The Tutorials – Your First Steps in the IDE

Introduction

These tutorials will get you down to work with Forte™ for Java™ Community Edition, quickly and easily. We'll walk you through building several simple applications, step by step, to familiarize you with the IDE interface and operation.

Tutorial One: The Clock. Presented in three steps, this first builds a simple, functional clock, making use of the built in TimerBean. Steps two and three then extend the functionality of the clock, adding the ability to set the time, and date format.

Tutorial Two: The Color Picker. In part one of this tutorial, we will use Forte for Java's special bean support to generate a JavaBean with three properties – red, green and blue. In Step 2 we will build a form incorporating this bean and use RGB sliders to set the background color.

Tutorial Three: The Image Viewer. In this tutorial we build a simple image viewer.

Tutorial Four: The Debugger. A brief introduction to Forte for Java's Debugging subsystem.

Tutorial One: The Clock

For this tutorial, we will start with a simple clock form, making use of the built-in TimerBean. Once we have this clock compiled and running successfully, we will then extend its functionality, adding the ability to set the time, and change the time and date format.

Part One

Startup:

1   Click the **New from Template**... icon on the Main Window. The New From Template wizard will open, displaying a tree view of templates grouped into several categories. Expand the **SwingForms** node, select **JFrame**, and click **Next**.
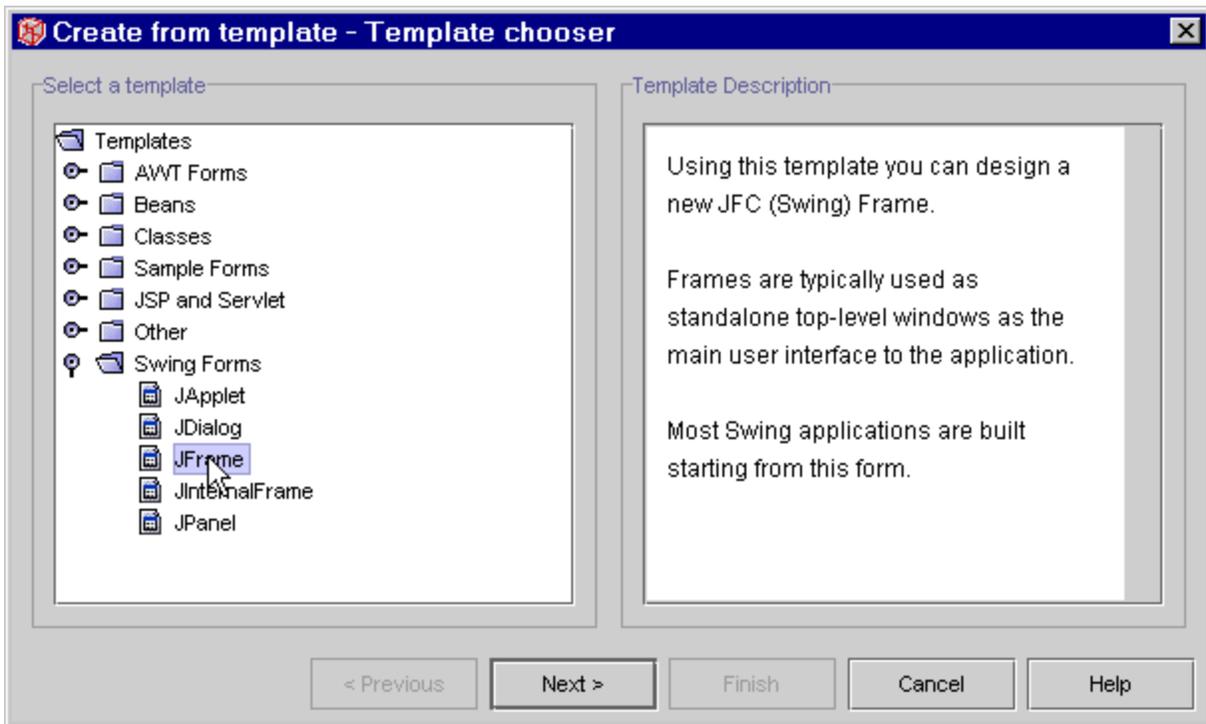
2 The Target Location page of the wizard will appear to prompt you to set a name and package for your new class. The tree view shows all directories that you have mounted as file systems in the Explorer. Expand the directory structure and choose `$FORTE4J_HOME/Development/examples` as the location, (where $FORTE4J_HOME is your Forte For Java installation directory) and type `ClockFrame` in the **Object Name** field. Click **Finish** when done.
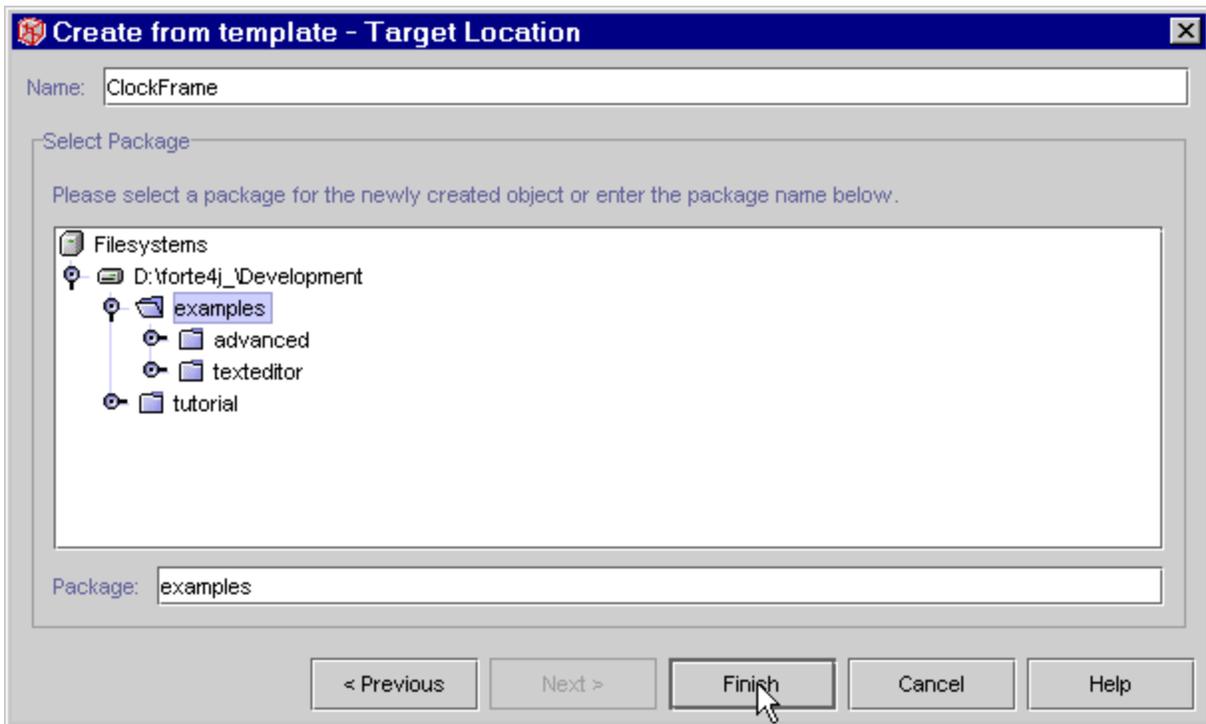
> **Note:** A dialog box may then appear and prompt you whether to make the class part of the current project. For the purposes of these tutorials, it does not matter whether you include them in the project or not.

3 You should see the status line of the Main Window read `Opening Form: ClockFrame`. Several windows will open – the source Editor, the Form Editor window and the Component Inspector. Note there are sections of the source in the Editor which have a colored background – these sections are those regenerated by the Form Editor and may not be modified.

The Component Inspector lists all components currently in the Form Editor and their properties. Initially there are no components except for the default layout (BorderLayout) and a heading for Non-Visible components – currently empty.

Add a component:

1 We will use a standard JLabel for our clock display. Flip to the **Swing** tab of the Component Palette. You will see a grouping of common Swing components. Position your mouse cursor over each icon to see a tooltip identification.

JLabel   JButton
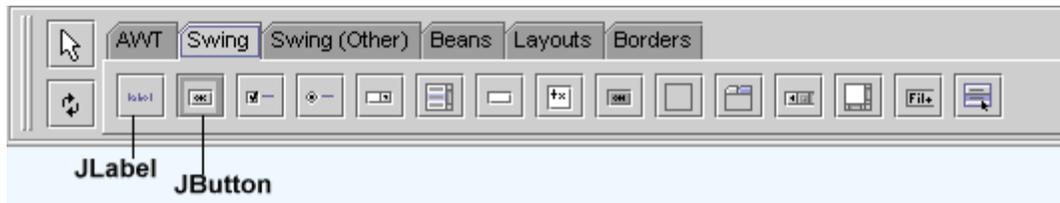
2   Select **JLabel** by single clicking on its icon. The icon will appear `clicked`, indicating it has been selected and is the active component.

3   Place it on the Center panel of the Form Editor surface by clicking once. You will see the generated code appear in the Editor and a new component listed in the Component Inspector. Note the currently selected component is marked with blue corner anchor marks. The item highlighted in the Component Inspector listing also indicates the selected component.

Modify the component's properties:

1   Now we will modify the properties of the `JLabel`. Make sure the JLabel is selected, either by clicking it in the Component Inspector or by clicking it in the Form Editor window. Flip to the **Code Generation** tab of the JLabel in the Component Inspector – you will see `Variable Name` property and its value listed.

2   Click on the `Variable Name` field. The cursor will appear in the variable value field, ready to accept keyboard input. Type `jlblCurrentTime`. Once you press ENTER or click elsewhere in the IDE, the new variable name will be assigned.
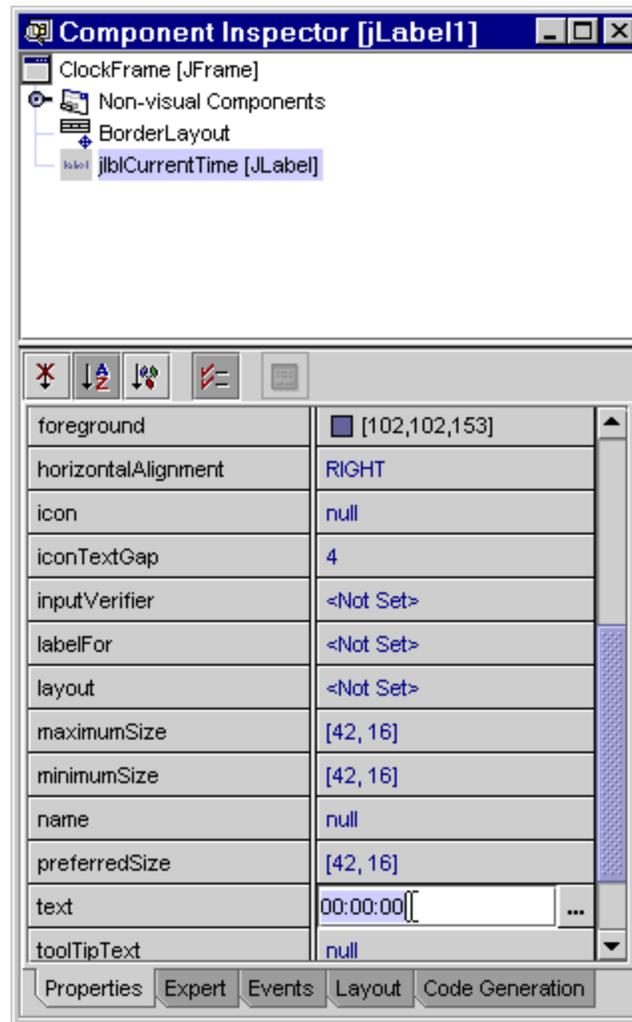
3 Next, find the `text` property on the **Properties** tab. Depending on the size of your Component Inspector widow, you may need to scroll down to see all available properties. Click on the property's value (currently set to a default of `jLabel1`) and enter the text to appear on the Label – type <u>00:00:00</u>.

