```
//THE IMPORTED LIBRARIES
import javax.swing.*;
import javax.swing.undo.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.event.*;
import java.util.*;
import javax.swing.text.*;
import java.text.*;


public class myedit extends JFrame implements ActionListener
{
	public static myedit e;

	//DECLARATION OF ALL THE VARIABLES USED IN THIS APPLICATION

	JTextArea text = new JTextArea(0,0);
	JScrollPane scroll = new JScrollPane(text);

	JMenuBar mb=new JMenuBar();

	JMenu FILE = new JMenu("File");
	JMenu EDIT = new JMenu("Edit");
	JMenu SEARCH = new JMenu("Search");
	JMenu HELP = new JMenu("Help");

	JMenuItem NEWFILE = new JMenuItem("New");
	JMenuItem OPENFILE = new JMenuItem("Open...");
	JMenuItem SAVEFILE = new JMenuItem("Save");
JMenuItem SAVEASFILE = new JMenuItem("Save As...");
JMenuItem EXITFILE = new JMenuItem("Exit");



JMenuItem CUTEDIT = new JMenuItem("Cut");
JMenuItem COPYEDIT = new JMenuItem("Copy");
JMenuItem PASTEDIT = new JMenuItem("Paste");
JMenuItem DELETEDIT = new JMenuItem("Delete");
JMenuItem SELECTEDIT = new JMenuItem("Select All");
JMenuItem TIMEDIT = new JMenuItem("Time/Date");
JCheckBoxMenuItem WORDEDIT = new JCheckBoxMenuItem("Word Wrap");
JMenuItem FONTEDIT = new JMenuItem("Set Font...");

JMenuItem FINDSEARCH = new JMenuItem("Find");
JMenuItem FINDNEXTSEARCH = new JMenuItem("Find Next");

JMenuItem ABOUTHELP = new JMenuItem("About");

JPopupMenu POPUP = new JPopupMenu();
JMenuItem CUTPOPUP = new JMenuItem("Cut");
JMenuItem COPYPOPUP = new JMenuItem("Copy");
JMenuItem PASTEPOPUP = new JMenuItem("Paste");
JMenuItem DELETEPOPUP = new JMenuItem("Delete");
JMenuItem SELECTPOPUP = new JMenuItem("Select All");



UndoManager undo = new UndoManager();
UndoAction undoAction = new UndoAction();

boolean opened = false;
String wholeText,findString,filename = null;
```

```
int ind = 0;

//CLASS FOR HANDLING UNDO MENU OPTION OF EDIT AND POPUP MENU
class UndoAction extends AbstractAction
{
	public UndoAction()
	{
		super("Undo");
	}

	public void actionPerformed(ActionEvent e)
	{
		try
		{
			undo.undo();
		}
		catch (CannotUndoException ex)
		{
			System.out.println("Unable to undo: " + ex);
			ex.printStackTrace();
		}
		update();
	}

	protected void update()
	{
		if(undo.canUndo())
		{
			setEnabled(true);
			putValue("Undo", undo.getUndoPresentationName());
		}
```

```
                else
                {
                        setEnabled(false);
                        putValue(Action.NAME, "Undo");
                }
        }
}

//DEFAULT CONSTRUCTOR OF THE MYEDIT CLASS
public myedit()
{
        //SETING DEFAULT TITLE OF THE FRAME
        setTitle("Untitled");

        //SETTING DEFAULT SIZE OF THE FRAME
        setSize(600,400);

        //MAKING THE FRAME VISIBLE
        setVisible(true);

        //SETTING WORD WRAP TO TRUE AS DEFAULT
        text.setLineWrap(true);

        //SETTING THE DEFAULT STATE OF WORDWRAP MENU OPTION IN EDIT MENU
        WORDEDIT.setState(true);

        //SETTING THE LAYOUT OF THE FRAME
        getContentPane().setLayout(new BorderLayout());

        //ADDS THE SCROLLPANE CONTAINING THE TEXTAREA TO THE CONTAINER
        getContentPane().add(scroll, BorderLayout.CENTER);
```

```
//ADDING THE MAIN MENUBAR TO THE FRAME
setJMenuBar(mb);

//ADDING MENUS TO THE MAIN MENUBAR
mb.add(FILE);
mb.add(EDIT);
mb.add(SEARCH);
mb.add(HELP);

//ADDING MENUITEMS TO THE FILE MENU
FILE.add(NEWFILE);
FILE.add(OPENFILE);
FILE.add(SAVEFILE);
FILE.add(SAVEASFILE);
FILE.addSeparator();
FILE.add(EXITFILE);

//ADDING MENUITEMS TO THE EDIT MENU
EDIT.add(undoAction);
EDIT.add(CUTEDIT);
EDIT.add(COPYEDIT);
EDIT.add(PASTEDIT);
EDIT.add(DELETEDIT);
EDIT.addSeparator();
EDIT.add(SELECTEDIT);
EDIT.add(TIMEDIT);
EDIT.addSeparator();
EDIT.add(WORDEDIT);
EDIT.add(FONTEDIT);
```

```
//ADDING MENUITEMS TO THE SEARCH MENU
SEARCH.add(FINDSEARCH);
SEARCH.add(FINDNEXTSEARCH);

