

Gareth Evans

Assignment 2: “Xmas Work”

“Take a copy of the `tstComponents.java` file from the `E1027labs` directory and produce an analysis of its use of the 'awt' functions. Then using this information redesign it to provide the 'normal' browser frame for Windows95 Explorer.”

Section A: An analysis of the use of the AWT functions in the “tstcomponents.java” file

First of all, let us take a look at the code we are concerned with.

```
import java.awt.Frame;
import java.awt.Button;
import java.awt.Label;
import java.awt.Checkbox;
import java.awt.List;
import java.awt.FlowLayout;
import java.awt.BorderLayout;
import java.awt.Panel;
import java.awt.Choice;
import java.awt.Canvas;
import java.awt.Color;
import java.awt.Event;
import java.awt.TextArea;
import java.awt.TextField;
import java.awt.Scrollbar;

class componentHolder extends Frame {
    componentHolder()
    {
        super("AWT Basic Components");
        setLayout(new BorderLayout());

        Panel topP = new Panel();
        topP.setLayout(new FlowLayout(FlowLayout.LEFT));
        topP.add(new Button("my Button"));
        topP.add(new Label("my Label"));
        topP.add(new Checkbox("my CheckBox"));
        add("North", topP);

        Panel midP = new Panel();
        midP.setLayout(new FlowLayout(FlowLayout.LEFT));
        List l = null;
        midP.add(l = new List(5,true));
        l.addItem("listItem1");
        l.addItem("listItem2");
        l.addItem("listItem3");
        Choice c = null;
        midP.add(c = new Choice());
        c.addItem("choiceItem1");
        c.addItem("choiceItem2");
        c.addItem("choiceItem3");
        Canvas can;
        midP.add(can = new Canvas());
        can.setBackground(Color.white);
        add("Center", midP);

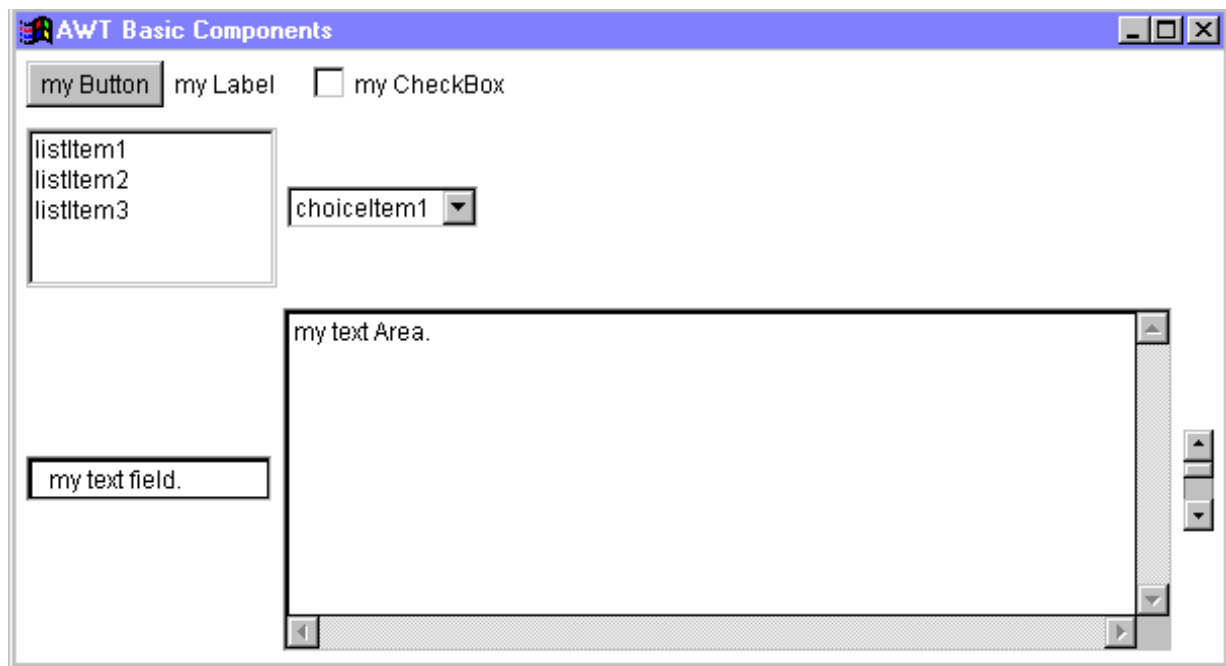
        Panel botP = new Panel();
        botP.setLayout(new FlowLayout(FlowLayout.LEFT));
        botP.add(new TextField("my text field.));
        botP.add(new TextArea("my text Area.));
        botP.add(new Scrollbar(Scrollbar.VERTICAL));
        add("South", botP);

        pack();
        show();
    }

    public boolean handleEvent(Event evt)
    {
        switch (evt.id) {
            case Event.WINDOW_DESTROY:
                System.exit(0);
        }
        return false;
    }
}

class tstComponents {
    public static void main(String args[])
    {
        new componentHolder();
    }
}
```

