GUIS: Graphical User Interfaces

Their mouse had a mean time between failure of … a week … it would jam up irreparably, or ... jam up on the table-- ... It had a flimsy cord whose wires would break. Steve Jobs: "... Xerox says it can't be built for < $400, I want a $10 mouse that will never fail and can be mass produced, because it's going to be the primary interface of the computer ..."

... Dean Hovey ... came back, "I've got some good and some bad news. Good news: we've got a new project with Apple. Bad news: I told Steve we'd design a mouse for 10 bucks."

... year later ... we … filed … and were granted a patent, on the electro-mechanical-optical mouse of today; ... we ended up ... [making] the mouse as invisible to people as it is today.

Steve Sachs interview on first computer with GUI: Apple Lisa (~$10K in 1982).
http://library.stanford.edu/mac/primary/interviews/sachs/trans.html

Have you downloaded A4 stuff and put it into Eclipse?

The instructions may be hard to follow. It's best to do this before you leave for the fall break so you can get help if you need it.

Adding the javadoc to the lib file and getting doc.zip in place it not essential. If you don't do it, then when you pass your mouse over a method name its specification won't pop up. That is all.

Have you picked up your prelim 1 from the handback room, Gates 216, and looked it over?

Why do we have a function DiseaseTree.equals? Who would ever want to compare to trees?

Explanation of parsing method

```
/** Return true iff the beginning of the input is an E.
  *  Remove the processed tokens from the input.
  *  Here is the definition of E:  E → integer
                                   E → ( E + E )
  */
public boolean parseE() {
        if (first token is an integer) remove it from input and return true;
        if (first token is not  ‘(‘ ) return false else remove it from input;
        if (!parseE()) return false;
        if (first token is not  ‘+‘ ) return false else remove it from input;
        if (!parseE()) return false;
        if (first token is not  ‘)‘ ) return false else remove it from input;
        return true;
}
```

Explanation of parsing method

$E \longrightarrow$ integer
$E \longrightarrow (E + E)$

input: 654 + (22 + 3)

true

```
public boolean parseE() {
        if (first token is an integer) remove it from input and return true;
        if (first token is not '(' ) return false else remove it from input;
        if (!parseE()) return false;
        if (first token is not '+' ) return false else remove it from input;
        if (!parseE()) return false;
        if (first token is not ')' ) return false else remove it from input;
        return true;
}
```

Explanation of parsing method

$E \longrightarrow$ integer
$E \longrightarrow (E + E)$

input: (72 + (6+2))))) + 2

true

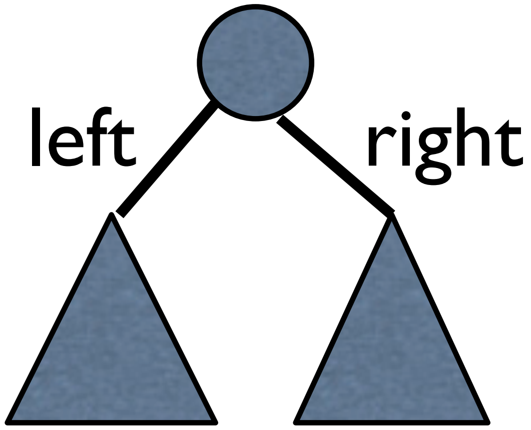
```
public boolean parseE() {
        if (first token is an integer) remove it from input and return true;
        if (first token is not '(' ) return false else remove it from input;
        if (!parseE()) return false;
        if (first token is not '+' ) return false else remove it from input;
        if (!parseE()) return false;
        if (first token is not ')' ) return false else remove it from input;
        return true;
}
```

Try for something simple --always

```
/** Return true iff following properties hold:
 *  1. All values in the tree with this node as
        root are >= min.
 *  2.  All values in the tree with this node as
        root are <= max.
 *  3. This tree is a BST. */
 public boolean isBST(int min, int max) {

        return


 }
```



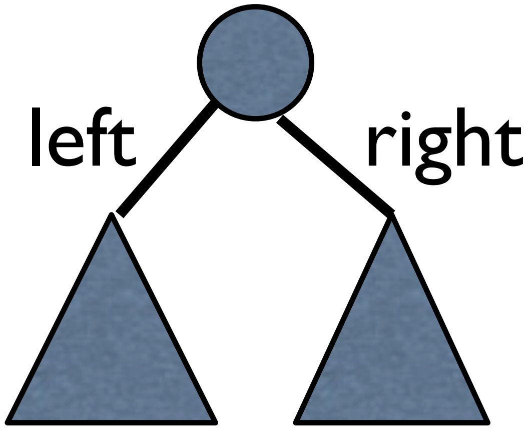
Try to keep things simple!