//ADDING MENUITEM TO THE HELP MENU
HELP.add(ABOUTHELP);

//ADDING MENUITEMS TO THE POPUPMENU
POPUP.add(undoAction);
POPUP.addSeparator();
POPUP.add(CUTPOPUP);
POPUP.add(COPYPOPUP);
POPUP.add(PASTEPOPUP);
POPUP.add(DELETEPOPUP);
POPUP.addSeparator();
POPUP.add(SELECTPOPUP);

//SETTING SHORTCUT KEYS OF MENUS IN THE MAIN MENUBAR
FILE.setMnemonic(KeyEvent.VK_F);
EDIT.setMnemonic(KeyEvent.VK_E);
SEARCH.setMnemonic(KeyEvent.VK_S);
HELP.setMnemonic(KeyEvent.VK_H);

//SETTING SHORTCUT KEYS OF MENUITEMS IN THE FILE MENU
NEWFILE.setMnemonic(KeyEvent.VK_N);
OPENFILE.setMnemonic(KeyEvent.VK_O);
SAVEFILE.setMnemonic(KeyEvent.VK_S);
SAVEASFILE.setMnemonic(KeyEvent.VK_A);
EXITFILE.setMnemonic(KeyEvent.VK_X);

//SETTING SHORTCUT KEYS OF MENUITEMS IN THE EDIT MENU
```

```
        CUTEDIT.setMnemonic(KeyEvent.VK_T);
        COPYEDIT.setMnemonic(KeyEvent.VK_C);
        PASTEDIT.setMnemonic(KeyEvent.VK_P);
        DELETEDIT.setMnemonic(KeyEvent.VK_L);
        SELECTEDIT.setMnemonic(KeyEvent.VK_A);
        TIMEDIT.setMnemonic(KeyEvent.VK_D);
        WORDEDIT.setMnemonic(KeyEvent.VK_W);
        FONTEDIT.setMnemonic(KeyEvent.VK_F);

        //SETTING SHORTCUT KEYS OF MENUITEMS IN THE SEARCH MENU
        FINDSEARCH.setMnemonic(KeyEvent.VK_F);
        FINDNEXTSEARCH.setMnemonic(KeyEvent.VK_N);

        //SETTING SHORTCUT KEYS OF MENUITEM IN THE HELP MENU
        ABOUTHELP.setMnemonic(KeyEvent.VK_A);

        //SETTING SHORTCUT KEYS OF MENUITEMS IN THE POPUPMENU
        CUTPOPUP.setMnemonic(KeyEvent.VK_T);
        COPYPOPUP.setMnemonic(KeyEvent.VK_C);
        PASTEPOPUP.setMnemonic(KeyEvent.VK_P);
        DELETEPOPUP.setMnemonic(KeyEvent.VK_D);
        SELECTPOPUP.setMnemonic(KeyEvent.VK_A);

        //SETTING ACCELERATOR KEYS OF SOME MENUITEMS IN THE EDIT MENU

    CUTEDIT.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X,ActionEvent.CTRL_MASK));

    COPYEDIT.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C,ActionEvent.CTRL_MASK));

    PASTEDIT.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V,ActionEvent.CTRL_MASK));

        //ADDING LISTENERS TO THE MENUITEMS IN FILE MENU
```

```
NEWFILE.addActionListener(this);
OPENFILE.addActionListener(this);
SAVEFILE.addActionListener(this);
SAVEASFILE.addActionListener(this);
EXITFILE.addActionListener(this);

//ADDING LISTENERS TO THE MENUITEMS IN EDIT MENU
text.getDocument().addUndoableEditListener(new UndoListener());
CUTEDIT.addActionListener(this);
COPYEDIT.addActionListener(this);
PASTEDIT.addActionListener(this);
DELETEDIT.addActionListener(this);
SELECTEDIT.addActionListener(this);
TIMEDIT.addActionListener(this);
WORDEDIT.addActionListener(this);
FONTEDIT.addActionListener(this);

//ADDING LISTENERS TO THE MENUITEMS IN SEARCH MENU
FINDSEARCH.addActionListener(this);
FINDNEXTSEARCH.addActionListener(this);

//ADDING LISTENERS TO THE MENUITEM IN HELP MENU
ABOUTHELP.addActionListener(this);

//ADDING LISTENERS TO THE MENUITEMS IN POPUPMENU
CUTPOPUP.addActionListener(this);
COPYPOPUP.addActionListener(this);
PASTEPOPUP.addActionListener(this);
DELETEPOPUP.addActionListener(this);
SELECTPOPUP.addActionListener(this);
```

```
        //ADDING MOUSELISTENER TO RIGHT CLICK FOR THE POPUPLISTENER
        text.addMouseListener(new MouseAdapter()
        {
          public void mousePressed(MouseEvent e)
          {
                    if (e.isPopupTrigger())
                    {
                POPUP.show(e.getComponent(),e.getX(), e.getY());
          }

          }
           public void mouseReleased(MouseEvent e)
           {
                        if (e.isPopupTrigger())
                        {
                        POPUP.show(e.getComponent(),e.getX(), e.getY());
                }
           }
        });

        //ADDING WINDOWLISTENER TO HANDLE CLOSE WINDOW EVENT

    /*  addWindowListener( new WindowAdapter()
