A Mini Project Report

On

Scratch Pad

Submitted in partial fulfillment of requirements for the Course
CSE18R272 - JAVA PROGRAMMING

Bachelor's of Technology

In

Computer Science and Engineering

Submitted By

T V S K LIKHIT

9918004116

M SAI DHANUNJAI

9918004070

Under the guidance of

Dr. R. RAMALAKSHMI

(Associate Professor)

Department of Computer Science and Engineering

Kalasalingam Academy of Research and Education

Anand Nagar, Krishnankoil-626126

APRIL 2020

ABSTRACT

The idea of our project is to design "Scratch Pad".A Scratch pad is simple text editor of Microsoft windows and a basic text editing program which enables computer users to create documents.it will take notes easily.In that we can add to-do lists. It is a desktop application which is implemented in java platform.It was first released as a mouse-based MS-DOS program in 1983, and has been included in all versions of Microsoft Windows since Windows 1.0 in 1985.

DECLARATION

I hereby declare that the work presented in this report entitled "**Scratch Pad**", in partial fulfilment of the requirements for the course CSE18R272-Java Programming and submitted in **Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education** (**Deemed to be University**) is an authentic record of our own work carried out during the period from **Jan 2020** under the guidance of Mr. **Dr. R. Ramalakshmi** (Associate Professor).

The work reported in this has not been submitted by me for the award of any other degree of this or any other institute.

T V S K LIKHIT

9918004116

M SAI DHANUNJAI

9918004070

ACKNOWLEDGEMENT

First and foremost, I wish to thank the **Almighty God** for his grace and benediction tocomplete this Project work successfully. I would like to convey my special thanks from the bottom of my heart to my dear **Parents** and affectionate **Family members** for their honest support for the completion of this Project work.

I express deep sense of gratitude to "Kalvivallal" Thiru. **T. Kalasalingam** B.com., Founder Chairman, "Ilayavallal" **Dr.K.Sridharan** Ph.D.,Chancellor, **Dr.S.ShasiAnand** , Ph.D.,Vice President (Academic) , **Mr.S.ArjunKalasalingam** M.S., Vice President (Administration) , **Dr.R.Nagaraj** Vice-Chancellor, **Dr.V.Vasudevan** Ph.D., Registrar**Dr.P.Deepalakshmi** Ph.D., Dean (School of Computing) . And also a special thanks to **Dr. A. FRANCIS SAVIOUR DEVARAJ.** Head Department of CSE, Kalasalingam Academy of Research and Education forgranting the permission and providing necessary facilities to carry out Project work.

I would like to express my special appreciation and profound thanks to my enthusiastic Project Supervisor **Dr.R.Ramalakshmi** Ph.D, Associate Professor at Kalasalingam Academy of Research and Education [KARE] for her inspiring guidance, constant encouragement with my work during all stages. I am extremely glad that I had a chance to do my Project under my Guide, who truly practices and appreciates deep thinking. I will be forever indebted to my Guide for all the time he has spent with me in discussions. And during the most difficult times when writing this report, he gave me the moral support and the freedom I needed to move on

T V S K LIKHIT

9918004116

M SAI DHANUNJAI

9918004070

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Our project is about Scratch Pad.It is a simple text editor for Microsoft Windows and a basic text editing program which enables computer users to create documents.

In this project we can change set pad color and font color in the Scratch Pad.We can save Scratch Pad in the PC and easily find contains in the Scratch Pad and we replace contains in the Scratch Pad.It will copy and paste in the Scratch Pad

1.0.1 Objectives

List the objectives of the project work...

1. To develop a code on Scratch Pad

Chapter 2

HISTORY

Microsoft introduced Multi-Tool Notepad, a mouse-based text editor written by Richard Brodie, in May 1983 at the Spring COMDEX computer expo in Atlanta. Also introduced at that COMDEX was Multi-Tool Word, designed by Charles Simonyi to work with the mouse. Most watching Simonyi's demonstration had never heard of a mouse. Microsoft released the Microsoft Mouse in June 1983, and the boxed mouse and Multi-Tool Notepad began shipping in July. Initial sales were modest, as there was little one could do with it except run the three demonstration programs included in the box (a tutorial, practice application and Notepad) or program interfaces to it. The Multi-Tool product line began with expert systems for the Multiplan spreadsheet. On the suggestion of Rowland Hanson, who also convinced Bill Gates to change the name "Interface Manager" to "Windows" before the release of Windows 1.0, the Multi-Tool name was killed by the time Word shipped in November 1983. Hanson's rationale was that "the brand is the hero". People didn't associate the stand-alone name Multi-Tool with Microsoft, and Hanson wanted to make Microsoft the hero, so the Microsoft name replaced "Multi-Tool".

It was added to the Microsoft Store in August 2019. Though It will still be included in Windows out of the box, as of Windows 10 version 20H1, It will no longer be a component of the operating system and updated through the bi-yearly Windows 10 version updates, and will instead be a separate application receiving updates through the Microsoft Store. This will allow updates to the app to be delivered more frequently.

Chapter 3

PACKAGES USED

3.0.1 Java AWT

Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java. Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java. Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS. The java.awt package provides classes for AWT api such as Text Field, Label, Text Area, Radio Button, Checkbox, Choice, List etc. The java.awt package provides classes for AWT api such as Text Field, Label, Text Area, Radio Button, Checkbox, Choice, List etc.

