JAVA NOTES

GRAPHICAL USER INTERFACES

Terry Marris July 2001

6 CHECK BOXES

6.1 LEARNING OUTCOMES

By the end of this lesson the student should be able to

- set the font, style and size of text
- understand and use check boxes

6.2 INTRODUCTION

Check boxes collect yes-no responses from users. You check (tick) a box for yes, leave it blank for no. Figure 6.1 shows a set of text boxes (labelled *Standard*, *Formatting*, *Tables*, *Borders*, ...) for customising the format and appearance of the Microsoft Word menu system.

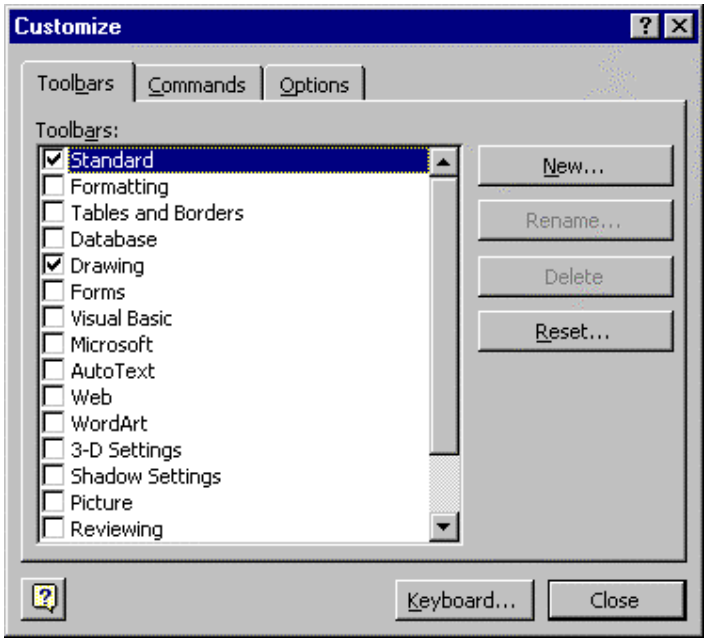


Figure 6.1 *Check Boxes*

A *JCheckBox* component comes with its own label. We usually arrange check box components in a vertical list.

What we are aiming to produce is shown in figure 6.2 below.

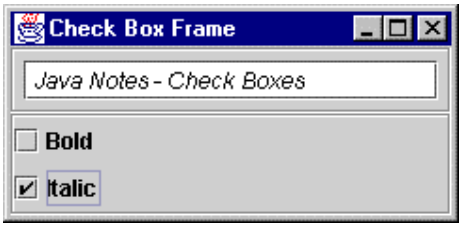


Figure 6.2 *Check Box Frame with Two Check Boxes*

The user may leave both *Bold* and *Italic* unchecked, in which case the text *Java Notes - Check Boxes* is plain - neither emboldened not italicised.

The user may chose the text to appear either bold, or italic, or both bold and italic.

So first we need to look at fonts.

6.3 FONTS

The text in these lesson notes are written in the Times New Roman font, size 12 point, with faces such as bold (for headings) and italic (for important words); the Java program code is written in Courier New font.

The *Font* class represents fonts.

A font has a name such as *Dialog*, *DialogInput*, *Monospaced*, *Serif*, *SansSerif* and *Symbol*.

A font has a size e.g. 12 point. A point is 1/72 of an inch. So a character displayed in a size of 72 points is one inch high.

java.awt.Font

Constructor	
Font(String name, int style, int size)	Initialises a new font object with a name from *Dialog*, *DialogInput*, *Monospaced*, *Serif*, *SansSerif* and *Symbol*, a style from *Font.PLAIN*, *Font.BOLD*, *Font.ITALIC* and *Font.BOLD+Font.ITALIC* a size in points

6.4 CHECK BOXES

When creating a new *JCheckBox* instance we supply the label as an argument value to the constructor.

```
JCheckBox boldCheckBox = new JCheckBox("Bold");
```

Since we want to arrange the check boxes in a vertical list in a panel we specify border layout and then add the check boxes to the north and south border.

```
setLayout(new BorderLayout());

boldCheckBox = new JCheckBox("Bold");
add(boldCheckBox, "North");

italicCheckBox = new JCheckBox("Italic");
add(italicCheckBox, "South");
```

If we had more than three check boxes we could either nest panels, each containing up to three text boxes, or use a different layout manager - see §9 Layout Managers.