        { public void windowClosing(WindowEvent evt)
          {
                int response = JOptionPane.showConfirmDialog(null, "Do you really want to
quit?");
                System.out.println("Inside Window Listener");
                switch (response)
                {
                        case 0:
```

```
                                        {
                                        dispose();
                                        break; }

                                case 2:
                                {
                                        //myedit x = new myedit();
                                        System.out.println("Inside 2");
                                        e=new myedit();
                                        e.setVisible(true);
                                        break;}
                         }
                        System.out.println("response is :="+response);
                }
                } ); */




                addWindowListener(new WindowAdapter()
                {
                        public void windowClosing(WindowEvent e)
                        {
                                exitApln();
                        }
                });
        }

        //HANDLING ALL EVENTS OF THE TEXT EDITOR
        public void actionPerformed(ActionEvent e)
        {
```

```
//ACTION FOR NEW MENU OPTION OF FILE MENU
if (e.getSource()==NEWFILE)
{
        newfile();
}

//ACTION FOR OPEN MENU OPTION OF FILE MENU
if (e.getSource()==OPENFILE)
{
        open();
}

//ACTION FOR SAVE MENU OPTION OF FILE MENU
if (e.getSource()==SAVEFILE)
{
        save();
}

//ACTION FOR SAVEAS MENU OPTION OF FILE MENU
if (e.getSource()==SAVEASFILE)
{
        opened=false;
        save();
}

//ACTION FOR EXIT MENU OPTION OF FILE MENU
if (e.getSource()==EXITFILE)
{
        exitApln();
}
```

```
//ACTION FOR CUT MENU OPTION OF EDIT MENU AND POPUPMENU
if ((e.getSource()==CUTEDIT)||(e.getSource()==CUTPOPUP))
{
        text.cut();
}

//ACTION FOR COPY MENU OPTION OF EDIT MENU AND POPUPMENU
if ((e.getSource()==COPYEDIT)||(e.getSource()==COPYPOPUP))
{
        text.copy();
}

//ACTION FOR PASTE MENU OPTION OF EDIT MENU AND POPUPMENU
if ((e.getSource()==PASTEDIT)||(e.getSource()==PASTEPOPUP))
{
        text.paste();
}

//ACTION FOR DELETE MENU OPTION OF EDIT MENU AND POPUPMENU
if ((e.getSource()==DELETEDIT)||(e.getSource()==DELETEPOPUP))
{
        text.replaceSelection(null);
}

//ACTION FOR SELECTALL MENU OPTION OF EDIT MENU AND POPUPMENU
if ((e.getSource()==SELECTEDIT)||(e.getSource()==SELECTPOPUP))
{
        text.selectAll();
}
```

```
		//ACTION FOR TIME/DATE MENU OPTION OF EDIT MENU
		if (e.getSource()==TIMEDIT)
		{
			Date currDate;
			String dd;
			currDate = new java.util.Date();
			dd=currDate.toString();
			text.insert(dd,text.getCaretPosition());
		}

		//ACTION FOR WORD WRAP MENU OPTION OF EDIT MENU
		if (e.getSource()==WORDEDIT)
		{
			if(WORDEDIT.isSelected())
			text.setLineWrap(true);
			else
			text.setLineWrap(false);
		}

		//ACTION FOR SET FONT MENU OPTION OF EDIT MENU
		if (e.getSource()==FONTEDIT)
		{
			fontDialogBox fontS = new fontDialogBox();
		}

		//ACTION FOR FIND MENU OPTION OF SEARCH MENU
		if (e.getSource()==FINDSEARCH)
		{
			wholeText=text.getText();
			findString =JOptionPane.showInputDialog(null, "Find What", "Find",
JOptionPane.INFORMATION_MESSAGE);
```

```
                ind = wholeText.indexOf(findString,0);
                text.setCaretPosition(ind);
                text.setSelectionStart(ind);
                text.setSelectionEnd(ind+findString.length());
            }

            //ACTION FOR FIND NEXT MENU OPTION OF SEARCH MENU
            if (e.getSource()==FINDNEXTSEARCH)
            {
                wholeText= text.getText();
                findString = JOptionPane.showInputDialog(null, "Find What","Find Next",
