```
/*
* ----------------------------------------------------------------------------

* "THE BEER-WARE LICENSE" (Revision 42):
* <doktor@dyregod.dk> wrote this file. As long as you retain this notice you
* can do whatever you want with this stuff. If we meet some day, and you
think
* this stuff is worth it, you can buy me a beer in return. Ulf Holm Nielsen
* ----------------------------------------------------------------------------

*
*/

package dk.dyregod.linda;

import org.eclipse.swt.*;
import org.eclipse.swt.custom.*;
import org.eclipse.swt.dnd.*;
import org.eclipse.swt.events.*;
import org.eclipse.swt.graphics.*;
import org.eclipse.swt.layout.*;
import org.eclipse.swt.program.*;
import org.eclipse.swt.widgets.*;

import org.eclipse.swt.widgets.List;
import java.util.*;
import java.io.*;
import java.text.*;

import dk.dyregod.linda.util.*;

public class Linda
{
    private static final int CPU_WIDTH = 200;
    private static final int CPU_HEIGHT = 200;
    private static final int BUS_WIDTH = 300;
    private static final int BUS_HEIGHT = 250;
    private static final int INPUT_WIDTH = 75;
    private static final int INPUT_HEIGHT = 75;
    private static final int OUTPUT_WIDTH = 75;
    private static final int OUTPUT_HEIGHT = 75;

    private static final int STORAGE_WIDTH = 180;

    private static Display display;
    private static Shell shell;

    private Font font;

    private StyledText editor;
    private Table table;
    private TableEditor tableEditor;
    private Group mem;
    private Group egroup;

    private static ResourceBundle resourceBundle;

    private Maskine maskine = new Maskine(this);
    DecimalFormat form = new DecimalFormat("####");
    DecimalFormat form2 = new DecimalFormat("##");

    Text xText;
    Text yText;
    Text aText;
```

```
    Text pText;
    Text iText;
    Text adrText;
    Text ldrText;
    Text larText;
    Text ioText;

    Label status;
    String statusStr = "Klar";

    private static String splashCmd = null;

    int type = 0;
    boolean go = true;

    Thread thread;

    static String getResourceString(String key)
    {
        try
        {
            return resourceBundle.getString(key);
        }
        catch (MissingResourceException e)
        {
            return key;
        }
        catch (NullPointerException e)
        {
            return "!" + key + "!";
        }
    }

    void fill()
    {
        shell.setText(getResourceString("title"));
        shell.setImage(Images.getImage("general.icons.main"));

        shell.setMenuBar(createMenu());

        GridLayout gridLayout = new GridLayout();
        gridLayout.numColumns = 3;
        gridLayout.marginWidth = 0;
        gridLayout.marginHeight = 0;

        shell.setLayout(gridLayout);

        doCPU();
        /*
        Composite comp2 = new Composite(shell, SWT.BORDER);
        comp2.setLayout(new FillLayout());
        comp2.setLayoutData(new GridData(GridData.FILL_VERTICAL));
        */
        doIO();
        doMemory();



        Composite comp2 = new Composite(shell, SWT.NONE);
        comp2.setLayout(new FillLayout());
        GridData dd = new GridData(GridData.HORIZONTAL_ALIGN_CENTER |
GridData.GRAB_HORIZONTAL);
        dd.horizontalSpan = 2;
```

```
        comp2.setLayoutData(dd);
        Button input = new Button(comp2, SWT.BORDER);
        //input.setText(getResourceString("button.run.text"));
        input.setImage(Images.getImage("button.icon.run"));
        input.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                run();
            }
        });
        input = new Button(comp2, SWT.BORDER);
        //input.setText(getResourceString("button.singlestep.text"));
        input.setImage(Images.getImage("button.icon.singlestep"));
        input.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                singlestep();
            }
        });
        input = new Button(comp2, SWT.BORDER);
        //input.setText(getResourceString("button.mikrostep.text"));
        input.setImage(Images.getImage("button.icon.mikrostep"));
        input.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                mikrostep();
