

10

User Interfaces

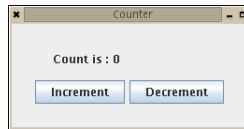
Self-review Questions

- 10.1 *What is a GUI component?*
- 10.2 *What does a layout manager do?*
- 10.3 *What are the advantages of using a layout manager?*
- 10.4 *Why are panels layered?*
- 10.5 *What role does a JFrame provide?*
- 10.6 *What does the method setVisible do?*
- 10.7 *How is an event represented?*
- 10.8 *What is a listener?*
- 10.9 *Why are listeners implemented as objects?*
- 10.10 *What is an anonymous class?*
- 10.11 *How is a menu constructed?*
- 10.12 *What does a JTextField component do?*
- 10.13 *What does paint do?*
- 10.14 *How is a GUI program terminated?*
- 10.15 *Why are GUI programs multi-threaded?*

Programming Exercises

10.1 Draw a pencil and paper picture illustrating how the GUI of the Convert program in Section 10.3, page 328 is laid out.

10.2 Implement a program with this interface:



Clicking the buttons increments or decrements the displayed counter value.

Hint: How do you position the two buttons? You might try this: add a panel, using a `FlowLayout`, to hold the two buttons.

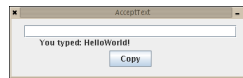
```
import java.awt.GridLayout ;
import java.awt.event.ActionEvent ;
import java.awt.event.ActionListener ;
import javax.swing.BorderFactory ;
import javax.swing.JButton ;
import javax.swing.JFrame ;
import javax.swing.JLabel ;
import javax.swing.JPanel ;
import javax.swing.SwingUtilities ;
/**
 * A counting application using Swing that keeps and displays a counter. There are two buttons,
 * one for increment the counter and one for decrementing the counter.
 *
 * @author Russel Winder
 * @version 2004.12.13
 */
public class Counter extends JFrame {
    private int counter = 0 ;
    private Counter() {
        super("Counter") ;
        final String leader = " Count is : " ;
        final JLabel label = new JLabel (leader + counter) ;
        final JButton incrementButton = new JButton ("Increment") ;
        incrementButton.addActionListener(new ActionListener () {
            public void actionPerformed(final ActionEvent ae) {
                ++counter ;
                label.setText(leader + counter) ;
            }
        }) ;
        final JButton decrementButton = new JButton ("Decrement") ;
        decrementButton.addActionListener(new ActionListener () {
            public void actionPerformed(final ActionEvent ae) {
```

```

        --counter ;
        label.setText(leader + counter) ;
    }
    });
    final JPanel buttonPanel = new JPanel () ;
    buttonPanel.add(incrementButton) ;
    buttonPanel.add(decrementButton) ;
    final JPanel panel = new JPanel (new GridLayout (2, 1)) ;
    panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
    panel.add(label) ;
    panel.add(buttonPanel) ;
    add(panel) ;
    pack() ;
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setVisible(true) ;
}
public static void main(final String[] args) {
    SwingUtilities.invokeLater(new Runnable () {
        public void run() {
            new Counter () ;
        }
    });
}
}
}

```

10.3 Write a Swing program to display this window:



When the button is clicked, the text typed into the JTextField at the top of the window is copied into the label in the middle of the window. Note the position and size of the button.

```

import java.awt.BorderLayout ;
import java.awt.event.ActionEvent ;
import java.awt.event.ActionListener ;
import javax.swing.BorderFactory ;
import javax.swing.JButton ;
import javax.swing.JFrame ;
import javax.swing.JLabel ;
import javax.swing.JPanel ;
import javax.swing.JTextField ;
import javax.swing.SwingUtilities ;
/**
 * A somewhat trivial application that copies text from a text entry box to the background of the
 * window when a button is pressed.
 *
 * @author Russel Winder
 * @version 2004.12.17
 */
public class AcceptText extends JFrame {
    private AcceptText() {

