

What produced this pseudocode?

Pseudo Code

Info

heapSort/shiftDown

shiftDown(list, current)

```
1 while( there has been a swap and current has at least one child )
2   if( current has only one child )
3     biggestChild = leftchild
4   else if( leftchild >= rightchild )
5     biggestChild = leftchild
6   else
7     biggestChild = rightchild
8   if( current biggestChild )
9     swap(current, biggestChild)
```

Variables:

current = 2

biggestChild = 4

In the sho file, there was a pseudocode tag (all on one line)

```
<pseudocode_url>fhtemp.php
    ?line=8
    &var [p]=heapSort/shiftDown
    &var [currentNode]=2
    &var [biggestChild]=4</pseudocode_url>
```

In “debug mode” you need the full path name

```
<pseudocode_url>http://www.cis.gvsu.edu/~vizxx/html_root/doc/heapmyles/fhtemp.php
    ?line=8
    &var [p]=heapSort/shiftDown
    &var [currentNode]=2
    &var [biggestChild]=4</pseudocode_url>
```

Where the file fhtemp.php looks like

```
<html>
<head>
<title>Program Listing</title>
<!-- you will need this style sheet declaration if you want red highlighting -->
<style>
  <!--
      @import url(../php_inc/program_listing.css);
  -->
</style>
</head>
<body>
<?php $file = 'heapDown.xml'; ?>
<?php include '../php_inc/program_listing.php'; ?> <!-- this is the only required tag -->
</body>
</html>
```

And the file heapDown.xml looks like

```
<?xml version="1.0"?>
<pseudocode>
  <call_stack>heapSort/shiftDown</call_stack>
  <program_listing>
    <signature>shiftDown(list, current)</signature>
    <line line_number="1">1   while( there has been a swap and current has at least one child )</line>
    <line line_number="2">2   if( current has only one child )</line>
    <line line_number="3">3       biggestChild = leftchild</line>
    <line line_number="4">4       else if( leftchild >= rightchild )</line>
    <line line_number="5">5         biggestChild = leftchild</line>
    <line line_number="6">6         else</line>
    <line line_number="7">7         biggestChild = rightchild</line>
    <line line_number="8">8       if( current < biggestChild )</line>
    <line line_number="9">9         swap(current, biggestChild)</line>
  </program_listing>
  <variables>
    <variable>current = <replace var="currentNode" /></variable>
    <variable> biggestChild = <replace var="biggestChild" /></variable>
  </variables>
</pseudocode>
```

Why, when your bubble sort runs on the server, do we see this instead of having to use command line parameters?



It's because there's an input generator “igs” file adhering to the following DTD

```
<!ELEMENT input_panel ( (textfield | combobox)* )>
<!ELEMENT textfield (label_line*, default_field, value_entered)>
<!ELEMENT label_line (#PCDATA)>
<!ELEMENT default_field (#PCDATA)>
<!ELEMENT value_entered (#PCDATA)>
<!ELEMENT combobox (label_line*, option*, option_entered)>
<!ELEMENT option (#PCDATA)>
<!ELEMENT option_entered (#PCDATA)>
```

The *myles_bubsort_example.igs* looks like ...

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE show PUBLIC "-//JHAVE//DTD INPUT PANEL//EN" "input_panel.dtd">

<input_panel>
  <textfield>
    <label_line>Enter the size of the array to be sorted:</label_line>
    <default_field>10</default_field>
    <value_entered></value_entered>
  </textfield>
</input_panel>
```

If you want to put your demo for Friday (or future visualizations you create) on the jhave.org server, send me a directory by ??? today containing ...

- Your Java source code ... we will:
 - ▶ Add an appropriate package statement to it
 - ▶ “Relativize” any pseudo or doc url paths in it
- Any pseudo or doc files
- Your igs file (if needed)

Andrew and Joe have graciously volunteered to make it all work

If you don't like GAIGS's built-in data structures, you do what every OO programmer does – extend and plug-in! I

The first step is to decide on the XML for your extension. That will essentially define its syntax in a sho file. For example:

```
<!ELEMENT show (snap+, questions?)>
```

```
<!-- Add other structure types to the (x|y|z)* part of snap -->
```

```
<!ELEMENT snap (title, doc_url?, pseudocode_url?,  
    (tree|array|graph|stack|queue|linkedlist| bargraph|node|foobar)*,  
    question_ref?)>
```