If things work out, may be able to write a single return statement, with each of the 3 points in it.

Points 1 and 2 for root value

```
/** Return true iff following properties hold:
 *  1. All values in the tree with this node as
       root are >= min.
 *  2.  All values in the tree with this node as
       root are <= max.
 *  3. This tree is a BST. */
public boolean isBST(int min, int max) {

    return    min  <= val   &&   val <= max
```

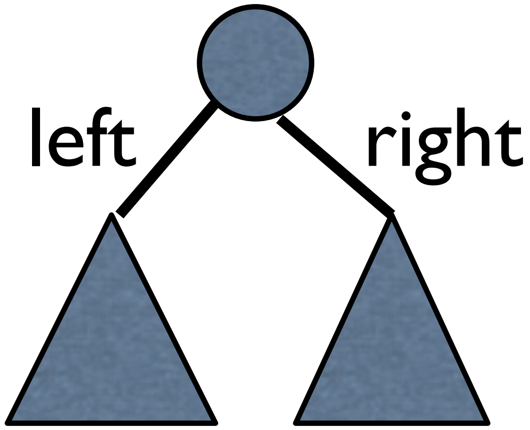


Try to keep things simple!
For points 1 and 2, have to test the root

Points 1 and 2 for subtrees

```
/** Return true iff following properties hold:
 *  1. All values in the tree with this node as
       root are >= min.
 *  2.  All values in the tree with this node as
       root are <= max.
 *  3. This tree is a BST. */
public boolean isBST(int min, int max) {

       return   min  <= val   &&   val <= max          &&
                (left == null || left. isBST(min, max))   &&
                (right == null || right.isBST(min, max))
```

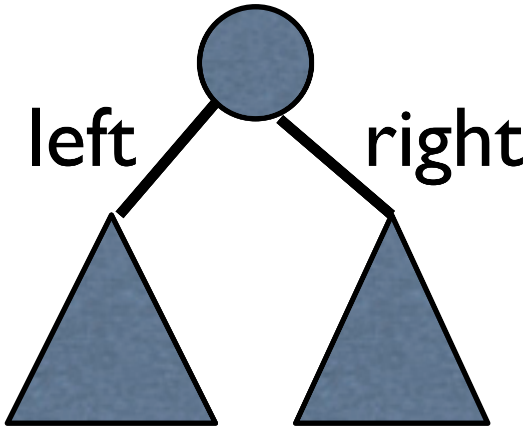


But the subtree values have to be in same range
(If the subtrees exist!!! Always think of this case)
Use recursion

Point 3

```
/** Return true iff following properties hold:
 *  1. All values in the tree with this node as
        root are >= min.
 *  2.  All values in the tree with this node as
        root are <= max.
 *  3. This tree is a BST. */
public boolean isBST(int min, int max) {

      return   min  <= val   &&   val <= max    &&
                 (left == null ||left. isBST(min,  val-1  )  &&
                 (right == null || right.isBST( val+1 , max)) ;
```



That takes care of points 1, 2. Point 3?
Values in left subtree have to be < val.
Change the argument to isBST.   Right subtree similar

GUI (Graphical User Interface)

- Provides a friendly interface between user and program
- Allows event-driven or reactive programming: The program reacts to events such as button clicks, mouse movement, keyboard input
- Often is multi-threaded: Different threads of execution can be going on simultaneously

We use Java's two packages for doing GUIs:

- **AWT** (Abstract or Awful Window Toolkit) —first one
- Swing —a newer one, which builds on AWT as much as possible

Two aspects to making a GUI:

1. Placing components (buttons, text, etc.) in it. TODAY
2. Listening/responding to events Next Lecture

Class JFrame

JFrame object: associated with a window on your monitor.