3.0.2 Java SWING

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform independent and lightweight components. The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

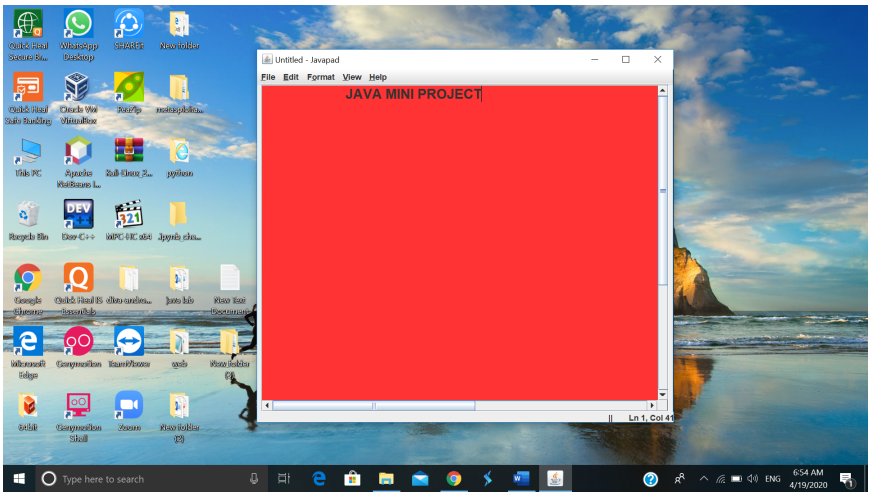

Figure 3.1: Output

Chapter 4

CONCLUSION

In this Scratch Pad, It can easily take note and add to do lists.It shows total Scratch pad. It will allow some types using extensions.Title button are significantly ease and speed up the work.