```
<!ELEMENT title (#PCDATA)>
```

```
<!ELEMENT name (#PCDATA)>
```

```
<!ELEMENT label (#PCDATA)>
```

```
<!ELEMENT doc_url (#PCDATA)>
```

```
<!ELEMENT pseudocode_url (#PCDATA)>
```

If you don't like GAIGS's built-in data structures, you do what every OO programmer does – extend and plug-in! II

```
<!ELEMENT bounds (EMPTY)>
<!ATTLIST bounds x1 CDATA #REQUIRED
                 y1 CDATA #REQUIRED
                 x2 CDATA #REQUIRED
                 y2 CDATA #REQUIRED
                 fontsize CDATA "0.03">

<!-- Here we define foobar -->
<!ELEMENT foobar (name?, bounds?, nodelabel)>
<!ATTLIST foobar x CDATA "0.5"
                 y CDATA "0.5"
                 color CDATA "white">
<!ELEMENT nodelabel (EMPTY)>
<!ATTLIST nodelabel text CDATA "">
```

The rest of the DTD remains unchanged.

Consider a sho file adhering to this DTD and defining a "foobar" I

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- <!DOCTYPE show PUBLIC "-//JHAVE//DTD GAIGS SHO//EN"
      "gaigs_sho.dtd"> -->
<!DOCTYPE show SYSTEM "gaigs_sho.dtd">

<show>
  <snap>
    <title>An example of a simple foobar</title>
    <foobar x="0.25" y="0.25" color="#FF0000">
      <name>foobar1</name>
      <bounds x1="0.2" y1="0.1" x2="0.8" y2="0.75"  fontsize="0.1"/>
      <nodelabel text="The first foobar"/>
    </foobar>
  </snap>

  <snap>
    <title>An example of two simple foobars</title>
```

Why?

Consider a sho file adhering to this DTD and defining a “foobar” II

```
<foobar x="0.25" y="0.25" color="#0000FF">
  <name>foobar2</name>
  <bounds x1="0.0" y1="0.0" x2="0.45" y2="0.45"  fontsize="0.1"/>
  <nodelabel text="The second"/>
</foobar>
<foobar x="0.25" y="0.25" color="#00FF00">
  <name>foobar3</name>
  <bounds x1="0.45" y1="0.45" x2="0.9" y2="0.9"  fontsize="0.1"/>
  <nodelabel text="The third"/>
</foobar>
</snap>

</show>
```

You need to extend the StructureType class in the GAIGS source code

```
StructureType
|-- Md_Array
|-- LinearList
    |-- Stack
    |-- Queue
    |-- LinkedList
    |-- Bar
|-- BinaryTree
|-- GeneralTree
|-- Graph_Network
    |-- Ggraph
    |-- Network
```

To actually extend StructureType, you'll need access to all the code in the repository

- Fire up eclipse
- Start a new project of the “CVS checkout” variety
 - ▶ Host: csf11.acs.uwosh.edu
 - ▶ Repository: /home/naps/cvsroot
 - ▶ User: anonymous
 - ▶ Password: anonymous
 - ▶ Connection type: pserver
- Module name: jhave2
- Create appropriate user.properties files in jhave2/server and jhave2/client (next page)
- Open the *build.xml* file – build and run your own client, server, or both

The user.properties files that work for your logins here at GVSU:

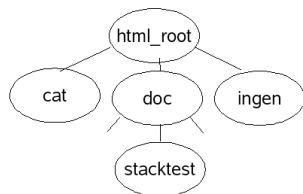
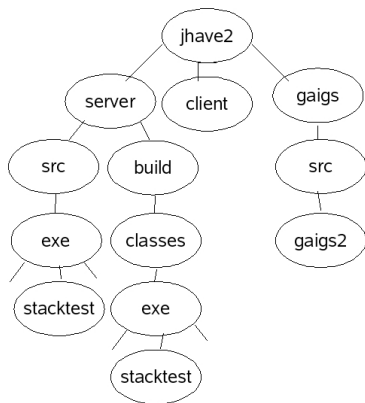
- Client:

```
default.server.url=eosxx.cis.gvsu.edu
default.server.port=7004
default.local.port=7004
default.webroot=http://www.cis.gvsu.edu/~viz01/html_root/
default.category=generic
application.args=-debug
```

- Server:

```
application.args=-r=http://www.cis.gvsu.edu/~vizxx/ -s=./build/classes/ -u=/home/viz01/public_html
```

How all this meshes together ...