            }
        });
        input = new Button(comp2, SWT.BORDER);
        //input.setText(getResourceString("button.stop.text"));
        input.setImage(Images.getImage("button.icon.stop"));
        input.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                stop();
            }
        });
        input = new Button(comp2, SWT.BORDER);
        //input.setText(getResourceString("button.reset.text"));
        input.setImage(Images.getImage("button.icon.reset"));
        input.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                reset();
            }
        });
        status = new Label(shell, SWT.BORDER);
        GridData dat =
            new GridData(GridData.FILL_HORIZONTAL | GridData.HORIZONTAL_ALIGN_
FILL);
        dat.horizontalSpan = 3;
        status.setLayoutData(dat);

    }

    void open()
    {
        shell.setSize(640, 480);
        shell.setLocation(
```

```
            (display.getClientArea().width - shell.getSize().x) / 2,
            (display.getClientArea().height - shell.getSize().y) / 2);
        shell.open();

        update();
        drive();

        // Cleanup
        //workerStop();
        Images.free();
        display.dispose();
    }
    void init()
    {
        // Create the wmain dtb windows
        form.setMinimumIntegerDigits(4);
        form2.setMinimumIntegerDigits(2);

        display = new Display();
        Images.init(display);
        shell = new Shell();

        fill();
    }
    void doIO()
    {
        Group comp3 = new Group(shell, SWT.NONE);
        comp3.setText(getResourceString("io.title.text"));
        GridLayout gLayout = new GridLayout();
        gLayout.numColumns = 2;
        gLayout.marginWidth = 3;
        gLayout.marginHeight = 3;
        comp3.setLayout(gLayout);

        GridData dd1 = new GridData(GridData.HORIZONTAL_ALIGN_BEGINNING |
GridData.VERTICAL_ALIGN_BEGINNING);
        dd1.horizontalSpan = 1;
        comp3.setLayoutData(dd1);


        Label lbl = new Label(comp3, SWT.NONE);
        lbl.setText(getResourceString("io.title.text"));
        ioText = new Text(comp3, SWT.BORDER | SWT.CENTER);
        ioText.setEditable(false);
        Button input = new Button(comp3, SWT.BORDER | SWT.CENTER);
        GridData dub = new GridData(GridData.FILL_HORIZONTAL);
        dub.horizontalSpan =2;
        input.setLayoutData(dub);
        input.setImage(Images.getImage("button.icon.input"));

        input.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                ioText.setEditable(false);
                maskine.registers[maskine.IO] = Integer.valueOf(ioText.
getText()).intValue();
                if(thread==null)
                    return;
                if (thread.isAlive())
                {

                    synchronized (this)
                    {
```

```
                    maskine.wait = false;
                    notify();
                }
            }
        }
    });
}
void doCPU()
{
    Group cpu = new Group(shell, SWT.NULL);
    GridData data = new GridData(GridData.VERTICAL_ALIGN_BEGINNING);
    data.horizontalSpan = 1;
    cpu.setLayoutData(data);
    cpu.setText(getResourceString("cpu.title.text"));

    GridLayout gLayout = new GridLayout();
    gLayout.numColumns = 2;
    gLayout.marginWidth = 3;
    gLayout.marginHeight = 3;
    cpu.setLayout(gLayout);

    Label xLbl = new Label(cpu, SWT.NONE);
    xText = new Text(cpu, SWT.BORDER | SWT.CENTER);
    xText.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
    xLbl.setText(getResourceString("cpu.x.text"));

    Label yLbl = new Label(cpu, SWT.NONE);
    yText = new Text(cpu, SWT.BORDER | SWT.CENTER);
    yText.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
    yLbl.setText(getResourceString("cpu.y.text"));

    Label aLbl = new Label(cpu, SWT.NONE);
    aText = new Text(cpu, SWT.BORDER | SWT.CENTER);
    aText.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
    aLbl.setText(getResourceString("cpu.a.text"));

    Label pLbl = new Label(cpu, SWT.NONE);
    pText = new Text(cpu, SWT.BORDER | SWT.CENTER);
    pText.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