4 This JLabel will be the main display of our clock, so let's change the default font properties. Click the `font` property in the Component Inspector and select the "..." browse button which appears. A font properties dialog box will open. Change the font face to Serif, Bold, 36 pt. Click **OK** to confirm the selection. You should see the default text (`00:00:00`) on the Form Editor window reflect your changes.

5 Lastly, we will center the time display. Change the `horizontalAlignment` property from its default value (`LEFT`) to `CENTER`.

This completes the visual aspect of the first stage of this tutorial. Now we will add functionality to this form by adding the TimerBean and some code.

Functionality – Adding Code:

1 First, we will add some imports to the code. Switch to the source Editor, and scroll to the top of the code. This form will require the standard date and time imports. Copy the following code, and paste it into the Editor directly under the line reading `package examples;`. When you paste your code into the Editor, it may not stay correctly indented. To indent a block of code, select the block and press TAB or SHIFT+TAB to correctly align the block.

```
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.Calendar;
import java.text.SimpleDateFormat;
```

We will also use the standard JOptionPane for error messages:

```
import javax.swing.JOptionPane;
```

2 Add the following three lines below the `Variables Declaration` block towards the end of the code (below the protected Form Editor code marked with a shaded background).

```
private GregorianCalendar gCal = new GregorianCalendar();
private String timeFormat = "hh:mm:ss";
private SimpleDateFormat formatter = new
       SimpleDateFormat(timeFormat);
```

```
61    // Variables declaration - do not modify
62    private com.netbeans.timerbean.Timer tmrSeconds;
63    private javax.swing.JLabel jlblCurrentTime;
64    // End of variables declaration
65 private GregorianCalendar gCal = new GregorianCalendar();
66    private String timeFormat = "hh:mm:ss";
67 private SimpleDateFormat formatter = new SimpleDateFormat(timeFormat);
68 }

  67:73    INS  'timeFormat' found at 83:18
 ClockFrame *  ClockFrame1
```
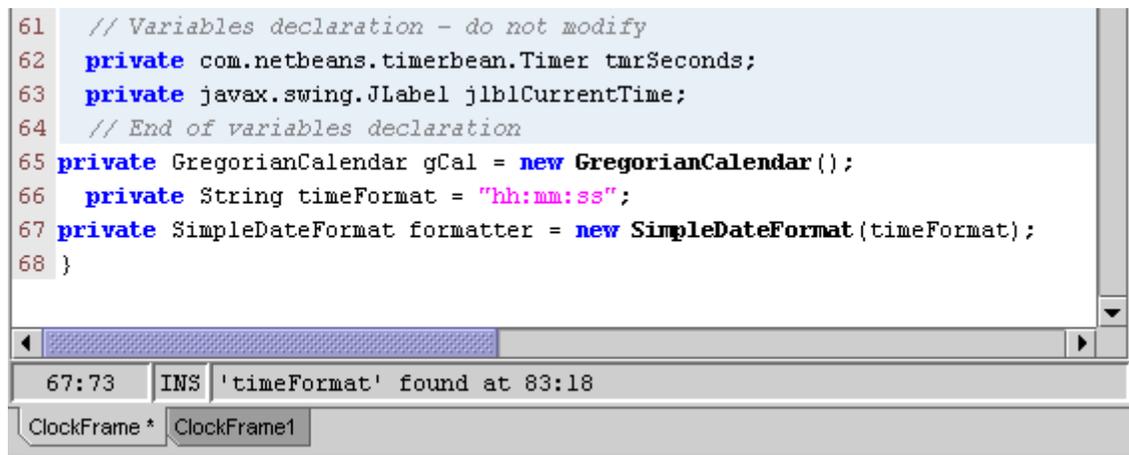
Add the TimerBean, and set an event handler:

1 Choose the **Beans** tab of the Component Palette and select **Timer**. Place it anywhere on the Form work surface. The TimerBean is a non-visual component, so you will not see anything appear on the Form Editor window. However, you will see the TimerBean listed in the Component Inspector under the `NonVisual Components` heading.

2 Select the TimerBean in the Inspector, and change its `variable name` to <u>tmrSeconds</u>.

3 Flip to the **Events** panel of the TimerBean's properties in the Inspector. Set the `onTime` event to `tmrSecondsOnTime`. You will see the Editor window generate the new method. If you scroll back up through the code, you will see that the listener which invokes this method has also been

generated.

4 Now we will add the code for this new method in the Editor. Add the following code under the `// Add your handling code here` line:

```
gCal.add(Calendar.SECOND,1);
String timeTxt = formatter.format(gCal.getTime());
if (jlblCurrentTime != null)
    jlblCurrentTime.setText(timeTxt);
```

Compiling and Executing the form:

1 The basic clock is now complete. Select the **Execute** icon from the Main Window. Watch the status bar of the Main Window – you will see the progress of the operation. Your form and code are first saved and then compiled.

2 Assuming there are no errors and compilation is completed successfully, Forte For Java will switch to the Running Workspace and the form will open. Note that the Execution View, also open on the Running Workspace, displays the ClockFrame as a currently running process.

That's it!

Again, assuming there are no errors, your clock should be displayed, showing the correct current time, with the seconds incrementing normally.
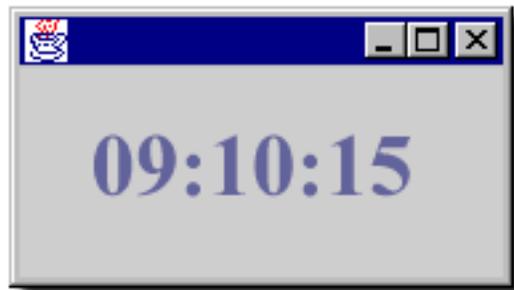
The Completed Clock

You've just built your first form!

To close the form, right-click on it in the Execution View, and choose **Terminate Process**. Note that while you can also terminate this form by closing the form window, this relies on the WindowClosing event being set. The JFrame Template we used to build this form has this event set to close the application as a default setting. Without it, closing the window would not actually terminate the process.

This concludes Part One of the Tutorial. In Tutorial One: Part Two, we will extend the functionality of this clock, adding the ability to set the current time.

Tutorial One: Part Two

Adding a Set Time panel

In Part One of this tutorial, we built a basic functional clock. We will now add the ability to set the time to our form.

First of all, since you have just executed the Clock form, Forte For Java has switched to the Running Workspace, where the source Editor, Form Editor and Component Inspector are not displayed (by default). Use the Workspace tabs on the Main Window to flip back to the GUI Editing Workspace, where all your editing tools are displayed.

Adding a panel and setting the Layout:

1 First add a new JPanel for our new components. Flip to the Swing tab of the Component Palette, and select JPanel. Place it on the `North` panel of the Form Editor surface. You will see the new item in the Component Inspector.

2 Expand this new JPanel in the Component Inspector – you will see the default FlowLayout listed below this JPanel. For this example we will not use this default layout – in the Component Palette, flip to the **Layout** tab. Position your mouse over each of the icons in this group to see the tool-tip identifiers. Select **GridLayout** by single-clicking its icon, and assign this layout to our new JPanel by clicking once on the JPanel in the North part of the Form Editor. You will see the new layout replace FlowLayout in the Component Inspector and a grid appear on the Form Editor surface.

3 The default GridLayout includes 3 columns and 2 rows. In fact, we only need 1 row – select the GridLayout in the Component Inspector, and change the `Rows` property from the default of 2 to 1. The grid displayed in the Form Editor window will change accordingly.

Adding new components:

1 Next we will add some visual Swing components to the new panel. Flip to the Swing panel of the Component Palette, select **JLabel**, and place it anywhere on the new panel.

2 Now select **JTextField** from the Swing panel, and place it on the panel by clicking anywhere on the

JPanel in the Form Editor window. Using this layout, visual components are ordered left to right in the order that you add them.

3 Lastly, select **JButton** from the Swing panel of the Component Palette, and place it on the JPanel – you should now see the three components, equally sized, across the top of the Form.
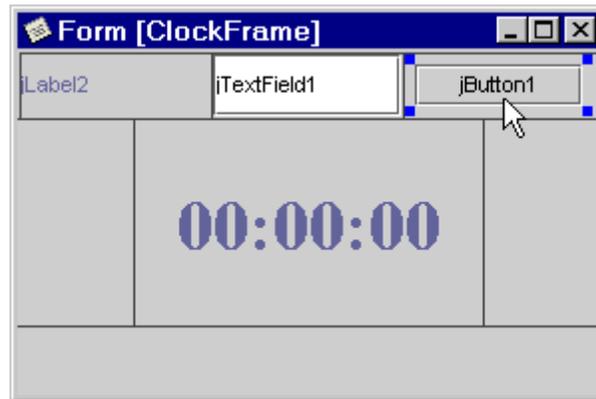
Changing properties:

1 Now we will modify the properties of these new components. Select the JLabel component, either by clicking on it in the Form Editor window or by clicking on it in the Component Inspector. Flip to the **Code Generation** tab in the Component Inspector.

2 First change the `variable name` to `jlblNewTime`.

3 Next change its `text` property on the **Properties** tab to New Time:. You will see the text appear on the Label in the Form Editor.

4 Find the `horizontalAlignment` property, and select the new alignment - `CENTER`.

5 Click the JTextField component in the Component Inspector, change its variable name property to jtfNewTime, and set the `text` property to a default time of 00:00:00

6 Select the JButton in the Component Inspector. Change the `variable name` of the JButton to jbtnNewTime. Change its `text` value to Set New Time.