When we **execute** the code shown on the previous page, we obtain the following running program:

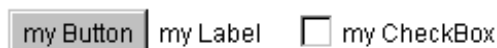


As you can see, there are *several* standard Windows components contained in the program's window, such as a **button**, a **text area**, a **text field** and a **list**. The window has a title (*AWT Basic Components*) and the components are laid out on the page, principally in 3 horizontal sections.

Let us now analyse the code on the previous page that produced the screenshot you see above. First of all, the code imports selected members of the **Abstract Window Toolkit** (AWT). These provide the routines needed to be able to “put” different components on the screen.

Next, we declare a class called a Component Holder (`componentHolder`) which will define all the components seen above. Within this class, we initially set the title of the window and set the initial layout used in the window (Border Layout). We then create the three row sections as three separate panels, called `topP`, `midP` and `botP`.

(1) **topP**



First of all, we say that we want to place every new component created on the first free space we have on the **left**. We do this using the LEFT flow layout command.

```
topP.setLayout(new FlowLayout(FlowLayout.LEFT));
```

We then add a *Button*, a *Label* and a *Check Box*, using the following syntax:

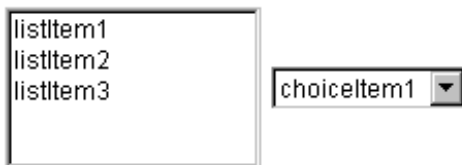
```
topP.add(new COMPONENT_NAME("COMPONENT LABEL"));
```

So, for example to add a new **Button** with the label **Hello!**, we would type in the following code:

```
topP.add(new Button("Hello!"));
```

Finally, we specify where we should place the frame on the screen. Here, we want to place this initial frame at the top of the window, so we use the command `add("North", topP);`

(2) **midP**



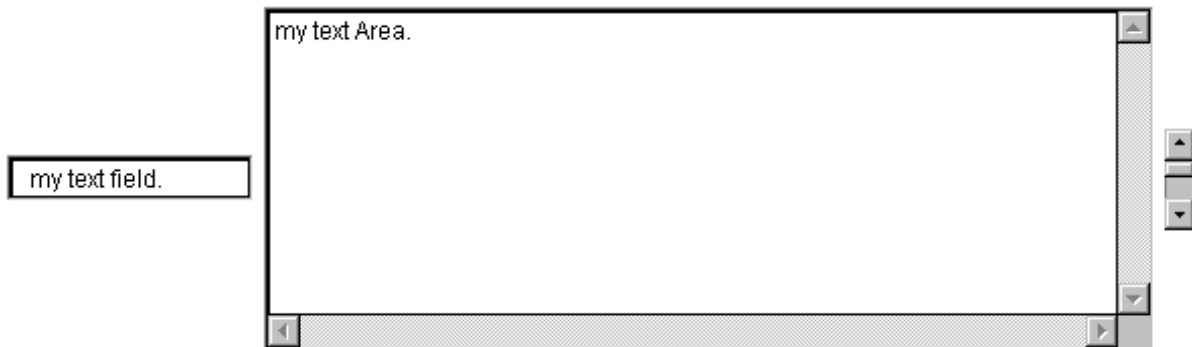
For this panel, we again set the **Left Flow Layout** at the beginning of the section and **add** the frame to the screen at the end of the section (This time adding the frame to the “Middle” of the window). There are two components in this section, namely a *list* and a *Drop-down list* (A “*Choice*” box).

