



# University of Connecticut

## Applets in Java using NetBeans as an IDE Creating an Interactive Browser using JEditorPane (Part 3)

C.W. David

Department of Chemistry

University of Connecticut

Storrs, CT 06269-3060

[Carl.David@uconn.edu](mailto:Carl.David@uconn.edu)

### The Next Step:

A screenshot of a Java applet window. At the top center, there is a button labeled "choose a question". Below this, a large rectangular area contains the word "Hello" in red text. At the bottom of the window, there is a text prompt "Enter your answer in scientific notation here:" followed by a "Submit Your Answer" button and a scientific notation input field. The input field contains the text "+ 1 . 0 0 E + 0 0", where each character is in a separate dropdown menu.

“We have several alternatives now, which we need to address. Here is a list of them:

1. How to get HTML interpreted correctly.”

So, after getting it wrong, before, we try and get it right now, using a JEditorPane just as before, but!

```
/*
 * applet1.java
 *
 * Created on October 26, 2005, 1:28 PM
 */

/**
 *
 * @author david
 */
import javax.swing.JEditorPane.*;
import javax.swing.text.html.*;
import java.beans.ExceptionListener;
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
import java.math.*;
import java.util.Formatter;
import javax.swing.event.*;
import java.net.URL;
import java.io.*;
public class applet1_1 extends javax.swing.JApplet {

    /** Initializes the applet applet1 */
    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                    initOurCommands();
                    setSize(600,600);
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

```

/** This method is called from within the init() method to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
SNIP SNIP SNIP =====
    jLabel3.setText("Enter your answer in scientific notation here:
");
    outputLabel = jLabel3;

    jButton1.setText("Submit Your Answer");

    jEditorPane1.setEditable(false);
    jEditorPane1.setFont(new java.awt.Font("Times New Roman", 0, 14));
    cwdText = jEditorPane1;
    jEditorPane1.setContentType("text/html");
    jEditorPane1.setText("<h1><font color = \"red\">Hello</font></h1>");
    jEditorPane1.addHyperlinkListener(new javax.swing.event.HyperlinkListener() {
        public void hyperlinkUpdate(javax.swing.event.HyperlinkEvent evt) {
            jEditorPane1HyperlinkUpdate(evt);
        }
    });

    jScrollPane1.setViewportView(jEditorPane1);

    jButton2.setText("choose a question");
SNIP SNIP SNIP =====
// </editor-fold>

    private void jEditorPane1HyperlinkUpdate(javax.swing.event.HyperlinkEvent evt) {
// TODO add your handling code here:
        // @author Santhosh Kumar T - santhosh@in.fiorano.com
        if(evt.getEventType()==HyperlinkEvent.EventType.ACTIVATED){
            try{
                ((JEditorPane)evt.getSource()).setPage(evt.getURL());
            } catch(Exception ex){
                ex.printStackTrace();
            }
        }
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:

        try {

```

```

//this has to be set differently in the final distribution
url = new URL("http://web.uconn.edu/~cdavid/SureMath.html");
/*
 *obtains the menu file from which questions are chosen
 */
} catch (java.net.MalformedURLException evt1){
    System.out.println("Oi, Malformed URL here");
}
try{
    cwdText.setPage(url);
} catch (java.io.IOException evt2){
    System.out.println("setPage failed");
}
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:

    String pm_s = (String)cs1.getSelectedItem();

    String mantissa =
(String)pm_s+cb1.getSelectedItem()+ "." +cb2.getSelectedItem()+cb3.getSelectedItem();
    double mant = Double.valueOf(mantissa).doubleValue();
    //we have just gotten the x.yz (signed) part of a student's answer!

    String pm_s2 = (String)cs2.getSelectedItem();
    String exp = (String)pm_s2+cb4.getSelectedItem()+cb5.getSelectedItem();
    double exponent = Double.valueOf(exp).doubleValue();
    //we have just gotten the exponent of the power of ten.

    student_answer = Math.pow(10,exponent)*mant;

    String answer_formatted =
String.format(Double.toString(student_answer),"%9.2g");
    correct_answer = 1.22;
    String correct_formatted =
String.format(Double.toString(correct_answer),"%9.2g");
    //temporary
    correct_answer = 1.22;

    if(answer_formatted.equals(correct_formatted)){
        outputLabel.setForeground(java.awt.Color.GREEN);
        outputLabel.setText("Enter your answer in scientific notation here; your prior
answer, "+answer_formatted+", was RIGHT!");
    }else{

```

```

        outputLabel.setForeground(java.awt.Color.RED);
        outputLabel.setText("Enter your answer in scientific notation here; your prior
answer, "+answer_formatted+", was WRONG!");
    }
}

public void initOurCommands(){
}

// Variables declaration - do not modify
// End of variables declaration
public static JEditorPane cwdText;
public static JComboBox cb1,cb2,cb3,cb4,cb5,cs1,cs2;
public static JLabel outputLabel;
public double student_answer, correct_answer;

public static URL url = null;
}

```

This time, we leave out a large portion of the code generated by the IDE, concentrating on that part which the IDE inserts under our control! One of the crucial changes was to add the post-initialization code:

```
jEditorPane1.setContentType("text/html");
```

which helps in the conversion of the JEditorPane into a active browser of the kind a student our be expecting.

We have omitted code concerning accepting student responses and echoing them to the screen, as well as code grading those responses against the (presumed) correct answer. The reason for this is we decided to create a javaBean to do this, rather than include it in the driving program. Instead, this example focuses on the single line:

```
cwdText.setPage(url);
```

which differs from part 2, and solves the problem at hand.

However, this scheme was abandoned, since as noted above, I decided to switch to a bean for the student floating point number input module, and also decided, tentatively, to spawn the default browser so I could include JavaScript in the student's problem, since we need to create random numbers to that each student gets his/her own unique problem. Be that as it may, the code above, suitably modified, generates a WWW browser of sorts.