```

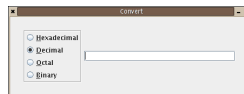
```

super("AcceptText");
final JTextField textField = new JTextField (30);
final String leader = " You typed: ";
final JLabel label = new JLabel (leader);
final JButton button = new JButton ("Copy");
button.addActionListener(new ActionListener () {
    public void actionPerformed(final ActionEvent ae) {
        label.setText(leader + textField.getText());
        textField.setText("");
        textField.requestFocusInWindow();
    }
});
final JPanel buttonPanel = new JPanel ();
buttonPanel.add(button);
final JPanel panel = new JPanel (new BorderLayout ());
panel.setBorder(BorderFactory.createEmptyBorder(10,20,10,20));
panel.add(textField, BorderLayout.NORTH);
panel.add(label, BorderLayout.CENTER);
panel.add(buttonPanel, BorderLayout.SOUTH);
add(panel);
pack();
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
textField.requestFocusInWindow();
}
public static void main(final String[] args) {
    SwingUtilities.invokeLater(new Runnable () {
        public void run() {
            new AcceptText ();
        }
    });
}
}

```

10.4 Modify the first version of the Very Simple Editor so that the buttons remain a fixed size when the window is made larger.

10.5 Implement a Swing program with this interface:



The buttons on the left are radio buttons (JRadioButton objects—see the JDK documentation for more information). When a number is entered in the JTextField on the right, and a radio button clicked, the number is converted to the new number base and displayed in place of the original. Note, only digits valid in the current number base can be typed into the JTextField.

Hint: Use an EtchedBorder on the panel holding the radio buttons, the BorderLayout class will create one for you. The radio buttons need to be in a ButtonGroup. A JTextField can generate a KeyEvent when a character is typed and accepts KeyListeners. Class Integer will do all the work of number base conversion.

```

import java.awt.GridLayout ;
import java.awt.event.ActionEvent ;
import java.awt.event.ActionListener ;
import java.awt.event.KeyAdapter ;
import java.awt.event.KeyEvent ;
import javax.swing.BorderFactory ;
import javax.swing.ButtonGroup ;
import javax.swing.JButton ;
import javax.swing.JFrame ;
import javax.swing.JLabel ;
import javax.swing.JOptionPane ;
import javax.swing.JPanel ;
import javax.swing.JRadioButton ;
import javax.swing.JTextField ;
import javax.swing.SwingUtilities ;
import javax.swing.border.EtchedBorder ;
/**
 * An application that accepts sequences of digits in a given radix and then performs conversion
 * from one radix to another.
 *
 * @author Russel Winder
 * @version 2004.12.17
 */
public class RadixConvert extends JFrame {
    private static enum Radix {
        HEXADECIMAL ( 16 ) ,
        DECIMAL ( 10 ) ,
        OCTAL ( 8 ) ,
        BINARY ( 2 ) ;
        private final static String hexadecimalDigits = "0123456789abcdefABCDEF" ;
        private final static String decimalDigits = "0123456789" ;
        private final static String octalDigits = "01234567" ;
        private final static String binaryDigits = "01" ;
        private final int radix ;
        private final String validDigits ;
        Radix ( final int radix ) {
            this.radix = radix ;
            switch ( radix ) {
                case 2 : this.validDigits = binaryDigits ; break ;
                case 8 : this.validDigits = octalDigits ; break ;
                case 10 : this.validDigits = decimalDigits ; break ;
                case 16 : this.validDigits = hexadecimalDigits ; break ;
                default : throw new RuntimeException ( "This cannot happen" ) ;
            }
        }
        public int value ( ) { return radix ; }
        public boolean isValidDigit ( final char c ) { return validDigits.indexOf(c) >= 0 ; }
    }
    private Radix radix = Radix.DECIMAL ;
    private final JTextField textField = new JTextField ( 25 ) ; {
        textField.addKeyListener ( new KeyAdapter ( ) {
            public void keyTyped ( final KeyEvent ke ) {
                char c = ke.getKeyChar ( ) ;
                if ( ( c == KeyEvent.VK_BACK_SPACE ) || ( c == KeyEvent.VK_DELETE ) ) { return ; }
                if ( ! radix.isValidDigit ( c ) ) { JOptionPane.showMessageDialog ( RadixConvert.this , "Character " + c + " is not a valid digit" ) ; }
            }
        }
    }
}