    pLbl.setText(getResourceString("cpu.p.text"));

    Label iLbl = new Label(cpu, SWT.NONE);
    iText = new Text(cpu, SWT.BORDER | SWT.CENTER);
    iText.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
    iLbl.setText(getResourceString("cpu.i.text"));

    Label adrLbl = new Label(cpu, SWT.NONE);
    adrText = new Text(cpu, SWT.BORDER | SWT.CENTER);
    adrText.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
    adrLbl.setText(getResourceString("cpu.adr.text"));

}
void doMemory()
{
    // Setup the memory group

    mem = new Group(shell, SWT.NULL);
    GridData grid =
        new GridData(
            GridData.FILL_VERTICAL
                | GridData.HORIZONTAL_ALIGN_END
                | GridData.VERTICAL_ALIGN_FILL);
    grid.verticalSpan = 2;
    mem.setLayoutData(grid);
```

```
        mem.setText(getResourceString("mem.title.text"));

        GridLayout gridLayout = new GridLayout();
        gridLayout.numColumns = 2;
        gridLayout.marginWidth = 3;
        gridLayout.marginHeight = 3;
        mem.setLayout(gridLayout);

        Label ldrLbl = new Label(mem, SWT.NONE);
        ldrText = new Text(mem, SWT.BORDER | SWT.CENTER);
        ldrText.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
        ldrLbl.setText(getResourceString("mem.ldr.text"));

        Label larLbl = new Label(mem, SWT.NONE);
        larText = new Text(mem, SWT.BORDER | SWT.CENTER);
        larText.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
        larLbl.setText(getResourceString("mem.lar.text"));

        table = new Table(mem, SWT.BORDER | SWT.FULL_SELECTION | SWT.SINGLE);
        GridData gridData = new GridData();
        gridData.verticalAlignment = GridData.FILL;
        gridData.horizontalSpan = 2;
        gridData.grabExcessVerticalSpace = true;
        gridData.horizontalAlignment = GridData.FILL;
        gridData.grabExcessHorizontalSpace = true;
        //gridData.verticalAlignment = GridData.GRAB_VERTICAL;
        table.setLayoutData(gridData);
        table.setHeaderVisible(true);
        table.setLinesVisible(true);

        table.setMenu(createPopUpMenu());
        tableEditor = new TableEditor(table);

        table.addSelectionListener(new SelectionAdapter()
        {
            public void widgetDefaultSelected(SelectionEvent e)
            {
                editTableCell();
            }
        });

        TableColumn column = new TableColumn(table, SWT.CENTER);
        column.setText(getResourceString("mem.add.text"));
        column.setWidth(50);
        column = new TableColumn(table, SWT.NONE);
        column.setText(getResourceString("mem.value.text"));
        column.setWidth(80);

        for (int i = 0; i < 100; i++)
        {
            TableItem item = new TableItem(table, SWT.CENTER);
            item.setText(new String[] { form2.format(i), form.format(maskine.
lager[i])});
        }
    }
    void editTableCell()
    {
        // Clean up any previous editor control
        Control oldEditor = tableEditor.getEditor();
        if (oldEditor != null)
            oldEditor.dispose();

        // Identify the selected row
```

```
    int index = table.getSelectionIndex();
    if (index == -1)
        return;
    final TableItem item = table.getItem(index);

    // The control that will be the editor must be a child of the Table
    final Text text = new Text(table, SWT.NONE);
    text.addFocusListener(new FocusAdapter()
    {
        public void focusLost(FocusEvent e)
        {
            try
            {

                int i = Integer.parseInt(text.getText());
                if (i < 10000)
                    maskine.lager[table.getSelectionIndex()] = i;
            }
            catch (NumberFormatException nfe)
            {
            }
            text.dispose();
            update();
        }
    });

    //The text editor must have the same size as the cell and must
    //not be any smaller than 50 pixels.
    tableEditor.horizontalAlignment = SWT.LEFT;
    tableEditor.grabHorizontal = true;
    tableEditor.minimumWidth = 50;

    // Open the text editor in the second column of the selected row.
