import javax.swing.event.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.BorderLayout;
import java.net.URL;
import java.awt.FlowLayout;
import java.awt.Component;

/**  *
 * Simple illustration of GUI construction for a rudimentary *
 * web-browser. *
 * Illustrates Layouts, Buttons, Menus, and simple GUI construction.
 * Exercises for the reader include making the buttons functional,
 * adding bookmarks, and more. This can be done via extension
 * or by modifying the existing class.
 *
 * When creating a browser, client code should set the size of
 * of the frame (note that Browser extends JFrame).
 *
 * Construct the web browser, calling code should set size of frame.
 */
public class Browser extends JFrame {
    private JEditorPane myEditor;  // displays web page
    private JLabel myNextURL;  // if link clicked, go here
    private JTextField myURLDisplay;  // user-entered url
    private JButton myBackButton;  // generates call of doBack()
    private JButton myNextButton;  // generates call of doNext()
    private JButton myGoButton;  // goes to file located in myURLDisplay
    private JButton myHomeButton;  // goes to myHome
    private String myHome;  // url of user selected homepage
    private JMenuItem myHomeMenuItem;  // home in Go menu
    private JMenuItem myNextMenuItem;  // next in Go menu
    private JMenuItem myBackMenuItem;  // back in Go menu
    private JMenuBar menubar;  // where all menus go
    private JEditorPane.setEditable(false);  // allows links to be followed
    myHome = "";  // no home page, but not null
    initGui();  // displays web page

    /**  *
     * Make menus
     */
    private void makeMenu() {
        JMenuBar menubar = new JMenuBar();  // where all menus go
        // create file menu, add quit to it
        JMenu fileMenu = new JMenu("File");
        menubar.add(fileMenu);
        fileMenu.add(new AbstractAction("Quit") {
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });

        // create go menu, for next/back/home
        JMenu goMenu = new JMenu("Go");
        menubar.add(goMenu);
        // store in instance variables back/next menu items
        myBackMenuItem = new JMenuItem(new AbstractAction("Back"){
            public void actionPerformed(ActionEvent e){
                doBack();
            }
        });
        goMenu.add(myBackMenuItem);
        myNextMenuItem = new JMenuItem(new AbstractAction("Next"){
            public void actionPerformed(ActionEvent e){
                doNext();
            }
        });
        goMenu.add(myNextMenuItem);
        myHomeMenuItem = new JMenuItem(new AbstractAction("Home"){
            public void actionPerformed(ActionEvent e){
                doHome();
            }
        });
        goMenu.add(myHomeMenuItem);

        // tools menu, set home page for now
        final JMenu toolsMenu = new JMenu("Tools");
        menubar.add(toolsMenu);
        toolsMenu.add(new AbstractAction("Set homepage"){
            public void actionPerformed(ActionEvent e){
                myHome = myURLDisplay.getText();
            }
        });
        setJMenuBar(menubar);
    }

    /**  *
     * Generates appropriate buttons for the browser.
     * Back, Next, Home are enabled, arguably should NOT be at first
     * though back/next should not be enabled at first.
     *
     * @return returns a JPanel containing all the buttons
     */
    private JPanel makeButtons() {
        myBackButton = new JButton("back", new ImageIcon("back.gif"));
        myNextButton = new JButton("next", new ImageIcon("next.gif"));
        myHomeButton = new JButton("home", new ImageIcon("home.gif"));
        // add listener so backButton calls doBack()
        myBackButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) { doBack();
            }
        });
        // add listener so nextButton calls doNext()
        myNextButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
        }
    }

    // construct browser, calling code should set size of frame
    public Browser() {
        myEditor = new JEditorPane();
        myURLDisplay.setEditable(false);
        myHome = "";
        initGui();
    }

    // construct browser, calling code should set size of frame
    public Browser(String myHome) {
        myEditor = new JEditorPane();
        myURLDisplay.setEditable(false);
        myHome = myHome;
        initGui();
    }

    private void initGui() {
        // set up the components
        myNextURL = new JLabel("Next URL");
        myURLDisplay = new JTextField("User entered URL");
        myBackButton = new JButton("Back");
        myNextButton = new JButton("Next");
        myGoButton = new JButton("Go");
        myHomeButton = new JButton("Home");

        // add action listeners
        myBackButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Back Button clicked");
            }
        });
        myNextButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Next Button clicked");
            }
        });
        myGoButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Go Button clicked");
            }
        });
        myHomeButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Home Button clicked");
            }
        });

        // add listeners to the JEditorPane
        myEditor.addHyperlinkListener(new HyperlinkListener() {
            public void hrefSelected(HyperlinkEvent e) {
                System.out.println("Hyperlink selected");
            }
            public void urlChange(HyperlinkEvent e) {
                System.out.println("URL changed");
            }
        });

        // add listeners to the JLabels
        myNextURL.addMouseMotionListener(new MouseMotionListener() {
            public void mouseDragged(MouseEvent e) {
                System.out.println("Mouse dragged");
            }
            public void mouseMoved(MouseEvent e) {
                System.out.println("Mouse moved");
            }
        });
        myNextURL.addMouseWheelListener(new MouseWheelListener() {
            public void mouseWheelMoved(MouseWheelEvent e) {
                System.out.println("Mouse wheel moved");
            }
        });