JOptionPane.INFORMATION_MESSAGE);
                ind = wholeText.indexOf(findString, text.getCaretPosition());
                text.setCaretPosition(ind);
                text.setSelectionStart(ind);
                text.setSelectionEnd(ind+findString.length());
            }

            //ACTION FOR ABOUT MENU OPTION OF HELP MENU
            if (e.getSource()==ABOUTHELP)
            {
                JOptionPane.showMessageDialog(null, "This is a simple Text Editor
application built using Java.",
                "About Editor",
                JOptionPane.INFORMATION_MESSAGE);
            }
        }

        //ACTION FOR NEW MENU OPTION OF FILE MENU
        public void newfile()
```

```
{
        if(!text.getText().equals(""))
        {
                opened=false;
                int confirm = JOptionPane.showConfirmDialog(null,
                "Text in the Untitled file has changed. \n Do you want to save the changes?",
                "New File",
JOptionPane.YES_NO_CANCEL_OPTION,JOptionPane.INFORMATION_MESSAGE);

                if( confirm == JOptionPane.YES_OPTION )
                {
                        save();
                        text.setText(null);
                }
                else if( confirm == JOptionPane.NO_OPTION )
                {
                        text.setText(null);
                }
        }
}

//ACTION FOR OPEN MENU OPTION OF FILE MENU
public void open()
{
        text.setText(null);
        JFileChooser ch = new JFileChooser();
        ch.setCurrentDirectory(new File("."));
        ch.setFileFilter(new javax.swing.filechooser.FileFilter()
        {
        public boolean accept(File f)
        {
                return f.isDirectory()
                        || f.getName().toLowerCase().endsWith(".java");
        }
                public String getDescription()
        {
              return "Java files";
        }
});
        int result = ch.showOpenDialog(new JPanel());
        if(result == JFileChooser.APPROVE_OPTION)
        {
                filename = String.valueOf(ch.getSelectedFile());
                setTitle(filename);
                opened=true;
                FileReader fr;
                BufferedReader br;

                try
                {
                        fr=new FileReader (filename);
                        br=new BufferedReader(fr);
                        String s;
                        while((s=br.readLine())!=null)
                        {
                                text.append(s);
                                text.append("\n");
                        }
                        fr.close();
                }
                catch(FileNotFoundException ex)
```

```
                    {
                                        JOptionPane.showMessageDialog(this, "Requested file not found",
"Error Dialog box", JOptionPane.ERROR_MESSAGE);}


                  catch(Exception ex)
                    {System.out.println(ex);}
                     }
          }


          //ACTION FOR SAVE MENU OPTION OF FILE MENU
          public void save()
          {
                    if(opened==true)
                    {
                              try
                              {
                                        FileWriter f1 = new FileWriter(filename);
                                        f1.write(text.getText());
                                        f1.close();
                                        opened = true;
                              }
                              catch(FileNotFoundException ex)
                    {
                                        JOptionPane.showMessageDialog(this, "Requested file not found",
"Error Dialog box", JOptionPane.ERROR_MESSAGE);}
                              catch(IOException ioe){ioe.printStackTrace();}
                    }
                    else