Appendices

SOURCE CODE

```
//package p1;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
class FindReplaceDemo extends JFrame
{
FindDialog dialog=null;
JTextArea ta;
JButton findButton,replaceButton;
FindReplaceDemo()
{
super("Find Demo");
ta=new JTextArea(7,20);
findButton=new JButton("Find text");
ActionListener ac1=new ActionListener()
{
public void actionPerformed(ActionEvent ev)
{
if(dialog==null)
        dialog=new FindDialog(FindReplaceDemo.this.ta);
dialog.showDialog(FindReplaceDemo.this,true);//find
}
};
findButton.addActionListener(ac1);
replaceButton=new JButton("Replace text");
ActionListener ac2=new ActionListener()
{
public void actionPerformed(ActionEvent ev)
{
if(dialog==null)
        dialog=new FindDialog(FindReplaceDemo.this.ta);
dialog.showDialog(FindReplaceDemo.this,false);//find
}
};
replaceButton.addActionListener(ac2);
```

```
add(ta,BorderLayout.CENTER);
add(replaceButton,BorderLayout.NORTH);
add(findButton,BorderLayout.SOUTH);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(50,50,400,400);
ta.append("Hello dear. how r u?");
ta.append("\nhey i said Hello and not hello or Hel or
    ↪ hello.");
ta.append("\nWell do u know what is the meaning of
    ↪ Hello");
ta.append("\n Hello is no hello but it is Hello");
ta.setCaretPosition(0);
setVisible(true);
}
public static void main(String[] args)
{
new FindReplaceDemo();
}
}
public class FindDialog extends JPanel implements
    ↪ ActionListener
{
JTextArea jta;
public int lastIndex;
JLabel replaceLabel;
private TextField findWhat;
private JTextField replaceWith;
private JCheckBox matchCase;
JRadioButton up, down;
JButton findNextButton, replaceButton, replaceAllButton
    ↪ , cancelButton;
JPanel direction,buttonPanel, findButtonPanel,
    ↪ replaceButtonPanel;
CardLayout card;
private boolean ok;
private JDialog dialog;
public FindDialog(JTextArea jta)
{
this.jta=jta;
findWhat=new TextField(20);
```

```
replaceWith=new JTextField(20);
matchCase=new JCheckBox("Match case");
up=new JRadioButton("Up");
down=new JRadioButton("Down");
down.setSelected(true);
ButtonGroup bg=new ButtonGroup();
bg.add(up);
bg.add(down);
direction=new JPanel();
Border etched=BorderFactory.createEtchedBorder();
Border titled=BorderFactory.createTitledBorder(etched,"
   ↪ Direction");
direction.setBorder(titled);
direction.setLayout(new GridLayout(1,2));
direction.add(up);
direction.add(down);
JPanel southPanel=new JPanel();
southPanel.setLayout(new GridLayout(1,2));
southPanel.add(matchCase);
southPanel.add(direction);
findNextButton=new JButton("Find Next");
replaceButton=new JButton("Replace");
replaceAllButton=new JButton("Replace All");
cancelButton=new JButton("Cancel");
/*
findButtonPanel=new JPanel();
findButtonPanel.setLayout(new GridLayout(2,1));
findButtonPanel.add(findNextButton);
findButtonPanel.add(cancelButton);
*/
replaceButtonPanel=new JPanel();
replaceButtonPanel.setLayout(new GridLayout(4,1));
replaceButtonPanel.add(findNextButton);
replaceButtonPanel.add(replaceButton);
replaceButtonPanel.add(replaceAllButton);
replaceButtonPanel.add(cancelButton);
/*
card=new CardLayout();
buttonPanel=new JPanel();
buttonPanel.setLayout(card);
```

```
buttonPanel.add(replaceButtonPanel,"replace");
buttonPanel.add(findButtonPanel,"find");
card.first(buttonPanel);
*/
JPanel textPanel=new JPanel();
textPanel.setLayout(new GridLayout(3,2));
textPanel.add(new JLabel("Find what "));
textPanel.add(findWhat);
textPanel.add(replaceLabel=new JLabel("Replace With "))
    ↪ ;
textPanel.add(replaceWith);
textPanel.add(new JLabel(" "));//dummy Lable
textPanel.add(new JLabel(" "));//dummy Lable
setLayout(new BorderLayout());
add(new JLabel("        "),BorderLayout.NORTH);
add(textPanel,BorderLayout.CENTER);
add(replaceButtonPanel,BorderLayout.EAST);
add(southPanel,BorderLayout.SOUTH);
setSize(200,200);
findNextButton.addActionListener(this);
replaceButton.addActionListener(this);
replaceAllButton.addActionListener(this);
cancelButton.addActionListener(new ActionListener()
        {public void actionPerformed(ActionEvent ev){
            ↪ dialog.setVisible(false);}});
findWhat.addFocusListener(
        new FocusAdapter(){public void focusLost(
            ↪ FocusEvent te){enableDisableButtons();}})
            ↪ ;
findWhat.addTextListener(
        new TextListener(){public void textValueChanged
            ↪ (TextEvent te){enableDisableButtons();}})
            ↪ ;

}
void enableDisableButtons()
{
if(findWhat.getText().length()==0)
{
findNextButton.setEnabled(false);
```

```
replaceButton.setEnabled(false);
replaceAllButton.setEnabled(false);
}
else
{
findNextButton.setEnabled(true);
replaceButton.setEnabled(true);
replaceAllButton.setEnabled(true);
}
}
public void actionPerformed(ActionEvent ev)
{

if(ev.getSource()==findNextButton)
        findNextWithSelection();
else if(ev.getSource()==replaceButton)
        replaceNext();
else if(ev.getSource()==replaceAllButton)
        JOptionPane.showMessageDialog(null,"Total␣
           ↪ replacements␣made=␣"+replaceAllNext());

}
int findNext()
{

String s1=jta.getText();
String s2=findWhat.getText();
lastIndex=jta.getCaretPosition();
int selStart=jta.getSelectionStart();
int selEnd=jta.getSelectionEnd();
if(up.isSelected())
{
if(selStart!=selEnd)
        lastIndex=selEnd-s2.length()-1;
/*****Notepad doesnt use the else part, but it should
   ↪ be, instead of using caretPosition.***
else
        lastIndex=lastIndex-s2.length();
******/
if(!matchCase.isSelected())
```

```
        lastIndex=s1.toUpperCase().lastIndexOf(s2.
            ↪ toUpperCase(),lastIndex);
else
        lastIndex=s1.lastIndexOf(s2,lastIndex);
}
else
{
if(selStart!=selEnd)
        lastIndex=selStart+1;
if(!matchCase.isSelected())
        lastIndex=s1.toUpperCase().indexOf(s2.
            ↪ toUpperCase(),lastIndex);
else
        lastIndex=s1.indexOf(s2,lastIndex);
}
return lastIndex;
}
public void findNextWithSelection()
{
int idx=findNext();
if(idx!=-1)
{
jta.setSelectionStart(idx);
jta.setSelectionEnd(idx+findWhat.getText().length());
}
else
        JOptionPane.showMessageDialog(this,
        "Cannot␣find"+"␣\""+findWhat.getText()+"\"",
        "Find",JOptionPane.INFORMATION_MESSAGE);
}
void replaceNext()
{
if(jta.getSelectionStart()==jta.getSelectionEnd())
        {findNextWithSelection();return;}

String searchText=findWhat.getText();
String temp=jta.getSelectedText();
if(
        (matchCase.isSelected() && temp.equals(
            ↪ searchText))
```

```
                                        ||
            (!matchCase.isSelected() && temp.
                ↪ equalsIgnoreCase(searchText))
    )
            jta.replaceSelection(replaceWith.getText());
findNextWithSelection();
}
int replaceAllNext()
{
if(up.isSelected())
            jta.setCaretPosition(jta.getText().length()-1);
else
            jta.setCaretPosition(0);
int idx=0;
int counter=0;
do
{
idx=findNext();
if(idx==-1) break;
counter++;
jta.replaceRange(replaceWith.getText(),idx,idx+findWhat
    ↪ .getText().length());
}while(idx!=-1);
return counter;
}
public boolean showDialog(Component parent, boolean
    ↪ isFind )
{
Frame owner=null;
if(parent instanceof Frame)
            owner=(Frame)parent;
else
            owner=(Frame)SwingUtilities.getAncestorOfClass(
                ↪ Frame.class,parent);
if(dialog==null || dialog.getOwner()!=owner)
{
dialog=new JDialog(owner,false);
dialog.add(this);
dialog.getRootPane().setDefaultButton(findNextButton);
}
```

```
if(findWhat.getText().length()==0)
        findNextButton.setEnabled(false);
else
        findNextButton.setEnabled(true);
replaceButton.setVisible(false);
replaceAllButton.setVisible(false);
replaceWith.setVisible(false);
replaceLabel.setVisible(false);
if(isFind)
{
//card.show(buttonPanel,"find");
dialog.setSize(460,180);
dialog.setTitle("Find");
}
else
{
replaceButton.setVisible(true);
replaceAllButton.setVisible(true);
replaceWith.setVisible(true);
replaceLabel.setVisible(true);
//card.show(buttonPanel,"replace");
dialog.setSize(450,200);
dialog.setTitle("Replace");
}
dialog.setVisible(true);
//System.out.println(dialog.getWidth()+" "+dialog.
   ↪ getHeight());
return ok;
}
}
```