        // add listeners to the JTextFields
        myURLDisplay.addTextChangedListener(new TextWatcher() {
            public void textChanged(ChangeEvent e) {
                System.out.println("Text changed");
            }
        });

        // add listeners to the JButtons
        myBackButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Back button clicked");
            }
        });
        myNextButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Next button clicked");
            }
        });
        myGoButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Go button clicked");
            }
        });
        myHomeButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Home button clicked");
            }
        });

        // add listeners to the JMenuItems
        myHomeMenuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Home menu item clicked");
            }
        });
        myNextMenuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Next menu item clicked");
            }
        });
        myBackMenuItem.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Back menu item clicked");
            }
        });

        // add listeners to the JMenus
        fileMenu.add(new JMenuItem("File");
        fileMenu.add(new JMenuItem("Quit");
        fileMenu.add(new JMenuItem("About");
        fileMenu.add(new JMenuItem("Help");

        // set layout
        setLayout(new BorderLayout());
        add(myEditor, BorderLayout.CENTER);
        add(myNextURL, BorderLayout.NORTH);
        add(myURLDisplay, BorderLayout.SOUTH);
        add(myBackButton, BorderLayout.WEST);
        add(myNextButton, BorderLayout.EAST);

        // set size and show
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }

    public static void main(String[] args) {
        Browser browser = new Browser();
        browser.setVisible(true);
    }
}
doNext();
}

// add listener so homeButton calls doHome()
myHomeButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        doHome();
    }
});

// create panel with back,next,home
// flow layout used, alignment set to 0 (far left)
JPanel panel = new JPanel(new FlowLayout(0));
panel.add(myBackButton);
panel.add(myNextButton);
panel.add(myHomeButton);
return panel;

/**
 * Make user-entered URL/text field and back/next/home/go buttons
 * @return returns JPanel containing buttons
 */
private JPanel makeTopPanel() {
    myGoButton = new JButton("Go",new ImageIcon("gosmall.gif");
    myGoButton.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            showPage(myURLDisplay.getText());
        }
    });
    myURLDisplay = new JTextField(35);
    JLabel topLabel = new JLabel("Address");
    JPanel urlPanel = new JPanel(new BorderLayout());
    urlPanel.add(topLabel, BorderLayout.WEST);
    urlPanel.add(myURLDisplay, BorderLayout.CENTER);
    urlPanel.add(myGoButton, BorderLayout.EAST);
    urlPanel.add(makeButtons(), BorderLayout.NORTH);
    return urlPanel;
}

/**
 * Make the panel where "would-be" clicked URL is displayed
 * and the
 * @return returns a JPanel containing the HTML window and the
 * panel at the bottom that
 * lists the URL of a rolled-over link
 */
private JPanel makeBottomPanel() {
    myNextURL = new JLabel();
    JLabel bottomLabel = new JLabel(" ");
    JPanel bottomPanel = new JPanel(new BorderLayout());
    bottomPanel.add(bottomLabel, BorderLayout.WEST);
    bottomPanel.add(myNextURL, BorderLayout.CENTER);
    return bottomPanel;
}

/**
 * Displays homepage.
 */
protected void doHome() {
    if (!myHome.equals("")){
        showPage(myHome);
    }
}

/**
 * Displays the url the user enters and echo
 * the url in the textfield (even if it came from there!)
 * the echo makes it so link-clicking shows the URL
 */
private void showPage(String url) {
    try {
        myEditor.setPage(url);
        myURLDisplay.setText(url);
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null,"could not load "+url);
    }
}

/**
 * Show where link would take us if clicked
 * @param s URL of link
 */
private void showNextURL(String s) {
    myNextURL.setText(s);
}

/**
* Inner class to deal with link-clicks and mouse-overs
*/
private class LinkFollower implements HyperlinkListener {
    public void hyperlinkUpdate(HyperlinkEvent evt) {
        if (evt.getEventType() ==HyperlinkEvent.EventType.ACTIVATED) {
            // user clicked a link, load it and show it
            URL url = null;
            try {
                url = evt.getURL();
                showPage(url.toString());
            } catch (Exception e) {
                String s = evt.getURL().toString();
                JOptionPane.showMessageDialog(Browser.this,
                    "loading problem for " + s + "
                    "+ e,""Load Problem",
                    JOptionPane.ERROR_MESSAGE);
            }
        } else if (evt.getEventType() == HyperlinkEvent.EventType.ENTERED) {
            // user moused-into a link, show what would load
            try {
                showNextURL(evt.getURL().toString());
            } catch (Exception e) {
                // nothing to do, if URL fails, don’t pre-announce
            }
        } else if (evt.getEventType() == HyperlinkEvent.EventType.EXITED) {
            // user moused-out of a link, erase what was shown
            showNextURL("*");
        }
    }
}

/** opens a new browser window */
public static void main(String args[]) {
    Browser b = new Browser();
b.setSize(600, 600);        b.setLocation(10, 20);        b.setVisible(true);
}
import java.util.Random;