Adding functionality:

**1** Select the JButton you have just added in the Component Inspector, and flip to its **Events** panel. Set the `actionPerformed` event to jbtnSetNewTimeClicked. You will see the new event handler generated in the Editor.

2 Add the following code to this new handler:

```
try {

String timeStr = jtfNewTime.getText();

gCal.setTime(formatter.parse(timeStr));
} catch (java.text.ParseException e) {
```

```
        JOptionPane.showMessageDialog(this,

    "Invalid date format",

    "I don't understand your date format.",
        JOptionPane.ERROR_MESSAGE);
    }
```

Compiling and executing:

1 Press CTRL+F9 to execute the new form. Again you will see the status bar of the Main Window indicating the progress of the execution. Once compiled and running, try setting a new time by clicking the **Set Time** button. If you enter a time not in the default "hh:mm:ss" format, an error dialog box will open.
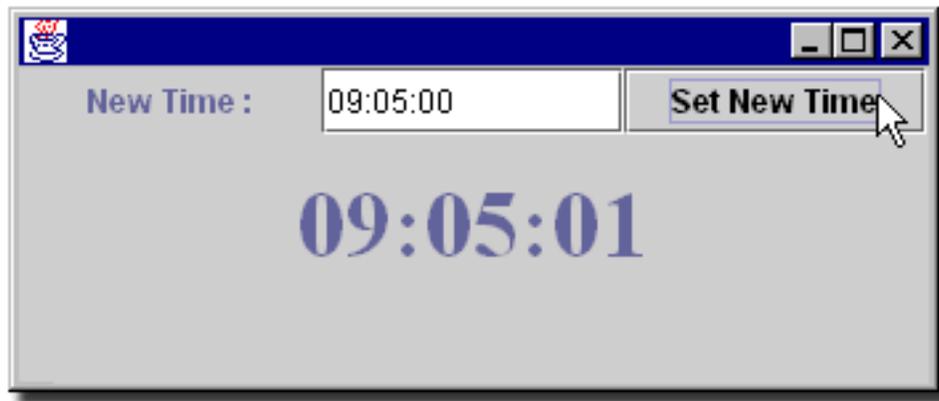
Part Two Completed

2 Once you have verified your Clock is working, again terminate the process using the context menu in the Execution View window.

3 This concludes Part Two of the Tutorial. In Tutorial One: Part Three, we will add a panel allowing the date and time format to be modified.

Tutorial One: Part Three

Adding a Set Format panel

In this final section we will add the option of setting the time format.

Adding a panel and setting the layout:

1   Switch back to the GUI Editing Workspace to see your editing windows – the Explorer, Form Editor window, Component Inspector, and Editor.

2   Add a new JPanel to the East panel of the Form Editor window surface. Select the new JPanel item in the Component Inspector, and flip to its **Layout** tab. You will see the Direction property is set to East, where you just placed the JPanel. In fact we want this new panel on the South part of the Form – click the direction, and select South from the drop-down list. You will see the JPanel repositioned in the new location.

3   We will again change the Layout of this new JPanel – select **GridLayout** from the **Layout** tab of the Component Palette, and drop it onto the new JPanel. Select the GridLayout in the Component Inspector, and change the `Rows` property from the default of 2 to 1.

Add some components:

1   Position a **JLabel** from the **Swing** tab of the Component Palette on the new JPanel. Also add a **JTextField**, and lastly, a **JButton**. The components will appear in the order you place them, across the South panel of the Form.

Setting the properties:

1   Again we will modify the default properties of these new components. Set the JLabel's `variable name` to `jlblNewFormat`, and its `text` property to `Time Format`. Change its `horizontalAlignment` property to `CENTER`.

2   Set the JTextField `variable name` to `jtfNewTimeFormat`, and change the `default text` to `hh:mm:ss`.

3   Set the JButton `variable name` to `jbtnNewTimeFormat`. Set the `text` to read `Set new`

time format.
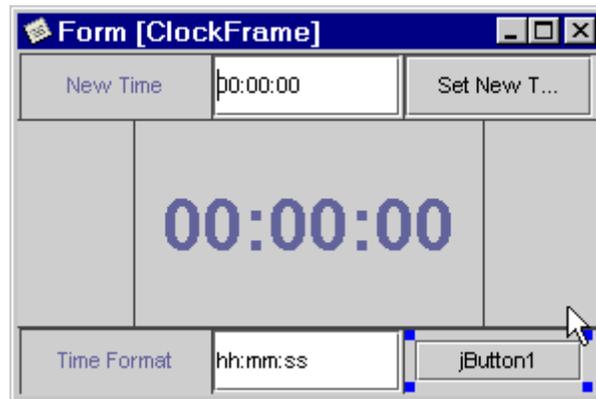
Adding functionality:

1 Select the jbtnNewTimeFormat button in the Component Inspector, and flip to its **Events** panel. Set the actionPerformed event to jbtnNewTimeFormatClicked. You will see the new event handler generated in the code.

2 Add the following to the handler generated:

```
String timeFormat = jtfNewTimeFormat.getText();
formatter = new SimpleDateFormat(timeFormat);
```

Compiling and executing:

1 Execute the completed code from the **Build** menu of the Main Window.

You can now set the time format using the SimpleDateFormat syntax (described in the JDK documentation – *$JDK_HOME*/docs/api/java/text/SimpleDateFormat.html, where *$JDK_HOME* is the directory where the JDK installed)

For example, try the entering following in your New Format text panel: EEEE, d MMMM, hh:mm:ss a .
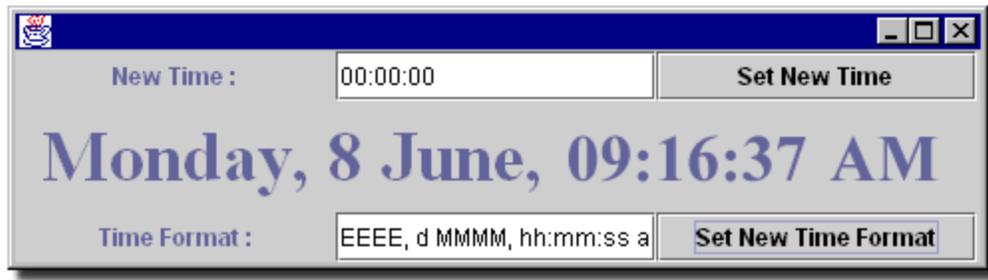
The Finished Form

This concludes Tutorial One. On the Running Workspace, right-click on the ClockFrame item appearing in the Execution Window, and select **Terminate Process**. This will close the currently running ClockFrame.

In Tutorial Two: Part Two, we will use Forte for Java's beans support to build a JavaBeans component, and then we will build a form using that bean to set the background color using RGB sliders.

Tutorial Two: The Color Picker

In this tutorial we will first use the Bean Patterns feature to build a JavaBeans component with three properties – red, green, and blue – which are used to set the background color. In Part Two, we will create a form incorporating this bean, which uses sliders to set these RGB values and display the resulting color.

Part One – Building a JavaBeans component

If you have not already done so, switch back to the GUI Editing Workspace. If you have just completed Tutorial One, you probably still have the Clock Form and Editor open – close these windows.

We will now create a new class from template and a package for that class at the same time.

Creating the class from template:

1   If you don't have one open, open an Explorer window from the **Explorer** icon on the Main Window.

2   Right-click on the `Examples` directory, and select **New From Template | Classes | Class** from the context menu. Call your new package `colorpicker`.

3   The Target Location dialog box will appear with `ClassName` as the default name for the class and `examples` as the default package. Type `ColorPreview` as the class name. For the **Package** name, place the cursor after `examples` and add `.colorpicker` so that the package name reads `examples.colorpicker`.

4   Click **Finish** to create the class.

We will use Bean Patterns to create a JavaBeans component. The bean will have three properties –

red, green and blue – and display these values as a background.

The Bean Patterns:

1 Expand the ColorPreview class node under the `ColorPreview` node. (The Bean Patterns node will appear.)

2 Since ColorPreview extends JPanel, you need to open the property sheet of the `class ColorPreview` node by choosing **Properties** from its popup menu and changing the `Extends` property from `java.lang.Object` to `javax.swing.JPanel` .
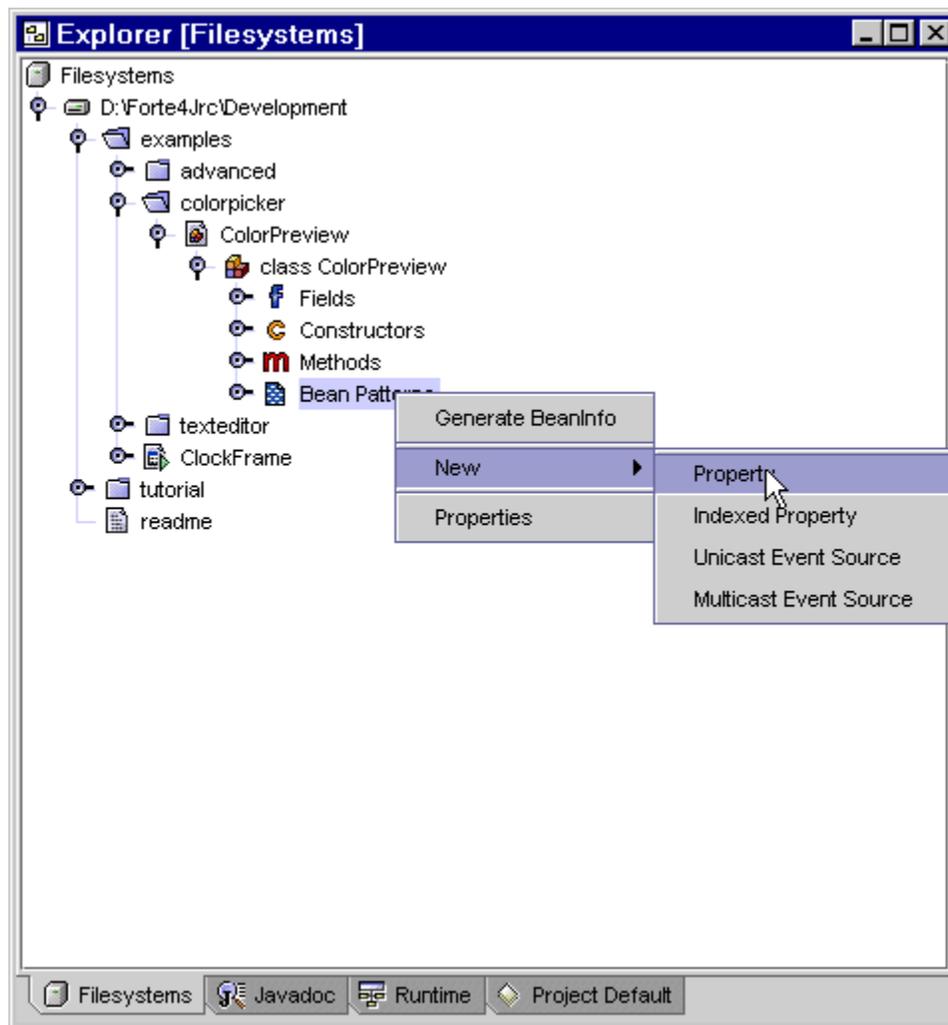
3 Right-click on the Bean Patterns node and select **New | Property** from the popup menu. The New Property Pattern dialog will appear.