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
class FontDemo extends JFrame
{
FontChooser dialog=null;
JTextArea ta;
```

```
JButton fontButton;
FontDemo()
{
super("Font");
ta=new JTextArea(7,20);
fontButton=new JButton("Set Font");
ActionListener ac=new ActionListener()
{
public void actionPerformed(ActionEvent ev)
{
if(dialog==null)
        dialog=new FontChooser(ta.getFont());
if(dialog.showDialog(FontDemo.this,"Choose a font"))
        {
        FontDemo.this.ta.setFont(dialog.createFont());
        }
}
};
fontButton.addActionListener(ac);
add(ta,BorderLayout.CENTER);
add(fontButton,BorderLayout.SOUTH);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(50,50,400,400);
ta.append("Hello dear. how r u?");
ta.append("\n\n A quick brown fox jumps over the lazy
   ↪ dog.");
ta.append("\n\n0123456789");
ta.append("\n~!@#$%^&*()_+|?><");
setVisible(true);
}
public static void main(String[] args)
{
new FontDemo();
}
}
public class FontChooser extends JPanel
{
private Font thisFont;
private JList jFace, jStyle, jSize;
private JDialog dialog;
```

```
private JButton okButton;
JTextArea tf;
private boolean ok;
public FontChooser(Font withFont)
{
thisFont=withFont;
String[] fontNames=GraphicsEnvironment.
   ↪ getLocalGraphicsEnvironment().
   ↪ getAvailableFontFamilyNames();
jFace=new JList(fontNames); jFace.setSelectedIndex(0);
jFace.addListSelectionListener(new
   ↪ ListSelectionListener()
{
public void valueChanged(ListSelectionEvent ev)
{
tf.setFont(createFont());
}
}
);
String[] fontStyles={"Regular","Italic","Bold","Bold␣
   ↪ Italic"};
jStyle=new JList(fontStyles);jStyle.setSelectedIndex(0)
   ↪ ;
jStyle.addListSelectionListener(new
   ↪ ListSelectionListener()
{
public void valueChanged(ListSelectionEvent ev){tf.
   ↪ setFont(createFont());}});
String[] fontSizes=new String[30];
for(int j=0; j<30; j++)
        fontSizes[j]=new String(10+j*2+"");
jSize=new JList(fontSizes); jSize.setSelectedIndex(0);
jSize.addListSelectionListener(new
   ↪ ListSelectionListener()
{public void valueChanged(ListSelectionEvent ev){tf.
   ↪ setFont(createFont());}});
JPanel jpLabel=new JPanel();
jpLabel.setLayout(new GridLayout(1,3));
jpLabel.add(new JLabel("Font",JLabel.CENTER));
jpLabel.add(new JLabel("Font␣Style",JLabel.CENTER));
```

```
jpLabel.add(new JLabel("Size",JLabel.CENTER));
JPanel jpList=new JPanel();
jpList.setLayout(new GridLayout(1,3));
jpList.add(new JScrollPane(jFace));
jpList.add(new JScrollPane(jStyle));
jpList.add(new JScrollPane(jSize));
okButton=new JButton("OK");
JButton cancelButton=new JButton("Cancel");
okButton.addActionListener(
new ActionListener()
{
public void actionPerformed(ActionEvent ev)
{
ok=true;
FontChooser.this.thisFont=FontChooser.this.createFont()
    ↪ ;
dialog.setVisible(false);
}
});
cancelButton.addActionListener(
new ActionListener()
{
public void actionPerformed(ActionEvent ev)
{
dialog.setVisible(false);
}
});
JPanel jpButton=new JPanel();
jpButton.setLayout(new FlowLayout());
jpButton.add(okButton);
jpButton.add(new JLabel("␣␣␣␣␣␣␣␣␣␣"));
jpButton.add(cancelButton);
tf=new JTextArea(5,30);
JPanel jpTextField=new JPanel();
jpTextField.add(new JScrollPane(tf));
JPanel centerPanel=new JPanel();
centerPanel.setLayout(new GridLayout(2,1));
centerPanel.add(jpList);
centerPanel.add(jpTextField);
setLayout(new BorderLayout());
```

```
add(jpLabel,BorderLayout.NORTH);
add(centerPanel,BorderLayout.CENTER);
add(jpButton,BorderLayout.SOUTH);
add(new JLabel("  "),BorderLayout.EAST);
add(new JLabel("  "),BorderLayout.WEST);
tf.setFont(thisFont);
tf.append("\nA quick brown fox jumps over the lazy dog.
   ↪ ");
tf.append("\n0123456789");
tf.append("\n~!@#$%^&*()_+|?><\n");
}
public Font createFont()
{
Font fnt=thisFont;
int fontstyle=Font.PLAIN;
int x=jStyle.getSelectedIndex();
switch(x)
{
case 0:
        fontstyle=Font.PLAIN;    break;
case 1:
        fontstyle=Font.ITALIC;   break;
case 2:
        fontstyle=Font.BOLD;     break;
case 3:
        fontstyle=Font.BOLD+Font.ITALIC;          break;
}
int fontsize=Integer.parseInt((String)jSize.
   ↪ getSelectedValue());
String fontname=(String)jFace.getSelectedValue();
fnt=new Font(fontname,fontstyle,fontsize);
return fnt;
}
public boolean showDialog(Component parent, String
   ↪ title)
{
ok=false;
Frame owner=null;
if(parent instanceof Frame)
        owner=(Frame)parent;
```

