More Swing

SetActionCommand

In our last example we had three buttons that invoked the same ActionListener. We were able to tell which button was clicked by checking the getActionCommand() method. This method returned the text that was set in the button.

```
public class JavaApplication18 extends JFrame implements ActionListener
{
    private JPanel redPanel;
    private JPanel whitePanel;
    private JPanel bluePanel;

    @Override
    public void actionPerformed(ActionEvent e)
    {
        String cmd = e.getActionCommand();
        if (cmd.equals("Red"))
        {
            redPanel.setBackground(Color.red);
        }
        else if (cmd.equals("White"))
        {
            whitePanel.setBackground(Color.white);
        }
        else if (cmd.equals("Blue"))
        {
            bluePanel.setBackground(Color.blue);
        }
    }

    public JavaApplication18()
    {
        super();
        setSize(810,640);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Master panel in the middle
        JPanel middle = new JPanel();
        middle.setLayout(new GridLayout(1,3));
        this.add(middle, BorderLayout.CENTER);

        // Three panels inside the middle panel
        redPanel = new JPanel();
        whitePanel = new JPanel();
        bluePanel = new JPanel();
        middle.add(redPanel);
        middle.add(whitePanel);
        middle.add(bluePanel);
```

```
        JPanel bottom = new JPanel();
        bottom.setLayout(new FlowLayout());
        this.add(bottom, BorderLayout.SOUTH);

        JButton redButton = new JButton("Red");
        redButton.addActionListener(this);
        bottom.add(redButton);
        JButton whiteButton = new JButton("White");
        whiteButton.addActionListener(this);
        bottom.add(whiteButton);
        JButton blueButton = new JButton("Blue");
        blueButton.addActionListener(this);
        bottom.add(blueButton);

        setVisible(true);
    }
```

But what do we do if the buttons have the same text? Then in this case we can use the setActionCommand method. This method sets a command associated with the button that can then be used to distinguish the button independent of the text:

```
        JButton redButton = new JButton("Color");
        redButton.setActionCommand("Red");
        redButton.addActionListener(this);
        bottom.add(redButton);

        JButton whiteButton = new JButton("Color");
        whiteButton.setActionCommand("White");
        whiteButton.addActionListener(this);
        bottom.add(whiteButton);

        JButton blueButton = new JButton("Color");
        blueButton.setActionCommand("Blue");
        blueButton.addActionListener(this);
        bottom.add(blueButton);
```

This program will behave the same as before except each button's text will say "Color".

Fonts

Use the setFont method with a new Font object to change the font. For example:

```
redButton.setFont(new Font("Courier", Font.BOLD, 24));
```

Menus, Menu Items, Menu Bars

A menu is an object of the class JMenu. A choice on a menu is called a menu item and is of class JMenuItem. A menu is often associated with a menu bar. In Java this is a class JMenuBar.

The following creates a menu to invoke the same ActionListener to color our three panels:

```
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JMenuBar;

        JMenu colorMenu =  new JMenu("Change Colors");
        JMenuItem redMenuItem = new JMenuItem("Red");
        redMenuItem.addActionListener(this);
        colorMenu.add(redMenuItem);

        JMenuItem whiteMenuItem = new JMenuItem("White");
        whiteMenuItem.addActionListener(this);
        colorMenu.add(whiteMenuItem);

        JMenuItem blueMenuItem = new JMenuItem("Blue");
        blueMenuItem.addActionListener(this);
        colorMenu.add(blueMenuItem);

        JMenuBar bar = new JMenuBar();
        bar.add(colorMenu);
        setJMenuBar(bar);
```

If you add additional JMenu objects to a JMenuBar then they will appear as another menu to the right of the previously added menu.

Note that you can also add a JMenuBar to a container just like any other component. For example, it could be added to the WEST or SOUTH quadrant of a BorderLayout. Of course the typical place to add a menu is on the top menu bar.

Nested Menus

We can make nested menus by adding a JMenu to a JMenu. Here is an example with Red:

```
        JMenu colorMenu =  new JMenu("Change Colors");
        JMenu redMenuItem = new JMenu("Red");
        colorMenu.add(redMenuItem);
        JMenuItem lightRed = new JMenuItem("Light Red");
        redMenuItem.add(lightRed);
        JMenuItem darkRed = new JMenuItem("Dark Red");
        darkRed.addActionListener(this);
        darkRed.setActionCommand("Red");
        redMenuItem.add(darkRed);
```

Textfields and TextAreas

You have undoubtedly interacted with a GUI that provides a space for you to enter text information. We have the JTextField and JTextArea components for this purpose.

A text field is a box that lets you enter a single line of text. Use the JTextField class to create a text field object. The constructor takes the number of characters that will fit (be visible) in the text field, although more can be typed but will not be visible. The default is 30 characters.

A text field can be used for input or output. However, be careful if the data is output only, because the mere use of a text field implies that the user is supposed to be entering text. If you have data the is output only you are normally better off using a JLabel instead.

To retrieve whatever string is inside a textfield use:

```
String s = txtfield.getText();
```

To set the text in a text field use:

```
txtfield.setText("New text goes here");
```

Here is an example (a rather ugly one) that allows the user to enter two numbers into two different text fields and then click a button to display their sum.

```
public class GUI_Demo extends JFrame implements ActionListener
{
    private JTextField tfNum1;
    private JTextField tfNum2;
    private JLabel lblSum;

    public GUI_Demo()
    {
        super();
        setSize(810,640);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        JPanel tfPanel = new JPanel();
        tfPanel.setLayout(new FlowLayout());
        this.add(tfPanel, BorderLayout.NORTH);

        tfNum1 = new JTextField(10);
        tfNum2 = new JTextField(10);
        tfPanel.add(tfNum1);
        tfPanel.add(tfNum2);

        JButton btnSum = new JButton("Sum Numbers");
        btnSum.addActionListener(this);
        this.add(btnSum, BorderLayout.SOUTH);

        lblSum = new JLabel("0");
        lblSum.setFont(new Font("Times New Roman", Font.BOLD, 48));
```

```
            this.add(lblSum, BorderLayout.CENTER);

            setVisible(true);
        }

        @Override
        public void actionPerformed(ActionEvent e) {  // Should error check
            String s1 = tfNum1.getText();
            String s2 = tfNum2.getText();
            int num1 = Integer.parseInt(s1);
            int num2 = Integer.parseInt(s2);
            String sum = (num1 + num2) + "";
            this.lblSum.setText(sum);
        }

        public static void main(String[] args) {
            GUI_Demo app = new GUI_Demo();
        }
    }
```

If you want multiple lines instead of a single line then use a JTextArea. For example, the following constructor creates a JTextArea that is 5 lines with 20 chars per line with the default text of "change me":

```
JTextArea myarea = new JTextArea("change me",5,20);
```

If setEditable(true) then the user can edit the text.

If setEnabled(true) then the user can use the component, otherwise it is grayed out.

If setLineWrap(true) then text will wrap from the right back to the left.

We will say more about scroll bars later, but if you want to add scroll bars to a JTextArea then the easiest way is to put the JTextArea object inside a JScrollPane. The JScrollPane will automatically add the scroll bars. Put the JScrollPanel into the JPanel:

```
taNum1 = new JTextArea("change me",5,20);
JScrollPane sp = new JScrollPane(taNum1);
tfNum2 = new JTextField(10);
tfPanel.add(sp);
tfPanel.add(tfNum2);
```

Class exercise: Make a GUI program with four buttons (up down left right) that moves a circle in a JPanel in the Center layout.