4 Now you will generate the property `red`. Enter <u>red</u> in the **Name** field and select `int` as the **Type** and `Read/Write` as the **Mode** of the property. Also check the **Bound, Generate Field, Generate Return Statement, Generate Set Statement**, and **Generate Property Change Support** options. Finally, click **OK** to confirm your selections.

Repeat these steps for the `green` and `blue` properties.
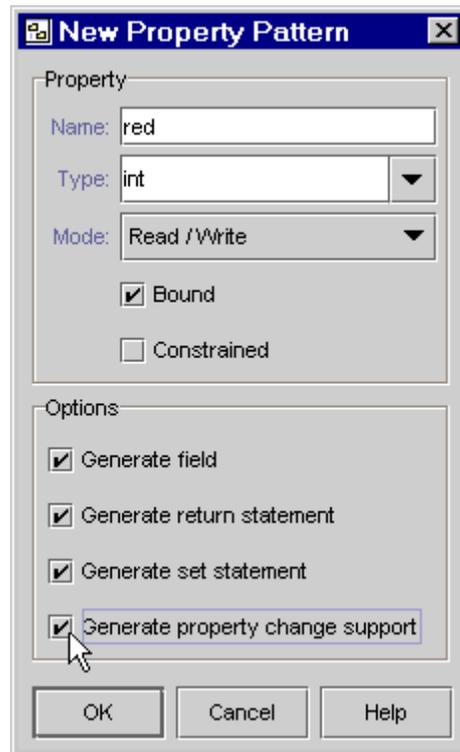
We will now need to manually add some code to the set methods of the color properties.

Adding code:

1   In the Editor window, find the `setRed` method. Immediately under the line reading:

```
propertyChangeSupport.firePropertyChange("red", new
        Integer(oldRed), new Integer(red));
```

add the following lines:

```
setBackground (new java.awt.Color(red,green,blue));
repaint();
```

2   Copy and paste this same code to each of the other methods – both `setGreen` and `setBlue`.

We would like to assign an icon to the ColorPreview bean. So we will need to generate its BeanInfo and in it specify the location of the icon.

The icon is already in the `tutorial/colorpicker` package, so we will copy it from there. Expand `tutorial/colorpicker`, right-click on the third node in the hierarchy (like the node above it, it is named `ColorPreview`), and choose **Copy** from its context menu. Then right-click on the `examples/colorpicker` node and choose **Paste | Copy** to put a copy of the icon into the package.

Generating BeanInfo

1 Right-click on the `Bean Patterns` node of the ColorPreview bean and select **Generate BeanInfo** from the popup menu. The Generate BeanInfo Dialog will appear.

2 Select the `Bean Info` node on the left tab and its properties will appear on the right tab

3 Set the `Icon 16x16 Color` property to `ColorPreview.gif`. Please note that the `ColorPreview.gif` must be presented in the `colorpicker` folder. So you should copy the `tutorial/colorpicker/ColorPreview.gif` to your `Examples/colorpicker` folder.

4 The `ColorPreviewBeanInfo` node will appear in the Explorer under the `Examples/colorpicker` folder.
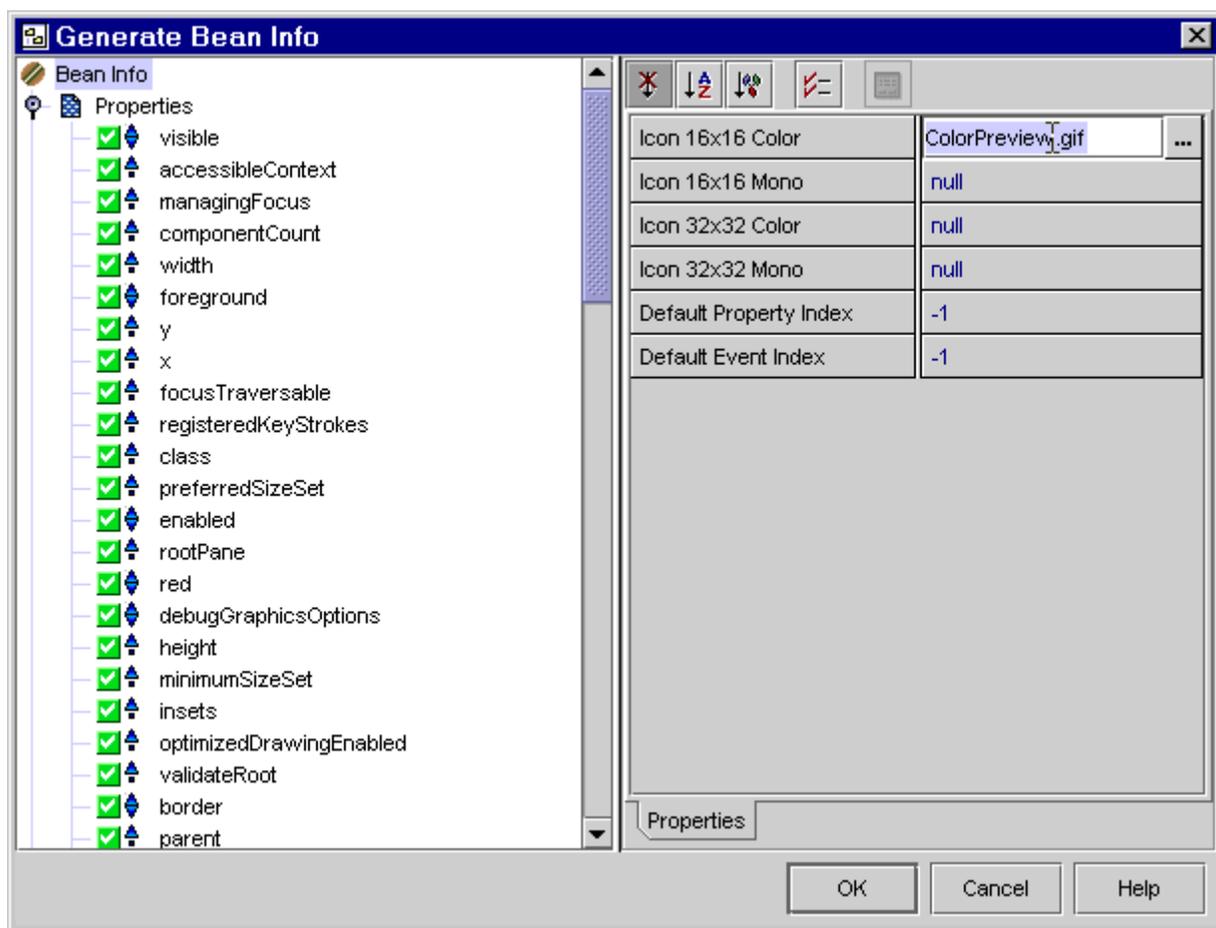
**5** Save and compile the ColorPreviewBeanInfo

Compiling your bean:

◊ Right-click on the `colorpicker` package in the Explorer, and select **Compile** from the context menu to compile all out-of-date classes in this package. Assuming there are no errors, you can close the Editor window.

◊ Once you have seen the bean in action click **Cancel** on the Customize dialog to close it.

We will now add our new bean to the Component Palette, where it will be available for use just like any standard component.

Add the new bean to the Component Palette:

1   Right-click on ColorPreview in the Explorer window, and select **Tools | Add To Component Palette** from the context menu. The Palette Category dialog will appear.

2   Select the **Beans** item from the Palette Category and confirm the selection.

Flip to the **Beans** tab of the Component Palette on the Main Window, and you will see your new bean installed and ready for use.
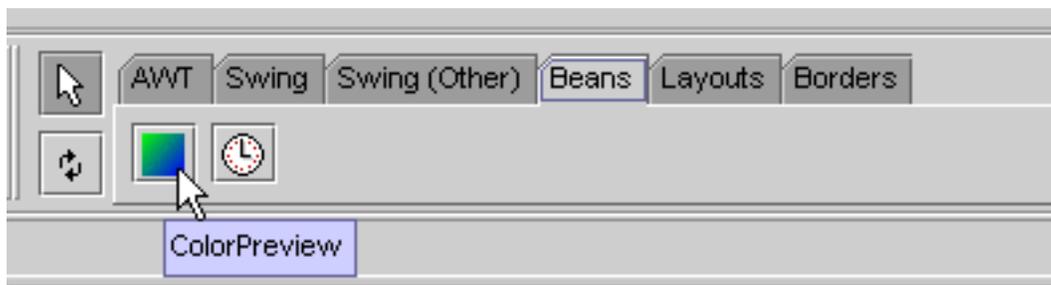
This concludes Part One of the Color Picker tutorial. In Tutorial Two: Part Two, we will build a Form which uses this bean and allows background color to be set via sliders.

Tutorial Two: Part Two

The Color Picker Form

Startup:

1 Right-click on the `colorpicker` package in the Explorer, and select **New From Template | Swing Forms | JFrame**.

2 Give your new JFrame the name <u>`ColorPicker`</u>, and click **Finish**. The JFrame template will open in the Editor window, and the Form Editor and Component Inspector windows will open.

Adding Components:

1 We will now add some components to the JFrame. On the Component Palette, flip to the Swing Tab, and select **JPanel**. Click on the Center panel of the Form Editor surface to add the JPanel to the form. You will see the new JPanel (named `JPanel1`) appear in the Component Inspector.

2 Rename this JPanel <u>`colorPreviewPanel`</u>, either by changing the `variable name` in the properties listed, or by in-place renaming of the item in the component listing at the top of the Component Inspector

3 We will not use the default layout for this JPanel. Right-click on the `colorPreviewPanel` node and choose **Set Layout | BorderLayout** from the context menu.

4 Now we will add the JavaBeans component created in Part One. Flip to the **Beans** tab of the Component Palette, select the **ColorPreview** bean, and click once on the center panel of the `colorPreviewPanel` JPanel to position the bean there.

5 Next we will add the three sliders which will be used to set the color.

Add a new JPanel to the North panel of the Form Editor surface: flip to the **Swing** tab of the Component Palette, select **JPanel,** and click on the north panel of the form surface.

6 Flip to the **Layouts** tab of the Component Palette, select the **BoxLayout** icon, and place it on the new (north) JPanel.