```
else
        owner=(Frame)SwingUtilities.getAncestorOfClass(
            ↪ Frame.class,parent);
if(dialog==null || dialog.getOwner()!=owner)
{
dialog=new JDialog(owner,true);
dialog.add(this);
dialog.getRootPane().setDefaultButton(okButton);
dialog.setSize(400,325);
}
dialog.setTitle(title);
dialog.setVisible(true);
//System.out.println(dialog.getWidth()+" "+dialog.
    ↪ getHeight());
return ok;
}
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class LookAndFeelDemo extends JFrame
{
JLabel myLabel;
JMenuBar jmb;
JMenu fileMenu;
LookAndFeelDemo()
{
super("Look and Feel Demo");
add(myLabel=new JLabel("This is a Label"));
add(new JButton("Button"));
add(new JCheckBox("CheckBox"));
add(new JRadioButton("RadioButton"));
setLayout(new FlowLayout());
setSize(350,350);
setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE)
    ↪ ;
jmb=new JMenuBar();
setJMenuBar(jmb);
fileMenu=new JMenu("Look and Feel");
```

```
jmb.add(fileMenu);
LookAndFeelMenu.createLookAndFeelMenuItem(fileMenu,this
    ↪ );
setVisible(true);
}
public static void main(String[] args)
{
new LookAndFeelDemo();
}
}
public class LookAndFeelMenu
{
public static void createLookAndFeelMenuItem(JMenu
    ↪ jmenu,Component cmp)
{
final UIManager.LookAndFeelInfo[] infos=UIManager.
    ↪ getInstalledLookAndFeels();
JRadioButtonMenuItem rbm[]=new JRadioButtonMenuItem[
    ↪ infos.length];
ButtonGroup bg=new ButtonGroup();
JMenu tmp=new JMenu("Change␣Look␣and␣Feel");
tmp.setMnemonic('C');
for(int i=0; i<infos.length; i++)
{
rbm[i]=new JRadioButtonMenuItem(infos[i].getName());
rbm[i].setMnemonic(infos[i].getName().charAt(0));
tmp.add(rbm[i]);
bg.add(rbm[i]);
rbm[i].addActionListener(new LookAndFeelMenuListener(
    ↪ infos[i].getClassName(),cmp));
}
rbm[0].setSelected(true);
jmenu.add(tmp);
}
}
class LookAndFeelMenuListener implements ActionListener
{
String classname;
Component jf;
LookAndFeelMenuListener(String cln,Component jf)
```

```
{
this.jf=jf;
classname=new String(cln);
}
public void actionPerformed(ActionEvent ev)
{
try
{
UIManager.setLookAndFeel(classname);
SwingUtilities.updateComponentTreeUI(jf);
}
catch(Exception e){System.out.println(e);}
}
}
```

```
import java.io.File;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.filechooser.FileFilter;
class FileFilterDemo extends JFrame
{
JLabel myLabel;
JButton myButton;
JFileChooser chooser;
FileFilterDemo()
{
super("File Filter Demo");
myLabel=new JLabel("No file is choosed yet");
myButton=new JButton("Choose file");
ActionListener listener=new ActionListener()
{
public void actionPerformed(ActionEvent ev)
{
if (FileFilterDemo.this.chooser==null)
        chooser=new JFileChooser();
chooser.addChoosableFileFilter(new MyFileFilter(".java"
   ↪ ,"Java Source Files(*.java)"));
chooser.addChoosableFileFilter(new MyFileFilter(".txt",
   ↪ "Text Files(*.txt)"));
if(chooser.showDialog(FileFilterDemo.this,"Select this"
```

```
    ↪ )==JFileChooser.APPROVE_OPTION)
FileFilterDemo.this.myLabel.setText(chooser.
    ↪ getSelectedFile().getPath());
}
};
myButton.addActionListener(listener);
add(myLabel,"Center");
add(myButton,"South");
setSize(300,300);
setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE)
    ↪ ;
}
public static void main(String[] args)
{
FileFilterDemo ffd=new FileFilterDemo();
ffd.setVisible(true);
}
}
public class MyFileFilter extends FileFilter
{
private String extension;
private String description;
public MyFileFilter()
{
setExtension(null);
setDescription(null);
}
public MyFileFilter(final String ext, final String desc
    ↪ )
{
setExtension(ext);
setDescription(desc);
}

public boolean accept(File f)
{
final String filename=f.getName();

if(      f.isDirectory() ||
         extension==null ||
```

```
        filename.toUpperCase()
        .endsWith(extension.toUpperCase()))
        return true;
return false;

}

public String getDescription()
{
return description;
}

public void setDescription(String desc)
{
if(desc==null)
        description=new String("All Files(*.*)");
else
        description=new String(desc);
}
public void setExtension(String ext)
{
if(ext==null)
{
extension=null;
return;
}
extension=new String(ext).toLowerCase();
if(!ext.startsWith("."))
extension="."+extension;
}
}
```