For the former component, we first add it to the frame (Note, in this line of code (`midP.add(l = new List(5,true));`), we are using the **list** command using two parameters. The **first** parameter, “5”, specifies the initial number of rows to be displayed on screen. The **second** parameter, “true”, specifies whether the list can have multiple sections). We then add items to the list, using the “addItem” routine with the list object we have created (called “1”).

For the latter component, the drop-down list, we first ensure that it is empty to begin with (using `Choice c = null;`) and then declare it in the usual manner. We are then free to add items to this list, again using the “addItem” method.

Interestingly, the final part of the midP section declares a **Canvas** with a white background. This doesn’t seem to affect anything, because changing the colour of the canvas doesn’t change the **appearance** of the program.

(3) **botP**



This section is almost identical to the **first** section, topP. It specifies the Left Flow Layout, asks the computer to place it on the screen (This time at the bottom of the screen (South)), and declares some components. Here we declare a *Text Field*, a *Text Area* and a *Scroll bar*. Again, the `topP.add(new COMPONENT_NAME("COMPONENT LABEL"));` syntax is used.

Two lines of code are almost hidden away after all the complicated definitions described above. The Pack command resizes the window so that it fits snugly around all the components declared so far. And the Show command enables us to see the window.

An important aspect of any program is how to handle **user input**, especially with Windows programs where this input can come from many locations. To deal with this, this program declares a small procedure to handle all relevant inputs or events.

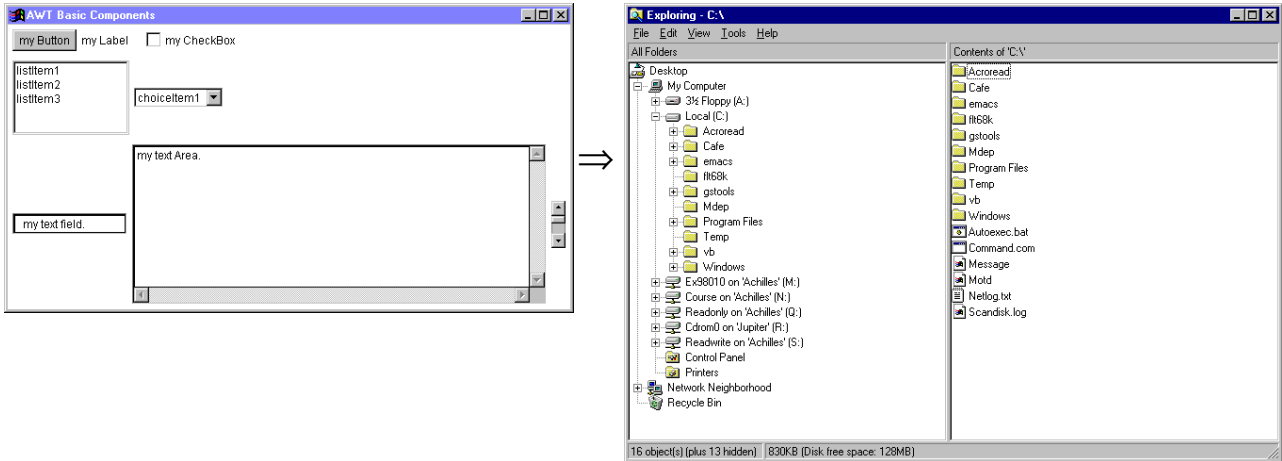
```
public boolean handleEvent(Event evt)
```

This piece of code basically in the occurrence of any event causing the window to be **destroyed**, closes the window as expected (Using the `System.exit(0)` command). Looking at the screenshot of the program, we see that ways the window may be “destroyed” include clicking on the close button on the top right (or double clicking the control menu on the top left, or pressing Alt+F4 etc.). This piece of code **concludes** the Component Holder class.