7 Flip to the **Swing (Other)** tab of the Component Palette, and select **JSlider**. Place a JSlider on the

new JPanel.
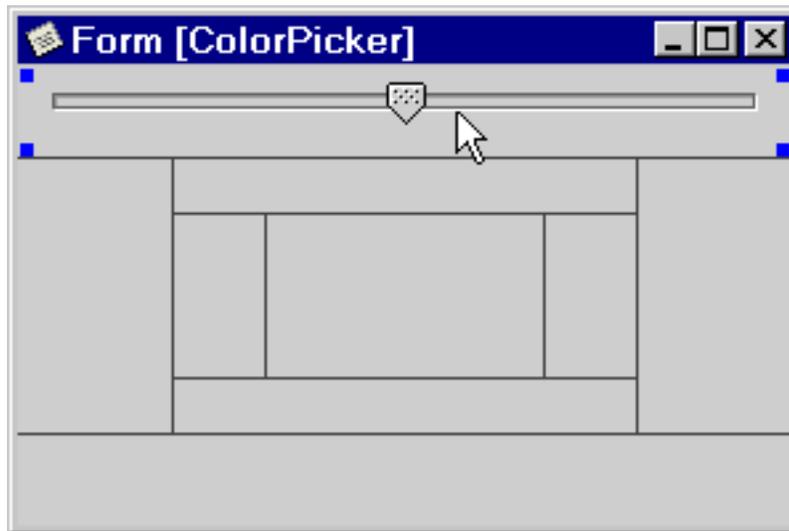
8 We will need a slider for each color property (red, green and blue). Select **JSlider** again: this time hold down the SHIFT key as you click on the form surface (on the same north JPanel where the first JSlider is). This will allow you to add multiple components without needing to reselect them from the Component Palette. Add two more JSliders, so that you have a total of three.

9 Name each of your new sliders – set the `variable name` property in the Component Inspector to <u>redSlider</u>, <u>greenSlider</u>, and <u>blueSlider</u>, for the first, second, and third sliders in the component listing, respectively.

10 Now we must set the maximum allowed value of each slider. Select the red slider on the form surface by clicking on it, and then by holding down CTRL, select both of the other sliders. You should see the anchor marks indicating the component is selected appear around each slider on the form. In addition, the components will be highlighted in the Component Inspector listing. Change the `Maximum` property in the property listing to 255, and press ENTER. This changes that property for all three sliders.

Adding Borders:

1 Next we will add a border to each of the sliders. Flip to the Borders tab of the Component Palette, and select **TitledBorder**. Again, hold down the SHIFT key to add multiple borders, and add one to each JSlider. Click directly on each JSlider on the form surface – you should see the borders appear around each.

2 Now we will set the text of the slider borders.

3 In the Component Inspector, select the red slider and click on its border property and then the "..." icon to open the Border Properties dialog. Change the `Title` property of this border to <u>Red</u>. Remember to press ENTER to set this new property, and click OK.

4 Repeat this procedure for both of the other sliders and title them <u>Green</u> and <u>Blue</u>, respectively.

5 We will also add a border to the `colorPreviewPanel` panel. Select **Titled Border** from the

Borders tab of the Component Palette, and place it anywhere on the `colorPreviewPanel` JPanel, except for the center panel, where the ColorPreview bean is located.

6 Select `ColorPreviewPanel` in the component listing, and open its Border Property dialog by clicking its `border` property and then "...". Set its title to <u>Color Preview</u>, press ENTER, and click **OK**.
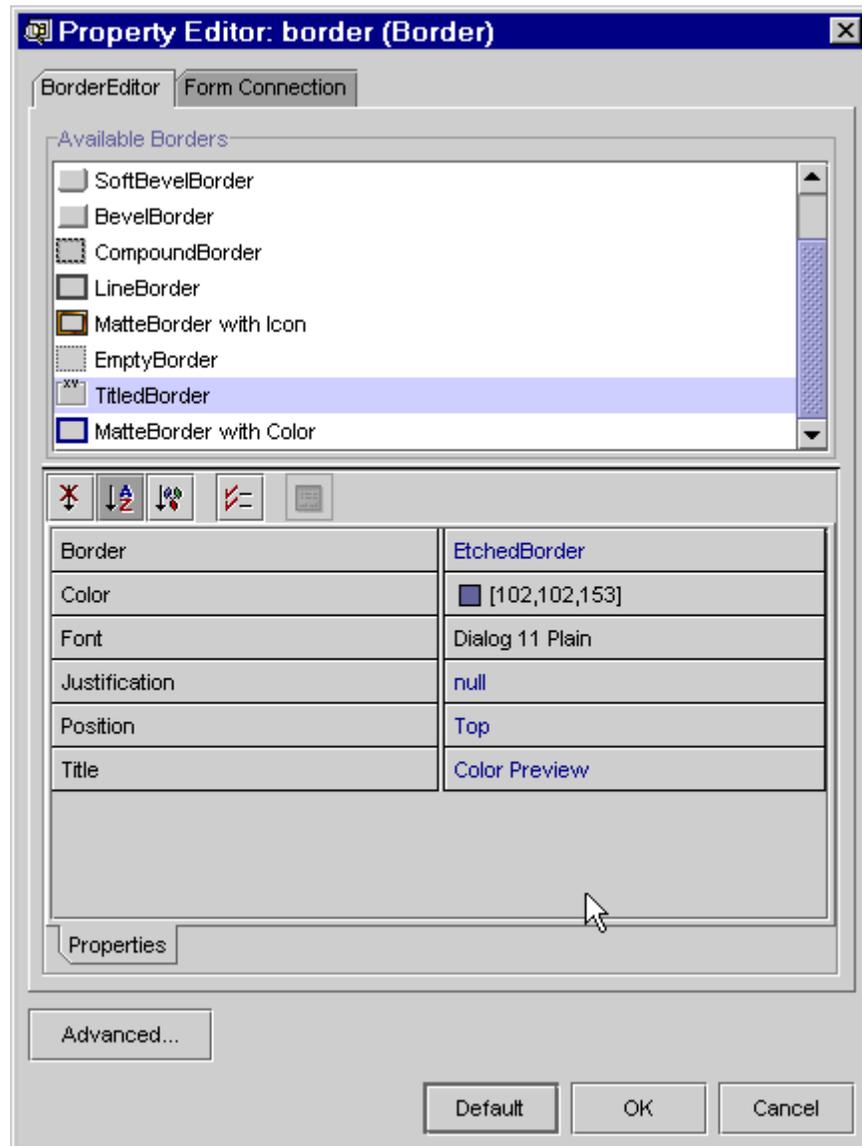
Finally, we will use the Connection Wizard to connect the sliders to the bean.

The Connection Wizard:

1 Click the **Connection Mode** icon, which appears on the Main Window immediately to the left of the Component Palette. The icon will appear "clicked", indicating Connection Mode is active.
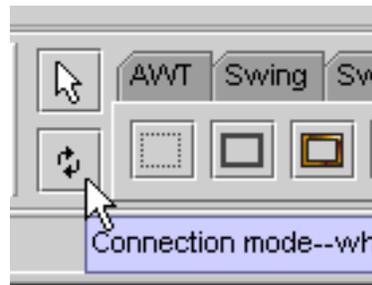
2 Click first on the Red JSlider on the Form Editor surface, and then the center panel of the Color Preview panel, where the `colorPreview1` bean is located. The Connection Wizard dialog will open.

3 Expand the "change" node, and select `stateChanged`. Click **Next** to continue.

4 With the **Set Property** radio button checked, select the `red` property, and click **Next**.

5 In the final Connection Wizard dialog, click the **Property** radio button, and select "..." to browse. Select `value` from the list, and click **OK**. Lastly, click **Finish** to dismiss the Connection Wizard.

6 Repeat the previous three steps for each of the other JSliders, selecting the green and blue properties respectively in step 2.

Repositioning the sliders

◊ Lastly, we will reposition the Sliders. In the Component Inspector, select the BoxLayout of the JPanel1 component. Double-click on the `Axis` property to toggle to the `Y Axis` value. This is a more convenient display of the JSliders in this example.

That's it! Save your form via the **File | Save** menu item or using the **Save** icon on the Main Window, and compile from the **Build | Compile** menu item. Execute it via the **Build | Execute** menu item, or by pressing CTRL+F9.

Forte For Java will switch to the Running Workspace and display the running Color Picker form. Adjust the sliders and check the displayed color changes accordingly.

To close the Color Picker, use the Execution View, which is also displayed on the Running Workspace. Right-click on the `ColorPicker` item listed in the Execution View, and select **Terminate process** from the context menu.

This concludes Tutorial Two.

Tutorial Three: The Image Viewer

In this tutorial, we build a simple image viewer.

Part One

Startup:

1   If Forte For Java is still on the Running Workspace from the previous tutorial, terminate any currently executing processes (via the Execution View listing), and switch back to the GUI Editing workspace. If you have any Forms or Editor windows open from previous tutorials, save your work if necessary, and close them.

2   In the Explorer, click the **Filesystems** tab and navigate to the examples directory. Right-click the examples directory, and select **New Package** – call it <u>imageviewer</u>. You will see the new package appear in the Explorer.

3   Right-click the new package, select **New From Template | Swing Forms | JFrame**. Name the new JFrame <u>ImageViewer</u>. Click **Finish** – the Editor, Form Editor and Component Inspector windows will open.

4   Select ImageViewer node in the Component Inspector and set its title property to <u>Image Viewer</u> and press ENTER.

Adding Components:

1   First we will add a menu to the JFrame. Flip to the **Swing** tab of the Component Palette, and select **JMenuBar**. Click anywhere on the Form Editor surface to add the menu bar. You will see a menu appear on the form surface. In the component listing in the Component Inspector, you will see a menu bar. If you expand its node, you will see a menu listed. Initially the menu has no item.