```
//package p1;
import java.io.*;
import java.util.Date;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
//import p1.FontChooser;
```

```
//import p1.FontDialog;
//import p1.FindDialog;
//import p1.LookAndFeelMenu;
//import p1.MyFileFilter;
class FileOperation
{
Notepad npd;
boolean saved;
boolean newFileFlag;
String fileName;
String applicationTitle="Javapad";
File fileRef;
JFileChooser chooser;
boolean isSave(){return saved;}
void setSave(boolean saved){this.saved=saved;}
String getFileName(){return new String(fileName);}
void setFileName(String fileName){this.fileName=new
    ↪ String(fileName);}
FileOperation(Notepad npd)
{
this.npd=npd;
saved=true;
newFileFlag=true;
fileName=new String("Untitled");
fileRef=new File(fileName);
this.npd.f.setTitle(fileName+" - "+applicationTitle);
chooser=new JFileChooser();
chooser.addChoosableFileFilter(new MyFileFilter(".java"
    ↪ ,"Java Source Files(*.java)"));
chooser.addChoosableFileFilter(new MyFileFilter(".txt",
    ↪ "Text Files(*.txt)"));
chooser.setCurrentDirectory(new File("."));
}
boolean saveFile(File temp)
{
FileWriter fout=null;
try
{
fout=new FileWriter(temp);
fout.write(npd.ta.getText());
```

```
}
catch(IOException ioe){updateStatus(temp,false);return
    ↪ false;}
finally
{try{fout.close();}catch(IOException excp){}}
updateStatus(temp,true);
return true;
}
boolean saveThisFile()
{
if(!newFileFlag)
        {return saveFile(fileRef);}
return saveAsFile();
}
boolean saveAsFile()
{
File temp=null;
chooser.setDialogTitle("Save␣As...");
chooser.setApproveButtonText("Save␣Now");
chooser.setApproveButtonMnemonic(KeyEvent.VK_S);
chooser.setApproveButtonToolTipText("Click␣me␣to␣save!"
    ↪ );
do
{
if(chooser.showSaveDialog(this.npd.f)!=JFileChooser.
    ↪ APPROVE_OPTION)
        return false;
temp=chooser.getSelectedFile();
if(!temp.exists()) break;
if(    JOptionPane.showConfirmDialog(
        this.npd.f,"<html>"+temp.getPath()+"␣already␣
            ↪ exists.<br>Do␣you␣want␣to␣replace␣it?<
            ↪ html>",
        "Save␣As",JOptionPane.YES_NO_OPTION
                                    )==JOptionPane.
                                        ↪ YES_OPTION)
        break;
}while(true);
return saveFile(temp);
}
```

```
boolean openFile(File temp)
{
FileInputStream fin=null;
BufferedReader din=null;
try
{
fin=new FileInputStream(temp);
din=new BufferedReader(new InputStreamReader(fin));
String str="␣";
while(str!=null)
{
str=din.readLine();
if(str==null)
break;
this.npd.ta.append(str+"\n");
}
}
catch(IOException ioe){updateStatus(temp,false);return
    ↪ false;}
finally
{try{din.close();fin.close();}catch(IOException excp)
    ↪ {}}
updateStatus(temp,true);
this.npd.ta.setCaretPosition(0);
return true;
}
void openFile()
{
if(!confirmSave()) return;
chooser.setDialogTitle("Open␣File...");
chooser.setApproveButtonText("Open␣this");
chooser.setApproveButtonMnemonic(KeyEvent.VK_O);
chooser.setApproveButtonToolTipText("Click␣me␣to␣open␣
    ↪ the␣selected␣file.!");
File temp=null;
do
{
if(chooser.showOpenDialog(this.npd.f)!=JFileChooser.
    ↪ APPROVE_OPTION)
        return;
```

```
temp=chooser.getSelectedFile();
if(temp.exists())          break;
JOptionPane.showMessageDialog(this.npd.f,
        "<html>"+temp.getName()+"<br>file not found.<br
            ↪ >"+
        "Please verify the correct file name was given
            ↪ .<html>",
        "Open", JOptionPane.INFORMATION_MESSAGE);
}
while(true);
this.npd.ta.setText("");
if(!openFile(temp))
        {
        fileName="Untitled"; saved=true;
        this.npd.f.setTitle(fileName+" - "+
            ↪ applicationTitle);
        }
if(!temp.canWrite())
        newFileFlag=true;
}
void updateStatus(File temp,boolean saved)
{
if(saved)
{
this.saved=true;
fileName=new String(temp.getName());
if(!temp.canWrite())
        {fileName+="(Read only)"; newFileFlag=true;}
fileRef=temp;
npd.f.setTitle(fileName + " - "+applicationTitle);
npd.statusBar.setText("File : "+temp.getPath()+" saved/
    ↪ opened successfully.");
newFileFlag=false;
}
else
{
npd.statusBar.setText("Failed to save/open : "+temp.
    ↪ getPath());
}
}
```

