	0
1	0	0
1	1	1

Gates

- Gates
  - Since logic so common, we design hardware to do this
- AND gate
  - Two input lines, one output line
  - Outputs a 1 when both inputs are 1
- OR gate
  - Two input lines, one output line
  - Outputs a 1 when *either* input is 1
- NOT gate
  - One input line, one output line
  - Outputs a 1 when input is 0 and vice versa



The Three Basic Gates and Their Symbols
(see pages 156-157 for implementation)

Big picture

- Abstraction in hardware design
  - Map hardware devices to Boolean logic
  - Design more complex devices in *terms of logic*, not electronics
  - Conversion from logic to hardware design may be automated
- A *circuit* is a realized collection of logic gates
  - Transforms a set of binary inputs into a set of binary outputs
  - Values of the outputs depend only on the current values of the inputs

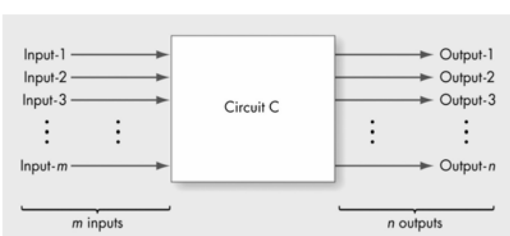


Diagram of a Typical Computer Circuit

A Circuit Construction Algorithm

- Sum-of-products algorithm
  - Truth table captures every input/output possible for circuit
  - Repeat process for each output line
    - Build a Boolean expression using AND and NOT for each 1 of the output line
    - Combine together all the expressions with ORs
    - Build circuit from whole Boolean expression

Two major kinds of circuits

- Computation circuits
  - Take two bits of data and combine them in some fashion
- Control circuits
  - Determine which computation circuits or data bits to use

A few examples of computation circuits

- 1-bit equality
  - Two inputs, one output
- *n*-bit equality
  - Composed of many 1-bit equality circuits ANDed together
- 1-bit adder
  - Three inputs, two outputs
- *n*-bit adder
  - Composed of many 1-bit adders chained together
- Let's do these on the board
  - Pages 165-172

Next time

- Continue computer architecture
- Start Java OO concepts