Generally, a GUI is a JFrame object with various components placed in it

Some methods in a JFrame object		
hide()	show()	setVisible(boolean)
getX()	getY()	(coordinates of top-left point)
getWidth()	getHeight()	setLocation(int, int)
getTitle()	setTitle(String)	
getLocation()	setLocation(int, int)	

Over 100 methods in a JFrame object!

Class JFrame is in package javax.swing

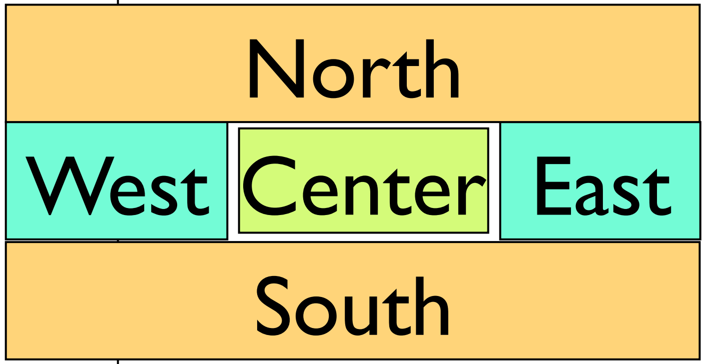
Placing components in a JFrame

Layout manager: Instance controls placement of components.

JFrame layout manager default: BorderLayout.

BorderLayout layout manager: Can place 5 components:

```
public class C extends JFrame {
   public C() {
      Container cp= getContentPane();
      JButton jb= new JButton("Click here");
      JLabel jl= new JLabel( "label 2");
      cp.add(jb, BorderLayout.EAST);
      cp.add(jl, BorderLayout.WEST);
      pack();
      setVisible(true);
  }
```



JFrameDemo.java

Putting components in a JFrame

```
import java.awt.*;    import javax.swing.*;
/** Demonstrate placement of components in a JFrame.
    Places five components in 5 possible areas:
      (1) a JButton in the east,            (2) a JLabel in the west,
      (3) a JLabel in the south,            (4) a JTextField in the north
      (5) a JTextArea in the center.   */

public class ComponentExample extends JFrame {
   /** Constructor: a window with title t and 5 components */
   public ComponentExample(String t) {
      super(t);
      Container cp=  getContentPane();
      cp.add(new JButton("click me"), BorderLayout.EAST);
      cp.add(new JTextField("type here", 22), BorderLayout.NORTH);
      cp.add(new JCheckBox("I got up today"), BorderLayout.SOUTH);
      cp.add(new JLabel("label 2"), BorderLayout.WEST);
      cp.add(new JTextArea("type\nhere", 4, 10), BorderLayout.CENTER);
      pack();
   }
```

Add components to its contentPane

ComponentExample.java

Packages --Components

Packages that contain classes that deal with GUIs:
java.awt: **Old package.** javax.swing: **New package.**

javax.swing has a better way of listening to buttons, text fields, etc. Components are more flexible.

Jxxxx: in Swing, with xxxx in awt.

Component: Something that can be placed in a GUI window. They are instances of certain classes, e.g.

JButton, Button:	Clickable button
JLabel, Label:	Line of text
JTextField, TextField:	Field into which the user can type
JTextArea, TextArea:	Many-row field into which user can type
JPanel, Panel:	Used for graphics; to contain other components
JCheckBox:	Checkable box with a title
JComboBox:	Menu of items, one of which can be checked
JRadioButton:	Same functionality as JCheckBox
Container:	Can contain other components
Box:	Can contain other components

Basic Components

- Component
  - Button, Canvas
  - Checkbox, Choice
  - Label, List, Scrollbar
  - TextComponent
    - TextField, TextArea
  - Container
    - JComponent
    - AbstractButton
      - JButton
      - JToggleButton
        - JCheckBox
        - RadioButton
    - JLabel, JList
    - JOptionPane, JPanel
    - JPopupMenu, JScrollBar, JSlider
    - JTextComponent
    - JTextField, JTextArea

Component: Something that can be placed in a GUI window. These are the basic ones used in GUIs

Note the use of subclasses to provide structure and efficiency. For example, there are two kinds of JToggleButtons, so that class has two subclasses.