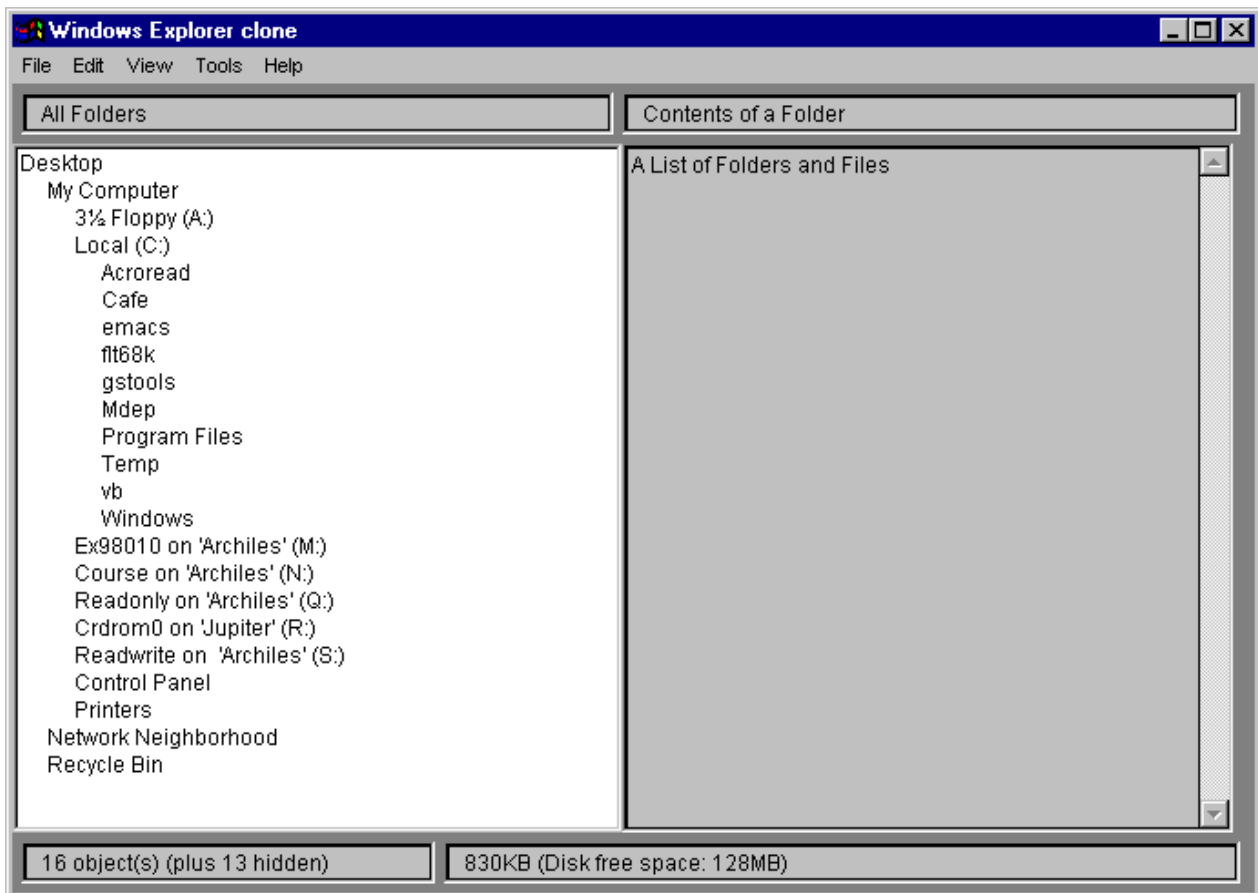
Next, and finally, we have the **main** class of the program, declared in the usual way. The main class is very simple indeed, in that it just declares a new Component Holder Object. This is all that is needed to display the window on the screen. All the code needed to deal with *component building* and *user interaction* is already contained **within** the Component Holder class.

Section B: Producing a Windows Explorer clone using the code given.

The task we have been given is to use the material given in the *tstComponents.java* file and attempt to replicate as accurately as possible the default Windows Explorer Frame.



The following is a screenshot of the “*Windows Explorer*” I created.