    tableEditor.setEditor(text, item, 1);

    // Assign focus to the text control
    text.setFocus();

}

void close()
{
    Images.free();
    display.dispose();
}

private Menu createMenu()
{
    Menu menuBar = new Menu(shell, SWT.BAR);
    shell.setMenuBar(menuBar);

    //create each header and subMenu for the menuBar
    createFileMenu(menuBar);
    createRunMenu(menuBar);
    //createEditMenu(menuBar);
    //createSearchMenu(menuBar);
    createHelpMenu(menuBar);

    return menuBar;
}
private void createRunMenu(Menu menuBar)
{
    MenuItem item = new MenuItem(menuBar, SWT.CASCADE);
    item.setText(getResourceString("menu.run.title"));
```

```
        Menu menu = new Menu(shell, SWT.DROP_DOWN);
        item.setMenu(menu);

        MenuItem subItem = new MenuItem(menu, SWT.NULL);
        subItem.setText(getResourceString("menu.run.run.title"));
        subItem.setAccelerator(SWT.CTRL + 'G');
        subItem.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                run();
            }
        });

        subItem = new MenuItem(menu, SWT.NULL);
        subItem.setText(getResourceString("menu.run.singlestep.title"));
        subItem.setAccelerator(SWT.CTRL + 'T');
        subItem.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                singlestep();
            }
        });
        subItem = new MenuItem(menu, SWT.NULL);
        subItem.setText(getResourceString("menu.run.mikrostep.title"));
        subItem.setAccelerator(SWT.CTRL + 'M');
        subItem.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                mikrostep();
            }
        });

        subItem = new MenuItem(menu, SWT.NULL);
        subItem.setText(getResourceString("menu.run.stop.title"));
        subItem.setAccelerator(SWT.CTRL + 'H');
        subItem.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                stop();
            }
        });
        item = new MenuItem(menu, SWT.SEPARATOR);
        subItem = new MenuItem(menu, SWT.NULL);
        subItem.setText(getResourceString("menu.run.reset.title"));
        subItem.setAccelerator(SWT.CTRL + 'R');
        subItem.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {

                reset();
            }
        });
    }
    private void createHelpMenu(Menu menuBar)
    {
        MenuItem item = new MenuItem(menuBar, SWT.CASCADE);
        item.setText(getResourceString("menu.help.title"));
        Menu menu = new Menu(shell, SWT.DROP_DOWN);
        item.setMenu(menu);
```

```
        //Help -> About Text Editor
        MenuItem subItem = new MenuItem(menu, SWT.NULL);
        subItem.setText(getResourceString("menu.help.about.title"));
        subItem.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                MessageBox box = new MessageBox(shell, SWT.ICON_INFORMATION |
SWT.OK);
                box.setText(getResourceString("menu.help.about.box.title"));
                box.setMessage(getResourceString("menu.help.about.box.text"));
                box.open();
            }
        });
    }
    private Menu createPopUpMenu()
    {
        Menu popUpMenu = new Menu(shell, SWT.POP_UP);

        /**
         * Adds a listener to handle enabling and disabling
         * some items in the Edit submenu.
         */
        popUpMenu.addMenuListener(new MenuAdapter()
        {
            public void menuShown(MenuEvent e)
            {
                Menu menu = (Menu) e.widget;
            }
        });

        //Edit
        MenuItem item = new MenuItem(popUpMenu, SWT.CASCADE);
        item.setText(getResourceString("popup.edit"));
        item.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                editTableCell();
            }
        });

        new MenuItem(popUpMenu, SWT.SEPARATOR);

        //Clear this
        item = new MenuItem(popUpMenu, SWT.CASCADE);
        item.setText(getResourceString("popup.clear"));
        item.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                maskine.lager[table.getSelectionIndex()] = 0;
                update();
            }
        });

        //Clear all
        item = new MenuItem(popUpMenu, SWT.CASCADE);
        item.setText(getResourceString("popup.clearall"));
        item.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
```

```
            clearAll();
        }
    });

    return popUpMenu;
}
void clearAll()
{
    for (int i = 0; i < maskine.lager.length; i++)
    {
        maskine.lager[i] = 0;
    }
    for (int i = 0; i < maskine.registers.length; i++)
    {
        maskine.registers[i] = 0;
    }
    update();
}
void run()
{
    maskine.type = 0;
    thread = new Thread(maskine);
    thread.setPriority(Thread.MIN_PRIORITY);
    thread.start();
}
void singlestep()
{
    if (thread == null)
    {
        thread = new Thread(maskine);
    }
    maskine.type = 1;
    if (thread.isAlive())
    {

        synchronized (this)
        {
            maskine.halt = false;
            notify();