2   We will add some elements to this menu, using the Menu Editor. Right-click on the menu

(JMenu1) in the Component Inspector and choose **New | JMenuItem**. You will see JMenuItem appear below JMenu1.
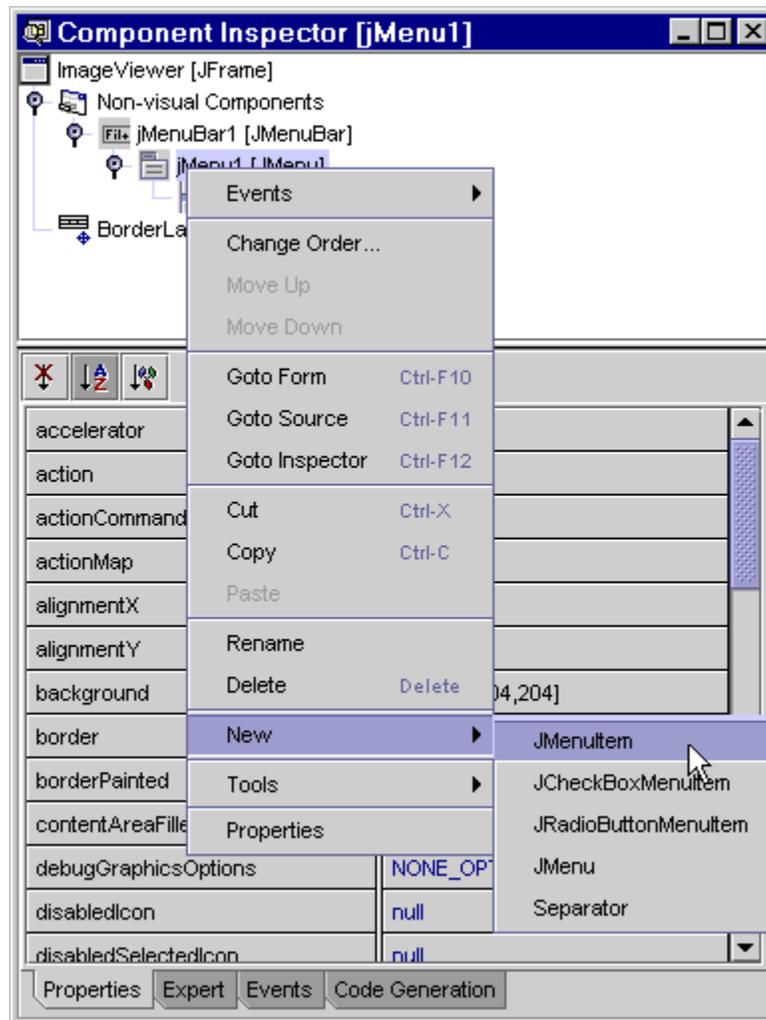
3 Right-click on JMenu1 again and choose **New | Separator**. You will see the separator appear below the first menu item.

4 Next we will add a second menu item. Again right-click on JMenu1 and select **New | JMenuItem.** It will appear below the separator in the menu listing.

5 With the JMenu selected in the Component Inspector, scroll through the list of its properties to the text property; set this to File. Press ENTER to set the new value. Change its variable name from jMenu1 to fileMenu.

6 Select the first menu item in the Component Inspector, change its text to Open, and its variable name to openMenuItem.

**7** Similarly for the second menu item, change its text to Exit, and its variable name to exitMenuItem.

8 Next we will add a JDesktop to the frame, where the images will be displayed. Select **JDesktopPane** from the **Swing (Other)** tab of the Component Palette, and place it on the center panel of the Form Editor surface. Set its `variable name` in the Component Inspector property listing to <u>desktop</u>.

Adding the Code:

Now we need to generate the event handlers for the menu items. There are several ways of doing this. We will demonstrate two of them here.

Adding the event handler:

1 Firstly, for the Open menu item, simply double-click the item in the component list in the Component Inspector. You will see the Editor jump towards the bottom of the code, and the new handler generated.

2 For the Exit menu item, this time actually select **Exit** from the Menu on the Form Editor surface. You will again see the new handler generated in the Editor window.
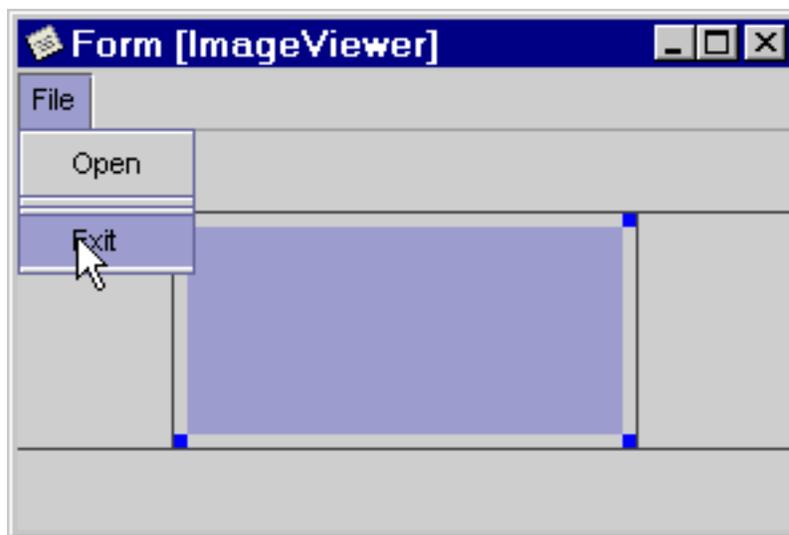
We will now add some code to these event handlers.

Adding code for the event handlers:

1 Firstly, for the **File | Exit Menu** item: find the `exitMenuItemActionPerformed` handler. There will be a line immediately following reading `// Add your handling code here`. Add the following line immediately below this:

```
System.exit(0);
```

2 Next, for the **File | Open** menu item: find the `openMenuItemActionPerformed` handler – this should be just below the Exit menu handler – and copy the following code immediately below the

```
// Add your handling code here comment line.

   java.awt.FileDialog fd = new java.awt.FileDialog (this);
   fd.show ();
   if (fd.getFile () == null) return;
```

This code simply displays the standard **File | Open** dialog, and returns if the **Cancel** button is clicked.

3   Add the following four lines immediately below this (making sure that any lines that wrap below do not wrap when you paste them to the Editor):

```
   ImageFrame ifr = new ImageFrame (fd.getDirectory () + fd.getFile
         ());
   desktop.add (ifr, javax.swing.JLayeredPane.DEFAULT_LAYER);
   ifr.setSize (200, 200);
   ifr.setLocation (0, 0);
```

This is the code that handles the display of the images. We will create ImageFrame in Part Two.

Save the form from the File menu, and close the Form Editor.

In Tutorial Three: Part Two, we will build the ImageFrame.

Tutorial Three: Part Two

ImageFrame

To build the ImageFrame, follow these steps:

1 Right-click on the `imageviewer` package in the Explorer, and select **New From Template | Swing Forms | JInternalFrame**. Name the new form <u>ImageFrame</u>. Click **OK**. The Form Editor will open, and the source will open (assuming your Editor window is still open from Part One) as a new tab in the Editor window.

2 From the Swing tab of the Component Palette, select **JScrollPane**, and place it on the center panel of the new form. Again from the Swing tab, select **JLabel**, and place it on the JScrollPane.

3 Change the JLabel `variable name` to imageLabel in the Component Inspector. Set the text to an empty string, by removing the default text (jLabel1) in the Component inspector listing.

4 Select the top level node of the Component Inspector – ImageFrame. Scroll through the list of properties, and double-click on each of the following items to toggle the property's value from `False` to `True`:

- `closable`
- `iconifiable`
- `resizable`

This will allow us to close, iconify and resize any images we have open in the Image Viewer.

5 Lastly, we will add some code to the Editor. In the code marked `/** Initializes the Form */` towards the top of the source, modify the declaration reading `public ImageFrame()` and add parameters so that the line reads

```
public ImageFrame(String imageName) {
```

Under the `initComponents ();` line in this same block, add the following:

```
    setTitle (imageName);
    imageLabel.setIcon (new javax.swing.ImageIcon (imageName));
```

The Image Viewer is now complete. Right-click on the imageviewer package in the Explorer, and select **Compile All**. Watch the status bar of the Main Window to see the progress of the compilation. Once completed, select the ImageViewer object in the Explorer, and execute it using the **Execute** icon on the Main Window.

Use the File menu to open any GIF or JPG images you have on your local drive(s). If you don't have any images handy, browse to *$FORTE4J_HOME*/docs/Tutorial/images/, where *$FORTE4J_HOME* is your Forte For Java installation directory, and select any file.

You can open multiple images, resize them, iconify them, and close them.
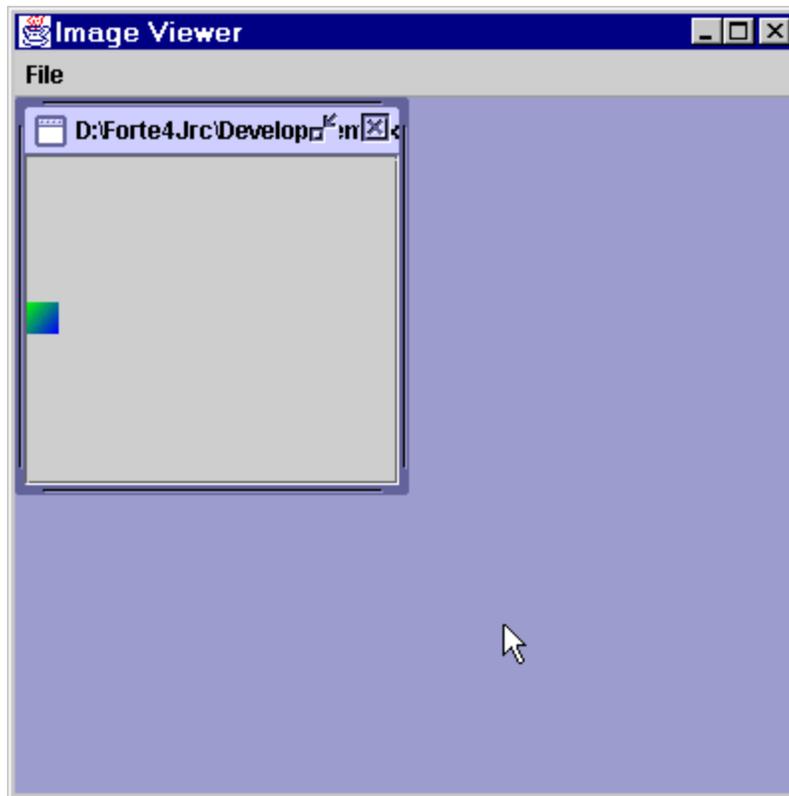

This concludes Tutorial Three.

Tutorial Four: The Debugger

In this tutorial we will demonstrate use of the debugging subsystem of Forte For Java. We will use the completed code for one of the earlier tutorials – part three of Tutorial One, the advanced version of the Clock. The completed code for this tutorial (and all other tutorials) is included with Forte For Java, and can be found under `Development/tutorial/` in the Forte For Java Explorer.

The Debugger allows you to set and remove breakpoints, watch variables, track the state of threads, and more. All of this can be done within the simple and intuitive graphical user interface.