Next you need to go into the gaigs/src/gaigs2 directory and extend the StructureType class there I

```
// All GAIGS Structures inherit from ...
abstract class StructureType {

    // StructureType has only one parameterless constructor.

    // All derived structures should provide their own parameterless
    // constructor and override the following two methods:

    // Load the structure from the root of its XML tree (JDOM style)
    void loadStructure(Element myRoot, LinkedList llist, draw d)

    // All derived StructureTypes should override this -- be sure to
    // call super on this method when your structure is empty
    void drawStructure (LinkedList llist, draw d)
```

Parse

And render ...sort of

Next you need to go into the gaigs/src/gaigs2 directory and extend the StructureType class there II

```
// *** USEFUL PROTECTED METHODS AND VARIABLES

// Access your GKS graphics routines thru this. send drawing
// commands to this in normalized [0,1]x[0,1] coordinates,
// describing your position within the bounds given to LGKS (whole
// draw-space default).
protected LocalizedGKS LGKS;

// For derived objects who want to know their bounds
protected double structure_fontsize, structure_left,
                structure_right, structure_bottom, structure_top;

// Load name and bounds info common to all localized derived structures
public void load_name_and_bounds(Element my_root, LinkedList llist, draw d)

// Given s, return its normalized width
protected double normalized_width(String s)

// Given color as a (usually hex) string, convert it to the right
// Java color as int
protected int colorStringToInt(String color)

// Given a fill-area color as a (usually hex) string, convert text
// to appear in the fill area to the right Java color as int
protected int colorStringToTextColorInt(String color)
```

Often used to size a box or circle around text

Your structure's LGKS object is what you “draw” with I

```
// YOUR LGKS OBJECT RESPONDS TO THE FOLLOWING MESSAGES
// The LinkedList and draw objects always tag along because ...

// Set the interior style for a fill area. The int color is
// typically obtained from the hex string by StructureType's
// colorStringToInt method. Sorry, only the constants bsClear and
// bsSolid are presently supported for style. And, of course, a
// clear fill area is just ...
public void set_fill_int_style(int style, int color, LinkedList seginfo, draw d)

// Draw a fill area with the specified number of points and their
// coordinates
public void fill_area(int numpts, double ptsx[], double ptsy[],
                    LinkedList seginfo, draw d)

// Draw a polyline with the specified number of points and their
// coordinates
public void polyline(int numpts, double ptsx[], double ptsy[],
                   LinkedList seginfo, draw d)

// Set the text alignment. Choices for horiz and vert are:
//     final static int TA_CENTER    = 0;
//     final static int TA_LEFT     = 1;
//     final static int TA_RIGHT    = 2;
//
//     final static int TA_BASELINE  = 0;
//     final static int TA_BOTTOM    = 1;
//     final static int TA_TOP       = 2;
public void set_text_align(int horiz, int vert, LinkedList seginfo, draw d)
```

Your structure's LGKS object is what you “draw” with II

```
// Set the color (as a Java int) for drawing text. Usually this  
// int is obtained from the hex string by your having called  
// colorStringToTextColorInt in StructureType.java  
public void set_textline_color(int color, LinkedList seginfo, draw d)  
  
// Change the font size  
public void set_text_height(double height, LinkedList seginfo, draw d)  
  
// Draw your text at the specified coordinate  
public void text(double x, double y, String str, LinkedList seginfo, draw d)  
  
// Draw an ellipse from start angle thru end angle  
public void ellipse(double x, double y, double stangle, double endangle,  
                    double xradius, double yradius,  
                    LinkedList seginfo, draw d)  
  
// Draw a (outlined) circle  
public void circle(double x, double y, double radius, LinkedList seginfo, draw d)  
  
// Draw a filled circle  
public void circle_fill(double x, double y, double radius,  
                       LinkedList seginfo, draw d)
```

Use the JDOM XML parsing class to write loadStructure and add your *foobar* class to the gaigs2 directory I

JDOM – www.jdom.org:

- The Element class provides the type of nodes in the XML tree, e.g.,

```
public void loadStructure(Element rootEl, LinkedList thingsToRender, draw drawerObj)
```
- Given an Element, you can use *getChild* to get the first child and *getChildren* to return a *List* of children
- *getText* returns the text (as a String) of a node in the tree
- *getAttributeValue(String which-attrib)* return the value of a particular attribute