        }
    }
    else
    {
        thread = new Thread(maskine);
        thread.setPriority(Thread.MIN_PRIORITY);
        thread.start();
    }
}
void mikrostep()
{
    if (thread == null)
    {
        thread = new Thread(maskine);
    }
    maskine.type = 2;
    if (thread.isAlive())
    {

        synchronized (this)
        {
            maskine.halt = false;
            notify();
        }
    }
```

```
        else
        {
            thread = new Thread(maskine);
            thread.setPriority(Thread.MIN_PRIORITY);
            thread.start();
        }
    }
    void stop()
    {
        maskine.stop = true;
    }
    void reset()
    {
        if (thread == null)
            return;
        if (thread.isAlive())
        {

            synchronized (this)
            {
                maskine.type = 0;
                maskine.stop = true;
                maskine.halt = false;
                notify();
            }
        }
        thread = null;
        maskine.reset();
        update();
    }
    void saveFile()
    {
        final String textString;
        FileDialog fileDialog = new FileDialog(shell, SWT.SAVE);

        fileDialog.setFilterExtensions(new String[] { "*.ls", "*.*" });
        fileDialog.open();
        String name = fileDialog.getFileName();

        if ((name == null) || (name.length() == 0))
            return;

        File file = new File(fileDialog.getFilterPath(), name);
        if (file.exists())
        {
            String message =
                MessageFormat.format(
                    getResourceString("warning.file.exist"),
                    new String[] { file.getName()});
            if (!YesNoMessage.display(new Shell(), message, getResourceString(
"warning")))
                return;
        }

        try
        {
            FileOutputStream stream = new FileOutputStream(file.getPath()+".
ls");
            try
            {
                Writer out = new BufferedWriter(new OutputStreamWriter(stream)
);
                for (int i = 0; i < maskine.lager.length; i++)
                {
```

```
                    out.write(String.valueOf(maskine.lager[i]));
                    out.write("\n");
                    out.flush();
                }
                stream.close();
            }
            catch (IOException e)
            {
                String message =
                    MessageFormat.format(
                        getResourceString("error.file.ioerror"),
                        new String[] { file.getName()});
                ErrorMessage.display(new Shell(), getResourceString("error"),
message);
                return;
            }
        }
        catch (FileNotFoundException e)
        {
            String message =
                MessageFormat.format(
                    getResourceString("error.file.notfound"),
                    new String[] { file.getName()});
            ErrorMessage.display(new Shell(), getResourceString("error"),
message);
            return;
        }
    }
    void openFile()
    {
        final String textString;
        FileDialog fileDialog = new FileDialog(shell, SWT.OPEN);

        fileDialog.setFilterExtensions(new String[] { "*.ls", "*.*" });
        fileDialog.open();
        String name = fileDialog.getFileName();

        if ((name == null) || (name.length() == 0))
            return;

        File file = new File(fileDialog.getFilterPath(), name);
        if (!file.exists())
        {
            String message =
                MessageFormat.format(
                    getResourceString("error.file.noexist"),
                    new String[] { file.getName()});
            ErrorMessage.display(new Shell(), message, getResourceString("
error"));
            return;
        }

        try
        {
            FileInputStream stream = new FileInputStream(file.getPath());
            try
            {
                BufferedReader in = new BufferedReader(new InputStreamReader(
stream));
                String st;
                int i = 0;
                while (i < 100)
                {
                    st = in.readLine();
```

```
                    maskine.lager[i] = Integer.parseInt(st);
                    i++;
                }
                stream.close();
                update();
            }
            catch (IOException e)
            {
                String message =
                    MessageFormat.format(
                        getResourceString("error.file.ioerror"),
                        new String[] { file.getName()});
                ErrorMessage.display(new Shell(), getResourceString("error"),
message);
                return;
            }
        }
        catch (FileNotFoundException e)
        {
            String message =
                MessageFormat.format(
                    getResourceString("error.file.notfound"),
                    new String[] { file.getName()});