                    {
                              JFileChooser fc = new JFileChooser();
                              fc.setCurrentDirectory(new File("."));
                              int result = fc.showSaveDialog(new JPanel());
```

```
                                if(result == JFileChooser.APPROVE_OPTION)
                                {
                                        filename = String.valueOf(fc.getSelectedFile());
                                        setTitle(filename);
                                        try
                                        {
                                        FileWriter f1 = new FileWriter(filename);
                                        f1.write(text.getText());
                                        f1.close();
                                        opened = true;
                                        }
                                catch(FileNotFoundException ex)
                        {
                                        JOptionPane.showMessageDialog(this, "Requested file not found",
"Error Dialog box", JOptionPane.ERROR_MESSAGE);}

                                        catch(IOException ioe){ioe.printStackTrace();}
                                }

                        }
                }

                //ACTION FOR EXIT MENU OPTION OF FILE MENU AND CLOSE WINDOW BUTTON
                public void exitApln()
                {
                        if(!text.getText().equals(""))
                        {
                                int confirm = JOptionPane.showConfirmDialog(null,
                                "Text in the file has changed. \n Do you want to save the changes?",
                                "Exit",
JOptionPane.YES_NO_CANCEL_OPTION,JOptionPane.INFORMATION_MESSAGE);

                    if( confirm == JOptionPane.YES_OPTION )
                    {
                              save();
                              dispose();
                              System.exit(0);
                    }
                    else if( confirm == JOptionPane.CANCEL_OPTION )
                    {
                              e=new myedit();
                              String s= text.getText();
                              e.setVisible(true);
                              e.text.setText(s);
                    }
                    else if( confirm == JOptionPane.NO_OPTION )
                    {
                              dispose();
                              System.exit(0);
                    }
          }
          else
          {
                    System.exit(0);
          }
}

//CLASS FOR UNDOLISTENER
class UndoListener implements UndoableEditListener
{
			public void undoableEditHappened(UndoableEditEvent e)
			{
			undo.addEdit(e.getEdit());
			undoAction.update();
			}
		}

		//CLASS FOR BUILDING AND DISPLAYING FONT DIALOG BOX
		class fontDialogBox extends JFrame implements ActionListener
		{
			//DECLARATION OF ALL VARIABLES USED IN fontDialogBox CLASS

			String availableFontString[] =
	GraphicsEnvironment.getLocalGraphicsEnvironment().getAvailableFontFamilyNames();
			JList fontList = new JList(availableFontString);
			JLabel fontLabel = new JLabel("Font");
			JTextField valueFont=new JTextField("Arial");
			JScrollPane fontPane = new JScrollPane(fontList);



			String fontStyleString[] = {"Normal","Bold","Italic","Bold Italic"};
			JList styleList = new JList(fontStyleString);
			JLabel styleLabel = new JLabel("Style");
			int v = ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS;
			int h = ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
			JScrollPane stylePane = new JScrollPane(styleList,v,h);
			JTextField valueStyle=new JTextField("Normal");

			String fontSizeString[] = {"8","10","12","14","16","18","20","22","24","28"};
			JList sizeList = new JList(fontSizeString);
