**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 is 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?**

Try for something simple --always

```java
/** Return true if 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
}

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, val-1) ) &&
            (right == null || right.isBST(val+1, max));
}

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)
getLocation() setLocation(int, int)
getName() setFont(String)
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 = new JLabel("label 2");
        cp.add(jb, BorderLayout.EAST);
        cp.add(jl, BorderLayout.WEST);
        pack();
        setVisible(true);
    }
}
### Putting components in a JFrame

```java
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 JFrame with title "Component Example", labels in east/west,
     * text fields, etc. Components are more flexible.
     */
    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 JLabel("I got up today"), BorderLayout.SOUTH);
        cp.add(new JTextArea("type here", 4, 10), BorderLayout.CENTER);
        pack();
    }
}
```

### Packages -- Components

- **javax.swing**: Old package.
- **javax.swing**: New package.

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

### Components that can contain other components

- **Box**: Same functionality as JCheckBox
- **JComboBox**: Many of items, one of which can be checked
- **JComponent**: Something that can be placed in a GUI window. They are instances of certain classes, e.g. JButton, Button, JLabel, Line of text
- **JFrame**: Same functionality as JFrame
- **JPanel**: Used for graphics; to contain other components
- **JToolBar**: Menu of items, one of which can be checked

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.

### Basic Components

- **Button**, **Canvas**
- **Checkbox**, **Choice**
- **Label**, **List**
- **ScrollBar**, **TextArea**
- **TextField**, **JTextArea**
- **JCheckBox**, **JList**
- **JComboBox**, **JOptionPane**
- **JPanel**, **JScrollPane**
- **JTextArea**, **JSlider**
- **JTextField**, **JTextComponent**

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

### Component Example java

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

/**
 * Constructor: frame with title "Box Demo", labels in the east/west,
 * text fields, etc. Components are more flexible.
 */
public class BoxDemo extends JFrame {
    /**
     * Constructor: a frame with title "Panel Demo", labels in east/west,
     * blank label in south, Panel of 4 buttons in center.
     */
    public BoxDemo() {
        super("Box demo");
        Box b = Box.createHorizontalBox();
        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 Label("east"), BorderLayout.EAST);
        cp.add(new Label("west"), BorderLayout.WEST);
        cp.add(new Label("south"), BorderLayout.SOUTH);
        cp.add(b, BorderLayout.CENTER);
        add(b, BorderLayout.NORTH);
        pack();
    }
}
```

### JPanel as a container

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

### Class Box: a container

- **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();
    }
}

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:

```java
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.