            ErrorMessage.display(new Shell(), getResourceString("error"),
message);
            return;
        }

    }
    private void createFileMenu(Menu parent)
    {
        Menu menu = new Menu(parent);
        MenuItem header = new MenuItem(parent, SWT.CASCADE);
        header.setText(getResourceString("menu.file.title"));
        header.setMenu(menu);

        MenuItem item;

        item = new MenuItem(menu, SWT.CASCADE);
        item.setText(getResourceString("menu.file.open.title"));
        item.setAccelerator(SWT.CTRL + 'O');
        item.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent event)
            {
                openFile();
            }
        });
        item = new MenuItem(menu, SWT.PUSH);
        item.setText(getResourceString("menu.file.save.title"));
        item.setAccelerator(SWT.CTRL + 'S');
        item.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
            {
                saveFile();
            }
        });
        item = new MenuItem(menu, SWT.SEPARATOR);
        item = new MenuItem(menu, SWT.PUSH);
        item.setText(getResourceString("menu.file.exit.title"));
        item.addSelectionListener(new SelectionAdapter()
        {
            public void widgetSelected(SelectionEvent e)
```

```
            {
                close();
            }
        });
    }

    public void update()
    {

        shell.getDisplay().syncExec(new Runnable()
        {
            public void run()
            {
                TableItem[] items = table.getItems();
                for (int i = 0; i < items.length; i++)
                {
                    items[i].setText(
                        new String[] { form2.format(i), form.format(maskine.
lager[i])});
                }
                larText.setText(form.format(maskine.registers[maskine.LAR]));
                ldrText.setText(form.format(maskine.registers[maskine.LDR]));
                xText.setText(form.format(maskine.registers[maskine.X]));
                yText.setText(form.format(maskine.registers[maskine.Y]));
                aText.setText(form.format(maskine.registers[maskine.A]));
                pText.setText(form2.format(maskine.registers[maskine.P]));
                iText.setText(form2.format(maskine.registers[maskine.I] / 100)
);
                adrText.setText(form2.format(maskine.registers[maskine.ADR]));
                ioText.setText(form.format(maskine.registers[maskine.IO]));

                status.setText(statusStr);

                shell.redraw();

            }
        });

    }
    public Shell getShell()
    {
        return shell;
    }
    void drive()
    {
        // Event loop
        while (!shell.isDisposed())
        {
            if (!display.readAndDispatch())
                display.sleep();
        }
    }

    public static void main(String[] args)
    {
        System.out.println("LINDA running on:");
        System.out.println(
            System.getProperty("java.vm.vendor")
                + " "
                + System.getProperty("java.version")
                + " "
                + System.getProperty("java.vm.name"));
        System.out.println(
            System.getProperty("os.name")
```

```
                + " "
                + System.getProperty("os.version")
                + " "
                + System.getProperty("os.arch"));
        System.out.println("Classpath = " + System.getProperty("java.class.
path"));
        System.out.println("Ext dirs = " + System.getProperty("java.ext.dirs")
);
        System.out.println(System.getProperty("java.class.version") + "\n");

        System.out.println(System.getProperty("user.name"));
        System.out.println(System.getProperty("user.home"));
        System.out.println(System.getProperty("user.dir"));

        System.out.println("\nstarting..");

        // Extract the command to end the splash window.
        for (int index = 0; index < args.length; index++)
        {
            System.out.println("    args[" + index + "] = '" + args[index] + "
'");
            if (args[index].equals("-endsplash") && (index + 1) < args.length)
            {
                splashCmd = args[index + 1];
            }
        }

        resourceBundle = ResourceBundle.getBundle("LindaResource");
        Linda linda = new Linda();
        linda.init();
        try
        {
            Thread.sleep(1000);
        }
        catch (Exception e)
        {
        }
        System.out.println("Initialization complete!");

        // Initialization is complete so execute the end splash command.
        if (splashCmd != null)
        {
            try
            {
                Runtime.getRuntime().exec(splashCmd);
            }
            catch (Exception e)
            {
            }
        }

        linda.open();

        try
        {
            Thread.sleep(2000);
        }
        catch (Exception e)
        {
        }
        System.out.println("Program is terminating!");

    }
}
```