			JLabel sizeLabel = new JLabel("Font size");
```

```
JScrollPane sizePane = new JScrollPane(sizeList);
JTextField valueSize=new JTextField("12");

JButton okButton = new JButton("OK");
JButton cancelButton = new JButton("Cancel");

JLabel sampleLabel = new JLabel("Sample:");
JTextField sample = new JTextField("   AaBbCc");

Font selectedFont;

//DEFAULT CONSTRUCTOR OF fontDialogBox CLASS
public fontDialogBox()
{
        setSize(500,300);
        setTitle("Font");
        setVisible(true);
        sample.setEditable(false);

        getContentPane().setLayout(null);

        fontLabel.setBounds(10,10,170,20);
        valueFont.setBounds(10,35,170,20);
        fontPane.setBounds(10,60,170,150);

        styleLabel.setBounds(200,10,100,20);
        valueStyle.setBounds(200,35,100,20);
        stylePane.setBounds(200,60,100,150);

        sizeLabel.setBounds(320,10,50,20);
        valueSize.setBounds(320,35,50,20);
```

```
sizePane.setBounds(320,60,50,150);

okButton.setBounds(400,35,80,20);
cancelButton.setBounds(400,60,80,20);

sampleLabel.setBounds(150,235,50,30);
sample.setBounds(200,235,100,30);

getContentPane().add(fontLabel);
getContentPane().add(fontPane);
getContentPane().add(valueFont);

getContentPane().add(styleLabel);
getContentPane().add(stylePane);
getContentPane().add(valueStyle);

getContentPane().add(sizeLabel);
getContentPane().add(sizePane);
getContentPane().add(valueSize);

getContentPane().add(okButton);
getContentPane().add(cancelButton);
getContentPane().add(sampleLabel);
getContentPane().add(sample);

okButton.addActionListener(this);
cancelButton.addActionListener(this);

fontList.addListSelectionListener(new ListSelectionListener()
{
        public void valueChanged(ListSelectionEvent event)
```

```
                {
                    if (!event.getValueIsAdjusting())
                    {
                    valueFont.setText(fontList.getSelectedValue().toString());
                    selectedFont = new
Font(valueFont.getText(),styleList.getSelectedIndex(),Integer.parseInt(valueSize.getText()));
                    sample.setFont(selectedFont);
                    }
                }
            });

            styleList.addListSelectionListener(new ListSelectionListener()
            {
                public void valueChanged(ListSelectionEvent event)
                {
                if (!event.getValueIsAdjusting())
                    {
                    valueStyle.setText(styleList.getSelectedValue().toString());
                    selectedFont = new
Font(valueFont.getText(),styleList.getSelectedIndex(),Integer.parseInt(valueSize.getText()));
                    sample.setFont(selectedFont);
                    }
                }
            });

            sizeList.addListSelectionListener(new ListSelectionListener()
            {
                public void valueChanged(ListSelectionEvent event)
                {
                    if (!event.getValueIsAdjusting())
                    {
```

```
		valueSize.setText(sizeList.getSelectedValue().toString());
									selectedFont = new
Font(valueFont.getText(),styleList.getSelectedIndex(),Integer.parseInt(valueSize.getText()));
									sample.setFont(selectedFont);
								}
							}
						});
				}//END OF DEFAULT CONSTRUCTOR OF fontdialogBox CLASS

				public void actionPerformed(ActionEvent ae)
				{
						if(ae.getSource()==okButton)
						{
								selectedFont = new
Font(valueFont.getText(),styleList.getSelectedIndex(),Integer.parseInt(valueSize.getText()));
								text.setFont(selectedFont);
								setVisible(false);
						}
						if(ae.getSource()==cancelButton)
						{
								setVisible(false);
						}
				}
		}// END OF fontDialogBox CLASS

		//MAIN FUNCTION OF MYEDIT CLASS
		public static void main(String args[])
		{
				e= new myedit();
		}
}//END OF MYEDIT CLASS
```