We want to know whether the user has checked a box. *isSelected()* returns either *true* or *false*.

```
if (checkBoxPanel.getBoldCheckBox().isSelected())
```

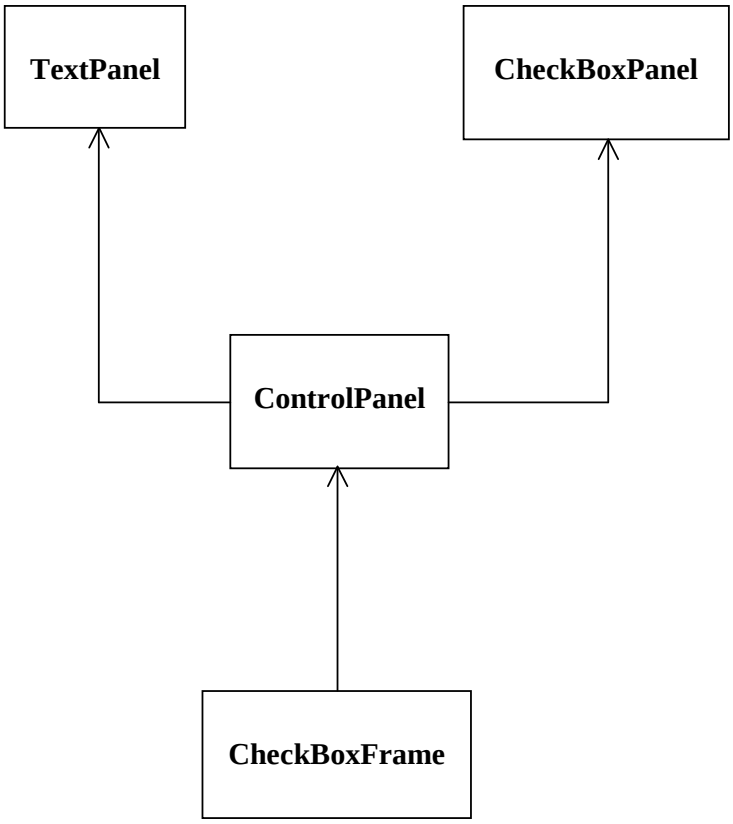
Then we want to set the text to the format requested by the user.

```
int m = 0;
if (checkBoxPanel.getBoldCheckBox().isSelected())
  m = Font.BOLD;

if (checkBoxPanel.getItalicCheckBox().isSelected())
  m += Font.ITALIC;

textPanel.setTextFont(m);
```

If the user selects both bold and italic, *m* contains the int value of *Font.BOLD + Font.ITALIC*.

The entire program is shown below.

6.5 CLASS DIAGRAM



A check box frame has a control panel. A control panel has a text panel and a check box panel.

6.6 CHECK BOX FRAME

```
/* CheckBoxFrame.java
   Terry Marris 25 June 2001
*/

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;


class TextPanel extends JPanel {
  private JTextField text;

  public TextPanel()
  {
    Border etched = BorderFactory.createEtchedBorder();
    setBorder(etched);

    text = new JTextField(" Java Notes - Check Boxes ", 20);
    add(text);
  }


  public void setTextFont(int font)
  {
    text.setFont(new Font("SansSerif", font, 12));
    repaint();
  }
}
```