Preliminary Setup:

◊   At a later stage in this tutorial we will need to access the included TimerBean source. To make this source accessible to the IDE, we need to mount it as a new file system. From the **Tools** menu in the Main Window, select **Add Directory**. A standard Browse dialog will open. Navigate to your Forte For Java installation directory, and select the `sources` subdirectory. Click **Add**, and you will see a new file system appear in under the **Filesystems** tab in the Explorer.
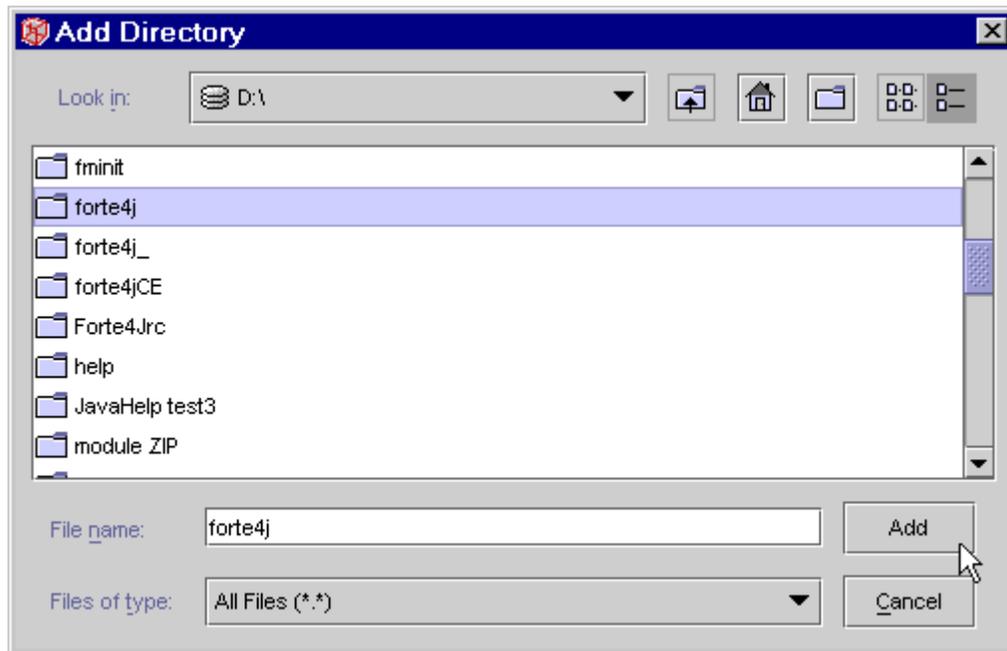
Working with Breakpoints

1   Close any sources and forms you may have open, and terminate any running processes. Flip to the Editing Workspace, open an Explorer window, and expand the `Development\tutorial\clock` hierarchy. We will use the final stage of this tutorial – in the part3 subdirectory. Double-click on `ClockFrame` to open this object in the Editor, Form Editor, and Component Inspector.

2   Click the **Compile** icon in the Main Window, or use the keyboard shortcut F9, to compile this source. You should see the Main Window status line indicating the progress of this command. (If you get a warning about a deprecated API, ignore it – it is harmless in this case.)

3   In the Editor window, find the main method, and position the cursor on the first line of the body. We will add a breakpoint to this line. Right-click on the line and choose **Add/Remove Breakpoint** from the contextual menu or press CTRL+F8. You will see the line highlighted in blue, indicating a breakpoint is set on that line. (Note that putting a breakpoint on the previous line, the one declaring the method, will not work for `main()` – Java calls the body of main methods in a special way for the debugger, and you cannot trace into it.)
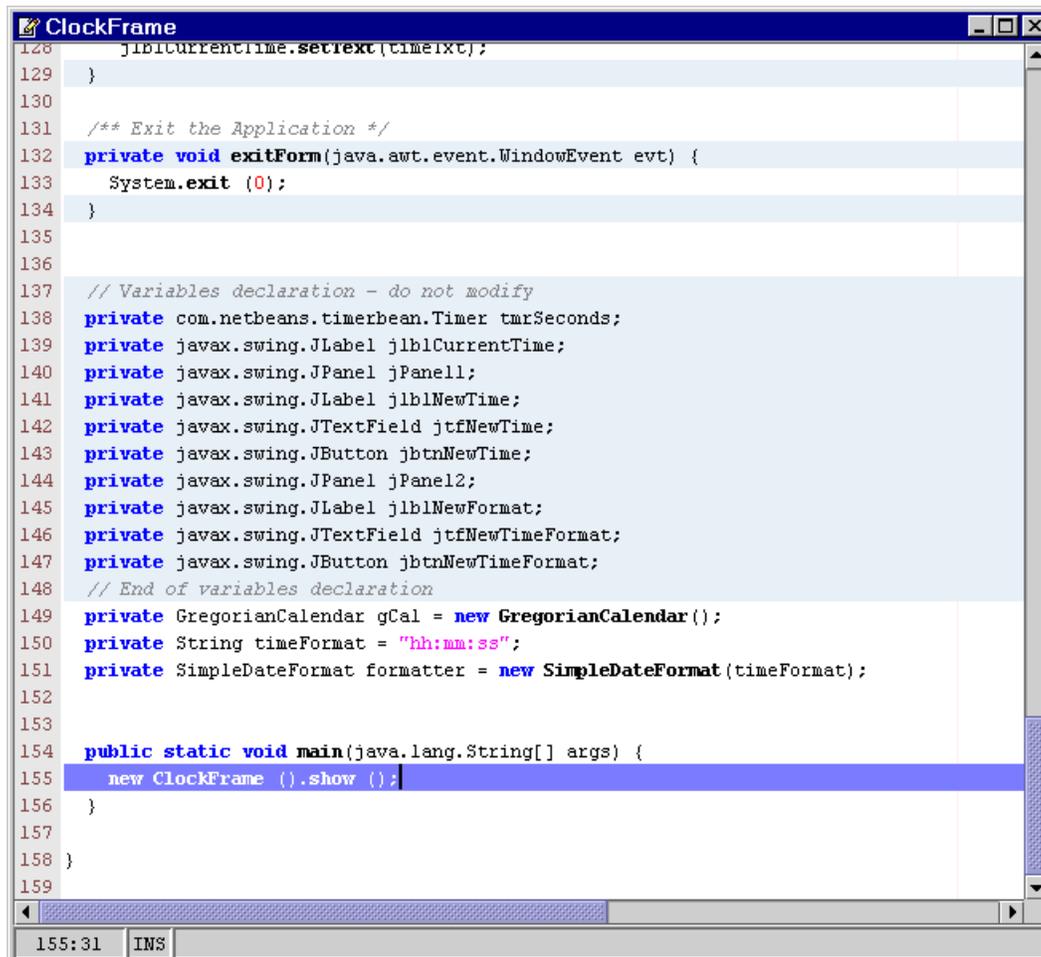
```
ClockFrame                                                    _ □ ×
128    jlblCurrentTime.setText(timeTxt);
129   }
130
131   /** Exit the Application */
132   private void exitForm(java.awt.event.WindowEvent evt) {
133     System.exit (0);
134   }
135
136
137   // Variables declaration - do not modify
138   private com.netbeans.timerbean.Timer tmrSeconds;
139   private javax.swing.JLabel jlblCurrentTime;
140   private javax.swing.JPanel jPanel1;
141   private javax.swing.JLabel jlblNewTime;
142   private javax.swing.JTextField jtfNewTime;
143   private javax.swing.JButton jbtnNewTime;
144   private javax.swing.JPanel jPanel2;
145   private javax.swing.JLabel jlblNewFormat;
146   private javax.swing.JTextField jtfNewTimeFormat;
147   private javax.swing.JButton jbtnNewTimeFormat;
148   // End of variables declaration
149   private GregorianCalendar gCal = new GregorianCalendar();
150   private String timeFormat = "hh:mm:ss";
151   private SimpleDateFormat formatter = new SimpleDateFormat(timeFormat);
152
153
154   public static void main(java.lang.String[] args) {
155     new ClockFrame ().show ();
156   }
157
158 }
159

155:31    INS
```

The debugging session

1 Let's start the debugging session. From the **Debug** menu in the Main Window, choose **Start Debugging**, or press F5. HotSpot users must set the `Classic` property to `True` on the property sheet for `Debugger Types / Standard Debugging/ Standard Debugging` in Project Settings (choose **Project | Settings**... from the main menu to open the Project Settings window). Forte For Java will switch to the Debugging Workspace, and two new windows will open – the Debugger window and the Output window.

The Output window is split vertically, the left panel displaying the output of the debugged program and the right panel showing messages from the debugger itself.

The Debugger window is used to manipulate breakpoints and watch program variables and the state of threads. These are each displayed under a separate tab. Currently under the **Breakpoints** tab you will see the breakpoint we have just set, listed by source name and line number.

34
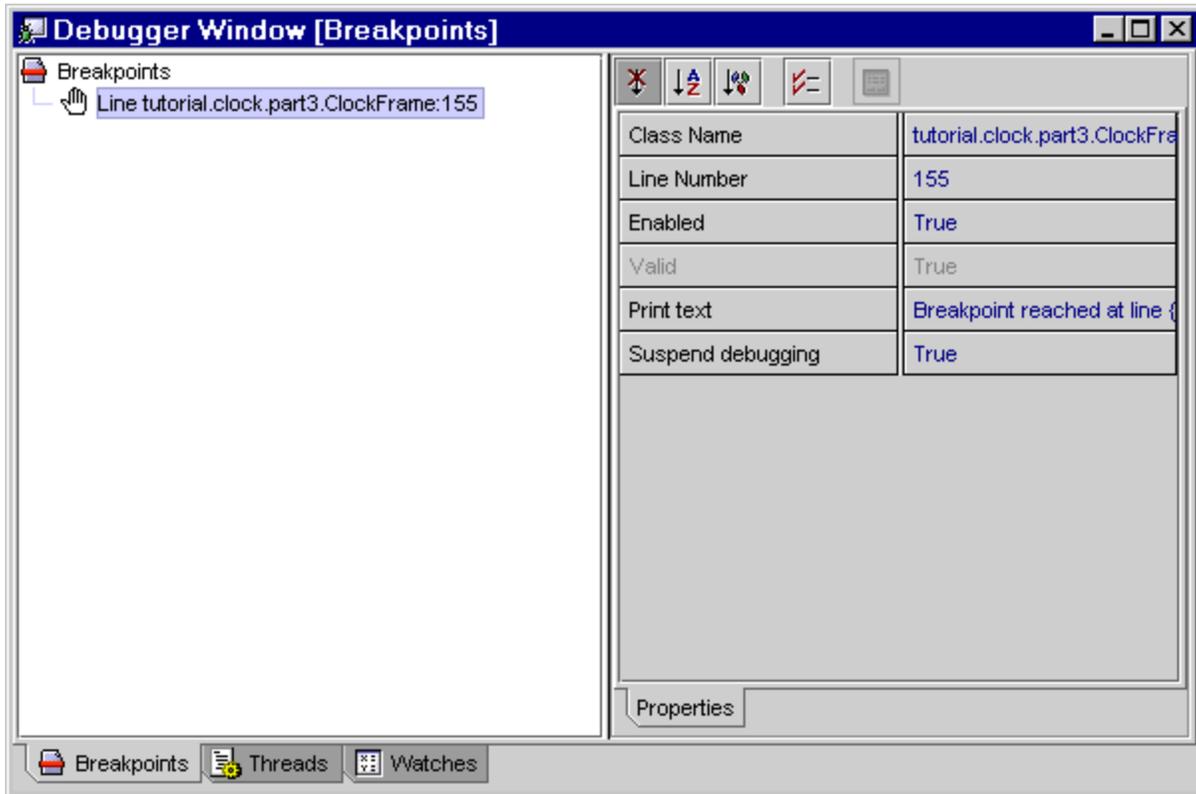
You will see several messages from the debugger in the Output window, and then the debugger will halt at the breakpoint in the main method. The blue-highlighted line in the Editor will change to pink to indicate where execution has halted.