/**
 * Class for simulating a die
 * (an object "rolled" to generate a random number)
 * @author Owen Astrachan
 */
public class Dice {
    private int myRollCount; // # times rolled
    private int mySides; // # sides
    private Random myRandGen; // the random number generator

    /**
     * Construct standard six−sided Dice object
     */
    public Dice() {
        this(6);
    }

    /**
     * Constructs a Dice object.
     * @param sides specifies the number of "sides" for the die, e.g.,
     * 2 is a coin, 6 is a "regular" die.
     */
    public Dice(int sides) {
        myRollCount = 0;
        mySides = sides;
        myRandGen = new Random();
    }

    /**
     * Returns the random "roll" of the die, a uniformly distributed
     * random number in the range [1..numSides()].
     * @return a random value in range [1..numSides()]
     */
    public int roll() {
        myRollCount++;
        return myRandGen.nextInt(mySides) + 1;
    }

    /**
     * Returns number of sides of die
     * @return number of sides of this dice object
     */
    public int numSides() {
        return mySides;
    }

    /**
     * Returns number of times this die object has been rolled.
     * @return number of times die object rolled
     */
    public int numRolls() {
        return myRollCount;
    }

    /**
     * Returns a string representing this Dice object
     */
    public String toString() {
        return "# sides: " + numSides() + " # rolls: " + numRolls();
    }

    public static void main(String[] args) {
        Dice die = new Dice(6);
        for (int k = 0; k < 100; k++) {
            int roll = die.roll();
            System.out.println("roll " + die.numRolls() + ": " + roll);
        }
    }
}
/**
 * Simple example of arrays.
 * @author Owen Astrachan
 */

public class DiceAnalyzer
{
    private int[] myCounts;

    /**
     * Construct an analyzer for two six−sided Dice
     */
    public DiceAnalyzer()
    {
        myCounts = new int[13];
    }

    /**
     * Clear all recorded statistics.
     */
    public void clear()
    {
        for (int k=0; k < myCounts.length; k++) {
            myCounts[k] = 0;
        }
        // java.util.Arrays.fill(myCounts,0);
    }

    /**
     * Rolls dice specified number of times, records results.
     * @param rollCount is the number of times to roll the dice
     */
    public void roll(int rollCount)
    {
        Dice d = new Dice(6);
        for (int k=0; k < rollCount; k++) {
            myCounts[d.roll()+d.roll()]++;
        }
    }

    /**
     * Prints a report of recorded statistics of rolls
     */
    public void report()
    {
        System.out.println("−−−");
        for (int k=2; k <= 12; k++) {
            System.out.println(k + "\t" + myCounts[k]);
        }
        System.out.println("−−−");
    }

    public static void main(String[] args)
    {
        int count = 1000;
        if (args.length > 0) {
            count = Integer.parseInt(args[0]);
        }
        DiceAnalyzer da = new DiceAnalyzer();
        da.roll(count);
        da.report();
        da.clear();
        da.roll(count);
        da.report();
    }
}
import java.io.*;
import javax.swing.JFileChooser;
import java.util.StringTokenizer;

/**  * Shows the differences between reading from a file with  * EasyReader and reading from a file with standard Java.  * These examples read words and count the number of words read.  *  * @author Owen Astrachan */

public class WordCounter
{
    /*  * Return number of words in file.  * @param filename is name of file to read (full path/accessible)  * @return number of words in file  */
    public int easyCount(String filename)
    {
        EasyReader input = new EasyReader(filename);
        int count = 0;
        String word;
        while ((word = input.readWord()) != null)
        {   count++;
            return count;
        }
    }

    /*  * Return number of words in file.  * @param filename is name of file to read (full path/accessible)  * @return number of words in file  */
    public int standardCount(String filename) throws IOException
    {
        BufferedReader input = new BufferedReader(
                new FileReader(filename));
        String line,word;
        while ((line = input.readLine()) != null)
        {   StringTokenizer tokens = new StringTokenizer(line);
            while (tokens.hasMoreTokens())
            {   count++; word = tokens.nextToken();
            }
            return count;
        }
    }

    /*  * Pops up File−choosing dialog and calls word−reading method  * for file chosen.  */
    public void readAll() throws IOException
    {
        JFileChooser chooser = new JFileChooser(".");
        String filename = null;
        int chooseVal = chooser.showOpenDialog(null);
        if (chooseVal == JFileChooser.APPROVE_OPTION)
        {   filename = chooser.getSelectedFile().getPath();
        }
        else { System.exit(0); }

        // time and read with EasyReader
        long start = System.currentTimeMillis();
        System.out.println("easy count " + easyCount(filename));
        long end = System.currentTimeMillis();
        System.out.println("time = " + (end−start)/1000.0 + " secs");

        // time and read with standard Java
        start = System.currentTimeMillis();
        System.out.println("standard count " + standardCount(filename));
        end = System.currentTimeMillis();
        System.out.println("time = " + (end−start)/1000.0 + " secs");
    }

    public static void main(String args[]) throws IOException
    {
        WordCounter wc = new WordCounter();
        wc.readAll();
        System.exit(0);
    }
}