Exercise: Explain each line of the *TextPanel* class shown above.

```
class CheckBoxPanel extends JPanel {
  private JCheckBox boldCheckBox;
  private JCheckBox italicCheckBox;

  public CheckBoxPanel()
  {
    Border etched = BorderFactory.createEtchedBorder();
    setBorder(etched);

    setLayout(new BorderLayout());

    boldCheckBox = new JCheckBox("Bold");
    add(boldCheckBox, "North");

    italicCheckBox = new JCheckBox("Italic");
    add(italicCheckBox, "South");
  }


  public JCheckBox getBoldCheckBox()
  {
    return boldCheckBox;
  }


  public JCheckBox getItalicCheckBox()
  {
    return italicCheckBox;
  }
}
```

Exercise: Explain each line of the *CheckBoxPanel* class shown above.

```
class ControlPanel extends JPanel implements ActionListener {
  private TextPanel textPanel;
  private CheckBoxPanel checkBoxPanel;

  public ControlPanel()
  {
    setLayout(new BorderLayout());

    textPanel = new TextPanel();
    add(textPanel, "North");

    checkBoxPanel = new CheckBoxPanel();
    add(checkBoxPanel, "South");
    checkBoxPanel.getBoldCheckBox().addActionListener(this);
    checkBoxPanel.getItalicCheckBox().addActionListener(this);
  }


  public void actionPerformed(ActionEvent e)
  {
    int m = 0;
    if (checkBoxPanel.getBoldCheckBox().isSelected())
      m = Font.BOLD;

    if (checkBoxPanel.getItalicCheckBox().isSelected())
      m += Font.ITALIC;

    textPanel.setTextFont(m);
  }
}
```

Exercise: Explain each line of the *ControlPanel* class shown above.

```
public class CheckBoxFrame extends JFrame {
  public CheckBoxFrame()
  {
    setTitle("Check Box Frame");

    Toolkit tk = Toolkit.getDefaultToolkit();
    Dimension screen = tk.getScreenSize();
    setLocation(screen.width/4, screen.height/4);

    addWindowListener(new WindowAdapter()
      {
        public void windowClosing(WindowEvent e)
        {
          System.exit(0);
        }
      });

    Container contentPane = getContentPane();
    contentPane.add(new ControlPanel());

    pack();
  }


  public static void main(String[] s)
  {
    JFrame aFrame = new CheckBoxFrame();
    aFrame.show();
  }
}
```

Exercise: Explain each line of the *CheckBoxFrame* class shown above.

6.7 DOCUMENTATION

javax.swing.JCheckBox

Constructors	
JCheckBox(String text)	Initialises a newly created unselected JCheckBox with the given text as its label.
JCheckBox(String text, boolean isSelected)	Initialises a newly created JCheckBox with the given text as its label and specifies whether or not is initially selected. isSelected state.

AbstractButton implements behaviour that is common to *JButton, JToggleButton, JCheckBox* and *JRadioButton* classes.

javax.swing.AbstractButton

Methods		
void	addActionListener(ActionListener l)	adds an action listener to this button.
void	doClick()	programmatically perform a click, just as if the user had clicked the button.
String	getActionCommand()	returns the action command for this button.
String	getText()	returns the button's label.
boolean	isSelected()	returns the state of the button, true if selected, false otherwise.
void	setActionCommand(String actionCommand)	sets the action command for this button.
void	setVerticalTextPosition(int textPosition)	sets the position of the label CENTER (default) TOP or BOTTOM.
void	setHorizontalTextPosition(int textPosition)	sets the horizontal position of the label RIGHT (default) LEFT, CENTER, LEADING, TRAILING.
void	setEnabled(boolean b)	enables (or disables) this button.
void	setSelected(boolean b)	sets the state of the button. Need to call doClick() to make it effective.
void	setText(String text)	sets this button's label.
void	setMnemonic(char mnemonic)	sets the character specifying the mnemonic value.

6.8 FURTHER READING

HORSTMANN & CORNELL *Core Java 2 Volume 1* pp 408
www.java.sun.com/docs/books/tutorial

In the next lesson we look at radio buttons.

6.9 REVIEW

6.10 EXERCISES

1 Try out program *CheckBoxFrame.java* shown in §6.5 above.