Here's the resulting *foobar.java* file |

```
////////////////////////////////////  
// A sample (and simple) extension of the GAIGS StructureType  
package gaigs2;  
  
import java.awt.image.*;  
import java.util.*;  
import org.jdom.*;  
  
public class foobar extends StructureType {  
  
    double circle_center_x, circle_center_y; // center coords  
    double circle_rad;           // radius  
  
    int circle_color;           // node color  
  
    int circle_labelcolor;      // text color  
    String circle_label;       // only set up for a single line of text  
  
    // Must provide a parameterless constructor for instantiation via reflection  
    public foobar() {  
        super();               // necessary  
  
        circle_color = White;   // our hex notation is "#RRGGBB"  
        circle_labelcolor = Black;  
        circle_label = null;  
        circle_center_x = 0.50;  
        circle_center_y = 0.50;  
        circle_rad = 0.25;  
    } // foobar()  
  
    // This initialization method gets passed a jdom.Element whose  
    // name is "foobar". So gaigs_sho.dtd must be modified, adding a  
    // "foobar" element to the list of structure types a snap can  
    // contain.  
    public void loadStructure(Element rootEl, LinkedList thingsToRender, draw drawerObj) {
```

Here's the resulting *foobar.java* file II

```
// This call loads the name and bounds if your xml
// structure-element has a name and/or bounds like the
// built-in structures.
load_name_and_bounds(rootEl, thingsToRender, drawerObj);

// JDOM, AT LEAST AS MUCH AS WE NEED IT, IS EASY TO USE
List children = rootEl.getChildren(); // getChildren returns a list
Iterator iter = children.iterator(); // which we will iterate through

Element labelEl;

// NOTE: This is only an unnecessary illustrative loop, since
// we could get what we want directly
while( iter.hasNext() ) {
    Element child = (Element) iter.next(); // walk through the list of children

    if( child.getName().equals("name") ) {
        // Just showing we could get it if we wanted,
        // but already done for us in load_name_and_bounds(..)
        String junkName;
        junkName = child.getText(); // get the text of this node in the XML tree
    }
    else if( child.getName().equals("bounds") ) {
        // Just showing we could get it if we wanted,
        // but already done for us in load_name_and_bounds(..)
        double junkBound;
        junkBound = Format.atof( child.getAttributeValue("x1") ); // get an attribute
    }
    else if( child.getName().equals("nodelabel") )
        labelEl = child;
} // End illustrative loop

// In this example, we could get the element we want directly --
labelEl = rootEl.getChild("nodelabel");
```

Here's the resulting *foobar.java* file III

```
// The XML is validated against the DTD, so if there is a
// #REQUIRED attribute or a default value we can safely assume
// it is there
circle_center_x = Format.atof( rootEl.getAttributeValue("x") );
circle_center_y = Format.atof( rootEl.getAttributeValue("y") );

circle_color = colorStringToInt( rootEl.getAttributeValue("color") );

if( labelEl != null ) {
    circle_label = labelEl.getAttributeValue("text");
    circle_labelcolor = colorStringToTextColorInt( rootEl.getAttributeValue("color") );
    circle_rad = (normalized_width(circle_label)+ structure_fontsize) / 2.0;
}
} // loadStructure

// Use the LGKS object to draw the structure
public void drawStructure(LinkedList thingsToRender, draw drawerObj) {
    // draw the circle(filled)
    LGKS.set_fill_int_style(bsSolid, circle_color, thingsToRender, drawerObj);
    LGKS.circle_fill(circle_center_x, circle_center_y, circle_rad, thingsToRender, drawerObj);

    // draw the circle outline
    LGKS.set_textline_color(Black, thingsToRender, drawerObj);
    LGKS.circle(circle_center_x, circle_center_y, circle_rad, thingsToRender, drawerObj);

    // draw the label
    LGKS.set_textline_color(circle_labelcolor, thingsToRender, drawerObj);
    LGKS.set_text_align(TA_CENTER, TA_BOTTOM, thingsToRender, drawerObj);
    LGKS.text(circle_center_x, circle_center_y, circle_label, thingsToRender, drawerObj);
} // drawStructure
} // class foobar
```