```
boolean confirmSave()
{
String strMsg="<html>The text in the "+fileName+" file
    ↪ has been changed.<br>"+
          "Do you want to save the changes?<html>";
if(!saved)
{
int x=JOptionPane.showConfirmDialog(this.npd.f,strMsg,
    ↪ applicationTitle,JOptionPane.YES_NO_CANCEL_OPTION
    ↪ );

if(x==JOptionPane.CANCEL_OPTION) return false;
if(x==JOptionPane.YES_OPTION && !saveAsFile()) return
    ↪ false;
}
return true;
}
void newFile()
{
if(!confirmSave()) return;
this.npd.ta.setText("");
fileName=new String("Untitled");
fileRef=new File(fileName);
saved=true;
newFileFlag=true;
this.npd.f.setTitle(fileName+" - "+applicationTitle);
}
}
public class Notepad  implements ActionListener,
    ↪ MenuConstants
{
JFrame f;
JTextArea ta;
JLabel statusBar;
private String fileName="Untitled";
private boolean saved=true;
String applicationName="Javapad";
String searchString, replaceString;
int lastSearchIndex;
FileOperation fileHandler;
```

```
FontChooser fontDialog=null;
FindDialog findReplaceDialog=null;
JColorChooser bcolorChooser=null;
JColorChooser fcolorChooser=null;
JDialog backgroundDialog=null;
JDialog foregroundDialog=null;
JMenuItem cutItem,copyItem, deleteItem, findItem,
    ↪ findNextItem, replaceItem, gotoItem,
    ↪ selectAllItem;
Notepad()
{
f=new JFrame(fileName+" - "+applicationName);
ta=new JTextArea(30,60);
statusBar=new JLabel("||        Ln 1, Col 1  ",JLabel.
    ↪ RIGHT);
f.add(new JScrollPane(ta),BorderLayout.CENTER);
f.add(statusBar,BorderLayout.SOUTH);
f.add(new JLabel("  "),BorderLayout.EAST);
f.add(new JLabel("  "),BorderLayout.WEST);
createMenuBar(f);
//f.setSize(350,350);
f.pack();
f.setLocation(100,50);
f.setVisible(true);
f.setLocation(150,50);
f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
fileHandler=new FileOperation(this);
ta.addCaretListener(
new CaretListener()
{
public void caretUpdate(CaretEvent e)
{
int lineNumber=0, column=0, pos=0;
try
{
pos=ta.getCaretPosition();
lineNumber=ta.getLineOfOffset(pos);
column=pos-ta.getLineStartOffset(lineNumber);
}catch(Exception excp){}
if(ta.getText().length()==0){lineNumber=0; column=0;}
```

```
statusBar.setText("||␣␣␣␣␣␣␣Ln␣"+(lineNumber+1)+",␣Col␣
    ↪ "+(column+1));
}
});
DocumentListener myListener = new DocumentListener()
{
public void changedUpdate(DocumentEvent e){fileHandler.
    ↪ saved=false;}
public void removeUpdate(DocumentEvent e){fileHandler.
    ↪ saved=false;}
public void insertUpdate(DocumentEvent e){fileHandler.
    ↪ saved=false;}
};
ta.getDocument().addDocumentListener(myListener);
WindowListener frameClose=new WindowAdapter()
{
public void windowClosing(WindowEvent we)
{
if(fileHandler.confirmSave())System.exit(0);
}
};
f.addWindowListener(frameClose);
/*
ta.append("Hello dear hello hi");
ta.append("\nwho are u dear mister hello");
ta.append("\nhello bye hel");
ta.append("\nHello");
ta.append("\nMiss u mister hello hell");
fileHandler.saved=true;
*/
}
void goTo()
{
int lineNumber=0;
try
{
lineNumber=ta.getLineOfOffset(ta.getCaretPosition())+1;
String tempStr=JOptionPane.showInputDialog(f,"Enter␣
    ↪ Line␣Number:","",""+lineNumber);
if(tempStr==null)
```

```
        {return;}
lineNumber=Integer.parseInt(tempStr);
ta.setCaretPosition(ta.getLineStartOffset(lineNumber-1)
    ↪ );
}catch(Exception e){}
}
public void actionPerformed(ActionEvent ev)
{
String cmdText=ev.getActionCommand();
if(cmdText.equals(fileNew))
        fileHandler.newFile();
else if(cmdText.equals(fileOpen))
        fileHandler.openFile();
else if(cmdText.equals(fileSave))
        fileHandler.saveThisFile();
else if(cmdText.equals(fileSaveAs))
        fileHandler.saveAsFile();
else if(cmdText.equals(fileExit))
        {if(fileHandler.confirmSave())System.exit(0);}
else if(cmdText.equals(filePrint))
JOptionPane.showMessageDialog(
        Notepad.this.f,
        "Get␣ur␣printer␣repaired␣first!␣It␣seems␣u␣dont
            ↪ ␣have␣one!",
        "Bad␣Printer",
        JOptionPane.INFORMATION_MESSAGE
        );
else if(cmdText.equals(editCut))
        ta.cut();
else if(cmdText.equals(editCopy))
        ta.copy();
else if(cmdText.equals(editPaste))
        ta.paste();
else if(cmdText.equals(editDelete))
        ta.replaceSelection("");
else if(cmdText.equals(editFind))
{
if(Notepad.this.ta.getText().length()==0)
        return; // text box have no text
if(findReplaceDialog==null)
```