```

```

    });
}
private RadixConvert () {
    super ( "Radix Convert" );
    ButtonGroup buttonGroup = new ButtonGroup ();
    JPanel buttonPanel = new JPanel ( new GridLayout ( 4 , 1 ) );
    buttonPanel.setBorder ( new EtchedBorder () );
    JRadioButton button = new JRadioButton ( "Hexadecimal" );
    button.setMnemonic ( KeyEvent.VK_H );
    button.addActionListener ( new ActionListener () {
        public void actionPerformed ( final ActionEvent ae ) { selectRadix ( Radix.HEXADECIMAL ); }
    });
    buttonGroup.add ( button );
    buttonPanel.add ( button );
    button = new JRadioButton ( "Decimal" );
    button.setMnemonic ( KeyEvent.VK_D );
    button.setSelected ( true );
    button.addActionListener ( new ActionListener () {
        public void actionPerformed ( final ActionEvent ae ) { selectRadix ( Radix.DECIMAL ); }
    });
    buttonGroup.add ( button );
    buttonPanel.add ( button );
    button = new JRadioButton ( "Octal" );
    button.setMnemonic ( KeyEvent.VK_O );
    button.addActionListener ( new ActionListener () {
        public void actionPerformed ( final ActionEvent ae ) { selectRadix ( Radix.OCTAL ); }
    });
    buttonGroup.add ( button );
    buttonPanel.add ( button );
    button = new JRadioButton ( "Binary" );
    button.setMnemonic ( KeyEvent.VK_B );
    button.addActionListener ( new ActionListener () {
        public void actionPerformed ( final ActionEvent ae ) { selectRadix ( Radix.BINARY ); }
    });
    buttonGroup.add ( button );
    buttonPanel.add ( button );
    final JPanel panel = new JPanel ();
    panel.setBorder ( BorderFactory.createEmptyBorder ( 20 , 20 , 20 , 20 ) );
    panel.add ( buttonPanel );
    panel.add ( textField );
    add ( panel );
    pack ();
    setDefaultCloseOperation ( JFrame.EXIT_ON_CLOSE );
    setVisible ( true );
    textField.requestFocusInWindow ();
}
private void selectRadix ( final Radix r ) {
    String s = textField.getText ();
    if ( ! s.equals ( "" ) ) {
        long value = Long.parseLong ( s , radix.value () );
        textField.setText ( Long.toString ( value , r.value () ) );
    }
    radix = r ;
    textField.requestFocusInWindow ();
}
public static void main ( final String[] args ) {

```

```
SwingUtilities.invokeLater ( new Runnable () {  
    public void run () { new RadixConvert (); }  
});  
}  
}
```

- 10.6** Repeat the last question but replace the radio buttons with a `JComboBox`.
- 10.7** Rewrite the palindrome program presented in Section 3.4.2, page 69, so that it has a GUI.

Challenges

- 10.1** Implement a calculator program based on the GUI shown in Figure 10.1, page 314.
- 10.2** Extend the graph program shown in Section 6.8.3, page 201 to include a complete GUI with a menu bar. It should be possible to select and load a data file, and select and configure the kind of graph to be displayed (support at least bar graphs, line graphs and pie charts).
- 10.3** Modify the second version of the Very Simple Editor program with the menu bar (Section 10.5, page 334), so that it uses a `JTextPane` rather than a `JTextArea` to display and edit text. Extend the program to provide support for different fonts, and bold and italic text.