Components that can contain other components

```
Component
   Box
   Container
         JComponent
         JPanel
         Panel
            Applet
   Window
         Frame
            JFrame
         JWindow
```

java.awt is the old GUI package.

javax.swing is the new GUI package. When they wanted to use an old name, they put J in front of it.

(e.g. Frame and JFrame)

When constructing javax.swing, the attempt was made to rely on the old package as much as possible.

So, JFrame is a subclass of Frame.

But they couldn't do this with JPanel.

```
import java.awt.*;    import javax.swing.*;
/** Instance has labels in east /west, JPanel with four buttons in center.  */
public class PanelDemo extends JFrame {
    JPanel p= new JPanel();
    /** Constructor: a frame with title "Panel demo", labels in east/west,
        blank label in south, JPanel of 4 buttons in the center */
    public PanelDemo() {
        super("Panel demo");
        p.add(new JButton("0"));   p.add(new JButton("1"));
        p.add(new JButton("2"));   p.add(new JButton("3"));
        Container cp= getContentPane();
        cp.add(new JLabel("east"), BorderLayout.EAST);
        cp.add(new JLabel("west"), BorderLayout.WEST);
        cp.add(new JLabel("    "), BorderLayout.SOUTH);
        cp.add(p, BorderLayout.CENTER);
        pack();
    }
}
```

JPanel as a container

JPanel layout manager default: FlowLayout.

FlowLayout layout manager: Place any number of components. They appear in the order added, taking as many rows as necessary.

```
import javax.swing.*;  import java.awt.*;
/** Demo class Box. Comment on constructor says how frame is laid out. */
public class BoxDemo extends JFrame {
   /** Constructor: frame with title "Box demo", labels in the east/west,
        blank label in south, horizontal Box with 4 buttons in center. */
   public BoxDemo() {
      super("Box demo");
      Box b= new Box(BoxLayout.X_AXIS);
      b.add(new JButton("0"));    b.add(new JButton("1"));
      b.add(new JButton("2"));    b.add(new JButton("3"));
      Container cp= getContentPane();
      cp.add(new JLabel("east"),  BorderLayout.EAST);
      cp.add(new JLabel("west"), BorderLayout.WEST);
      cp.add(new JLabel(" "),      BorderLayout.SOUTH);
      cp.add(b,                            BorderLayout.CENTER);
      pack();
   }
}
```

Class Box: a container

Box layout manager default: BoxLayout.

BoxLayout layout manager: Place any number of components. They appear in the order added, taking only one row.

```
public class BoxDemo2 extends JFrame {
 /** Constructor: frame with title t and 3 columns with n, n+1, and n+2 buttons. */
 public BoxDemo2(String t, int n) {
      super(t);
      // Create Box b1 with n buttons.
            Box b1= new Box(BoxLayout.Y_AXIS);
            for (int i= 0;  i != n;  i= i+1)
                  b1.add(new JButton(”1 " + i));
      // Create Box b2 with n+1 buttons.
            Box b2= …
      // Create Box b3 with n+2 buttons.
            Box b3= …
      // Create horizontal box b containing b1, b2, b3
            Box b= new Box(BoxLayout.X_AXIS);
            b.add(b1);
            b.add(b2);
            b.add(b3);
      Container cp= getContentPane();
      cp.add(b, BorderLayout.CENTER);
      pack();  show();
 }
```

Boxes within a Box

3 vertical boxes, each a column of buttons, are placed in a horizontal box

BoxLayout layout manager: Place any number of components. They appear in the order added, taking only one row.

Simulate BoxLayout Manager in a JFrame

To simulate using a BoxLayout manager for a JFrame, create a Box and place it as the sole component of the JFrame:

```
JFrame jf= new JFrame("title");
Box b= new Box(BoxLayout.X_AXIS);
Add components to b;
jf.add(b,BorderLayout.CENTER);
```

1. **Start developing a GUI by changing an already existing one.** A lot of details. Hard to get all details right when one starts from scratch and has little idea about the Java GUI package.

2. Showed how to place components in a GUI. Next time: how to "listen" to things like button clicks in a GUI.