On the next pages you will see the **code** used to make the window shown above.

```

/**
 *
 * Windows Explorer Clone
 *
 * Written by: Gareth Evans
 *
 * First Written: 1/Feb/99
 * Last Rewritten: 2/Feb/99
 *
 */

import java.awt.* ;

class componentHolder extends Frame {
    componentHolder()
    {

        // set up Basic Window

        super("Windows Explorer clone");
        setSize(690, 490) ;
        setBackground(Color.gray) ;
        setLayout(new BorderLayout());

        // set up menu system

        Menu menu = new Menu("File") ;
        Menu menu2 = new Menu("Edit");
        Menu menu3 = new Menu("View");
        Menu menu4 = new Menu("Tools");
        Menu menu5 = new Menu("Help");
        MenuBar menuBar = new MenuBar() ;
        menuBar.add(menu) ;
        menuBar.add(menu2);
        menuBar.add(menu3);
        menuBar.add(menu4);
        menuBar.add(menu5);
        setMenuBar(menuBar) ;

        // set up "Windows Explorer" components

        // top section: descriptive text boxes

        Panel topP = new Panel();
        topP.setLayout(new FlowLayout(FlowLayout.LEFT));

        topP.add(new TextField("All Folders" ));
        topP.add(new TextField("Contents of a Folder" ));
        add("North", topP);

        // middle section: system icons and file icons

        Panel midP = new Panel();
        midP.setLayout(new GridLayout(1,2));
        //(Number of rows, Number of columns)

        List l = null;
        midP.add(l = new List(25,true));
        l.addItem("Desktop");
        l.addItem("    My Computer");
        l.addItem("        3½ Floppy (A:)");
        l.addItem("        Local (C:)");
        l.addItem("        Acroread");
        l.addItem("        Cafe");
        l.addItem("        emacs");
        l.addItem("        flt68k");
        l.addItem("        gstools");
        l.addItem("        Mdep");
        l.addItem("        Program Files");
        l.addItem("        Temp");
        l.addItem("        vb");
        l.addItem("        Windows");
        l.addItem("    Ex98010 on 'Archiles' (M:)");
        l.addItem("    Course on 'Archiles' (N:)");
        l.addItem("    Readonly on 'Archiles' (Q:)");
        l.addItem("    Crdrom0 on 'Jupiter' (R:)");
    }
}

```

```

l.addItem("          Readwrite on 'Archives' (S:)");
l.addItem("          Control Panel");
l.addItem("          Printers");
l.addItem("    Network Neighborhood");
l.addItem("    Recycle Bin");

TextArea t = null;
midP.add(t = new TextArea("A List of Folders and Files",23,45,1));
// Syntax: (Number of rows, Number of columns, Scrollbar Visibility)
//                               1 = Vertical Only

Canvas can;
midP.add(can = new Canvas());
can.setBackground(Color.gray);
add("West", midP);

// bottom section: more descriptive text boxes

Panel botP = new Panel();
botP.setLayout(new FlowLayout(FlowLayout.LEFT));

botP.add(new TextField("16 object(s) (plus 13 hidden)"));
botP.add(new TextField("830KB (Disk free space: 128MB)"));
add("South", botP);

show();
}

public boolean handleEvent(Event evt)
{
    switch (evt.id) {
        case Event.WINDOW_DESTROY:
            System.exit(0);
    }
    return false;
}
}

class Explorer {
    public static void main(String args[])
    {
        new componentHolder();
    }
}

// End of Program

```

Summary of **changes** to the `tstComponents.java` file

- (1) A menu system was added to the program, simulating the main menu sections of Windows Explorer. This was done by declaring Menu items and then adding them to a Menu Bar. It would have been possible to program all the submenus etc., but as the assignment specified only the redesign of the frame, I thought it would have been pointless to program all the submenus.
- (2) The **Background** colour was set to grey, and the **size** of the window set, in the initial stages of the Component Holder class.
- (3) The top panel was restructured to display **2 text fields**. The first text field displayed the text “All folders” while the second field showed “Contents of a folder”. The code for implementing the text fields follows the form `topP.add(new TextField("Descriptive Label"));`. Note: To make the boxes the correct size, **spaces** were inserted

- (4) The middle panel was changed so that it showed **two** sections, the first section being a *list* showing all the objects on the “desktop”, and the second section being a text area. The list was declared with **25** initial rows, and labels were inserted into the list to simulate the appearance of several objects on the desktop. The Text Area was sized using the parameters of declaring the component (*see the program listing*).

Note: The middle panel used a Grid Layout so that I was able to size the components as I wanted (either by using **parameters** when declaring components, or by the fact that components behave **differently** under the control of the Grid Layout).

- (5) Finally, the bottom panel was structured similarly to the top panel. Like the top panel, we declared **two** Text Fields, this time one for displaying how many files (hidden & unhidden) were in the current folder; and the other for displaying the size of the current folder and current free disk space. (Note: These were of course *dummy* fields and were not in any way *dynamic*).

The **rest** of the program was identical to the `tstComponents.java` program.