2 At this point you can continue (ALT+F5), Trace Over the current line (F8), or Trace Into the function called on the current line (F7). We wish to step into ClockFrame, so push F7, or select **Trace Into** from the **Debug** menu in the Main Window and then push CTRL+F7 and F7 again.

You will see another line of output from the Debugger in the Output Window, and the Editor window will jump to the constructor of the ClockFrame class and again halt. You should see a pink highlighted line where the debugger is currently stopped.

3 You will next break at the first of the three variable declarations manually entered when creating the tutorial (private GregorianCalendar gCal ...). Push F8 to trace over this; trace over both others by pushing F8 twice more.

4 The pink line highlighting the current point in the code should now be at the line reading initComponents ();. Push F7 to trace into this. You are halted on the line where the instance of com.netbeans.timerbean.Timer is created. Press F7, then CTRL+F7, and then F7 again to step in. Assuming you have mounted the sources directory as a file system as described in the Preliminary Setup section, the Timer source will open in the Editor window, and the Debugger session will now step into it. The pink highlighted line indicates the point in the source where the Debugger is stopped.
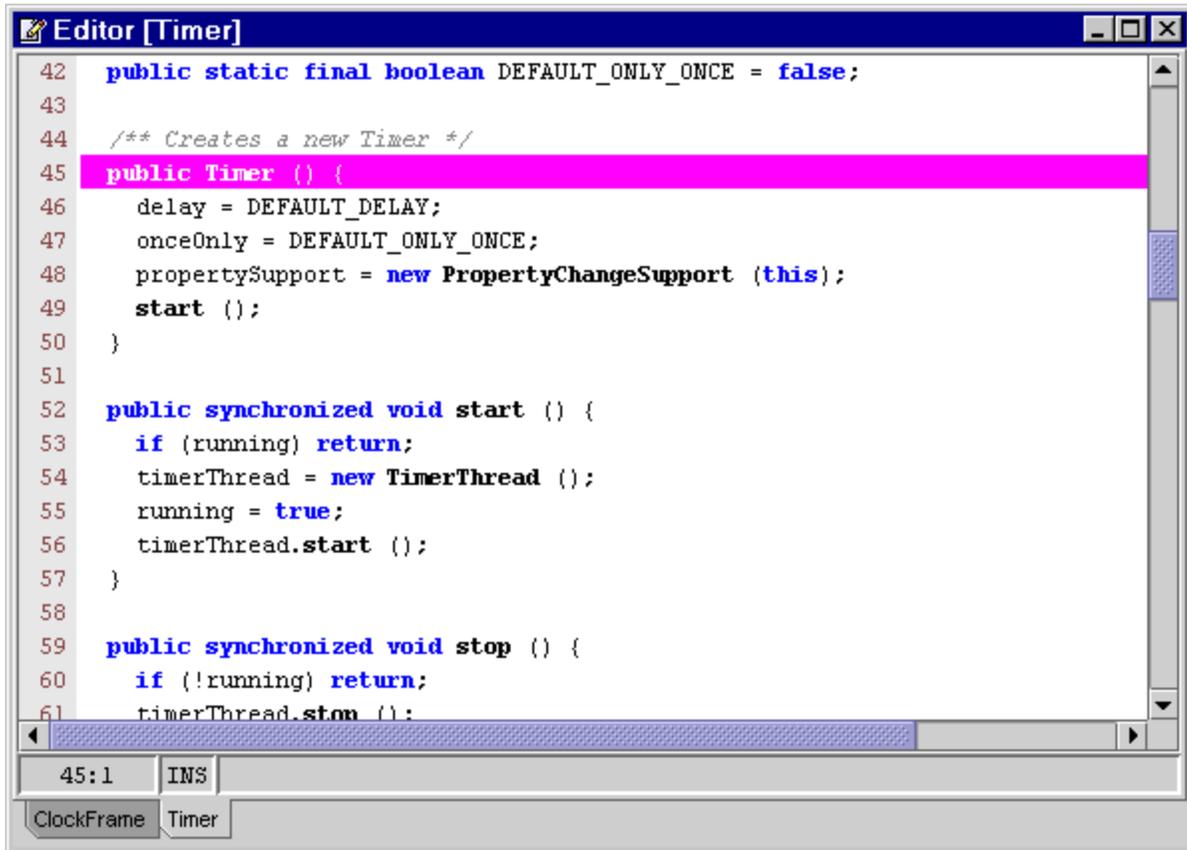
```
Editor [Timer]                                                   _ □ ✕
 42    public static final boolean DEFAULT_ONLY_ONCE = false;      ▲
 43
 44    /** Creates a new Timer */
 45    public Timer () {
 46       delay = DEFAULT_DELAY;
 47       onceOnly = DEFAULT_ONLY_ONCE;
 48       propertySupport = new PropertyChangeSupport (this);
 49       start ();
 50    }
 51
 52    public synchronized void start () {
 53       if (running) return;
 54       timerThread = new TimerThread ();
 55       running = true;
 56       timerThread.start ();
 57    }
 58
 59    public synchronized void stop () {
 60       if (!running) return;
 61       timerThread.stop ();                                       ▼
◄                                                                  ►
   45:1      INS
 ClockFrame  Timer
```

5   Press ALT+F5 to continue. The ClockFrame will open and run.

6   Find the `tmrSecondsOnTime()` method in the ClockFrame source, and set a breakpoint (CTRL+F8) on the line declaring the method. The next time the program flow goes through this point, execution will halt, and the blue breakpoint line will turn pink.

Watching Variables

1   Flip to the **Watches** tab of the Debugger window. Here you can monitor the values of individual variables during execution. To add a new watch, you can select **Debug | Add Watch** from the Main Window or right-click on the root item of the `Watches` tree on the **Watches** tab of the Debugger window, and select **Add Watch**. You can also right-click on the variable in the editor and select **Add Watch**. Now go to the `tmrSecondsOnTime()` method, right-click on the variable `timeTxt`, and press **OK** in the Add New Watch dialog box. `timeTxt` will appear in the `Watches` tree in the Debugger Window.

2   Push ALT+F5 to continue the debugging session. After one second of execution, `tmrSecondsOnTime()` will be called again, and execution will again halt. The value of `timeTxt` displayed in the Debugger window will update when debugger moves over it (an increment of one second).
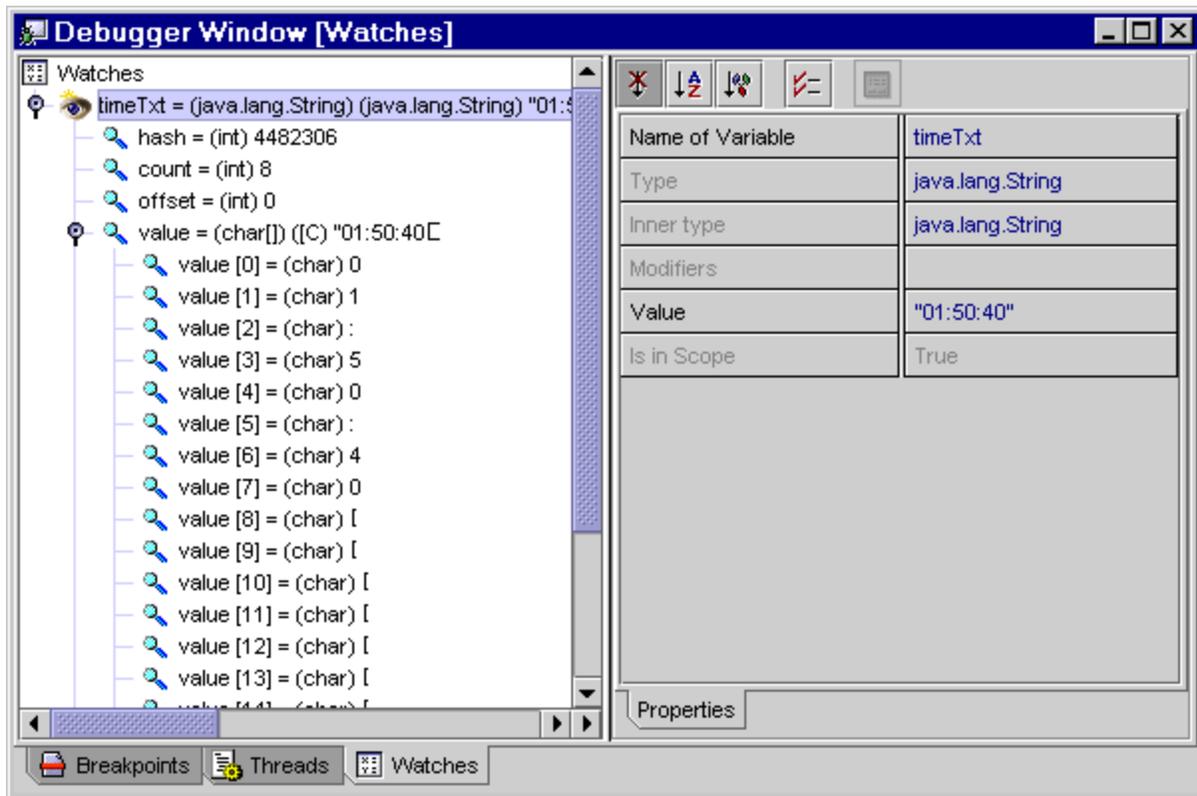
It is possible to watch multiple variables simultaneously – simply add a watch as before. All watched variables are listed in the Debugger Window. You can delete watched variables by selecting them in the `Watches` tree and pressing the DELETE key, or by selecting **Delete** from the popup menu. If a variable is not in the current scope, it does not display any value.

As you use ALT+F5, F7, and F8 to continue, step into, and step over the code, respectively, you can monitor the values of the watched variables at each stage.

Threads

Under the threads tab of the Debugger window, the current state of all threads of the program are listed.

Ending the debugging session

To end a debugging session, select **Debug | Finish Debugging** from the Main Window, or use the keyboard shortcut SHIFT+F5.

Other Features

- The state of the debugging session, including breakpoint locations and watched variables, is preserved across sessions. It is not necessary to explicitly save the session.

- You can customize the Debugging subsystem from the `Debugger Types` node in the Project Settings window and the `Debugger Settings` node in the Global Options window. For more information, see the User's Guide to Forte for Java, Community Edition.