```
        findReplaceDialog=new FindDialog(Notepad.this.
            ↪ ta);
findReplaceDialog.showDialog(Notepad.this.f,true);
}
else if(cmdText.equals(editFindNext))
{
if(Notepad.this.ta.getText().length()==0)
        return;
if(findReplaceDialog==null)
        statusBar.setText("Nothing␣to␣search␣for,␣use␣
            ↪ Find␣option␣of␣Edit␣Menu␣first␣!!!!");
else
        findReplaceDialog.findNextWithSelection();
}
else if(cmdText.equals(editReplace))
{
if(Notepad.this.ta.getText().length()==0)
        return;
if(findReplaceDialog==null)
        findReplaceDialog=new FindDialog(Notepad.this.
            ↪ ta);
findReplaceDialog.showDialog(Notepad.this.f,false);
}
else if(cmdText.equals(editGoTo))
{
if(Notepad.this.ta.getText().length()==0)
        return;
goTo();
}
else if(cmdText.equals(editSelectAll))
        ta.selectAll();
else if(cmdText.equals(editTimeDate))
        ta.insert(new Date().toString(),ta.
            ↪ getSelectionStart());
else if(cmdText.equals(formatWordWrap))
{
JCheckBoxMenuItem temp=(JCheckBoxMenuItem)ev.getSource
    ↪ ();
ta.setLineWrap(temp.isSelected());
}
```

```
else if(cmdText.equals(formatFont))
{
if(fontDialog==null)
        fontDialog=new FontChooser(ta.getFont());

if(fontDialog.showDialog(Notepad.this.f,"Choose a font"
    ↪ ))
        Notepad.this.ta.setFont(fontDialog.createFont()
            ↪ );
}
else if(cmdText.equals(formatForeground))
        showForegroundColorDialog();
else if(cmdText.equals(formatBackground))
        showBackgroundColorDialog();
else if(cmdText.equals(viewStatusBar))
{
JCheckBoxMenuItem temp=(JCheckBoxMenuItem)ev.getSource
    ↪ ();
statusBar.setVisible(temp.isSelected());
}
else if(cmdText.equals(helpAboutNotepad))
{
JOptionPane.showMessageDialog(Notepad.this.f,aboutText,
    ↪ "Dedicated 2 u!",JOptionPane.INFORMATION_MESSAGE)
    ↪ ;
}
else
        statusBar.setText("This "+cmdText+" command is
            ↪ yet to be implemented");
}
void showBackgroundColorDialog()
{
if(bcolorChooser==null)
        bcolorChooser=new JColorChooser();
if(backgroundDialog==null)
        backgroundDialog=JColorChooser.createDialog
                (Notepad.this.f,
                formatBackground,
                false,
                bcolorChooser,
```

```
                new ActionListener()
                {public void actionPerformed(
                    ↪ ActionEvent evvv){
                        Notepad.this.ta.setBackground(
                            ↪ bcolorChooser.getColor())
                            ↪ ;}},
                null);

backgroundDialog.setVisible(true);
}
void showForegroundColorDialog()
{
if(fcolorChooser==null)
        fcolorChooser=new JColorChooser();
if(foregroundDialog==null)
        foregroundDialog=JColorChooser.createDialog
                (Notepad.this.f,
                formatForeground,
                false,
                fcolorChooser,
                new ActionListener()
                {public void actionPerformed(
                    ↪ ActionEvent evvv){
                        Notepad.this.ta.setForeground(
                            ↪ fcolorChooser.getColor())
                            ↪ ;}},
                null);
foregroundDialog.setVisible(true);
}
JMenuItem createMenuItem(String s, int key,JMenu toMenu
    ↪ ,ActionListener al)
{
JMenuItem temp=new JMenuItem(s,key);
temp.addActionListener(al);
toMenu.add(temp);
return temp;
}
JMenuItem createMenuItem(String s, int key,JMenu toMenu
    ↪ ,int aclKey,ActionListener al)
{
```

```
JMenuItem temp=new JMenuItem(s,key);
temp.addActionListener(al);
temp.setAccelerator(KeyStroke.getKeyStroke(aclKey,
    ↪ ActionEvent.CTRL_MASK));
toMenu.add(temp);
return temp;
}
JCheckBoxMenuItem createCheckBoxMenuItem(String s, int
    ↪ key,JMenu toMenu,ActionListener al)
{
JCheckBoxMenuItem temp=new JCheckBoxMenuItem(s);
temp.setMnemonic(key);
temp.addActionListener(al);
temp.setSelected(false);
toMenu.add(temp);
return temp;
}
JMenu createMenu(String s,int key,JMenuBar toMenuBar)
{
JMenu temp=new JMenu(s);
temp.setMnemonic(key);
toMenuBar.add(temp);
return temp;
}
void createMenuBar(JFrame f)
{
JMenuBar mb=new JMenuBar();
JMenuItem temp;
JMenu fileMenu=createMenu(fileText,KeyEvent.VK_F,mb);
JMenu editMenu=createMenu(editText,KeyEvent.VK_E,mb);
JMenu formatMenu=createMenu(formatText,KeyEvent.VK_O,mb
    ↪ );
JMenu viewMenu=createMenu(viewText,KeyEvent.VK_V,mb);
JMenu helpMenu=createMenu(helpText,KeyEvent.VK_H,mb);
createMenuItem(fileNew,KeyEvent.VK_N,fileMenu,KeyEvent.
    ↪ VK_N,this);
createMenuItem(fileOpen,KeyEvent.VK_O,fileMenu,KeyEvent
    ↪ .VK_O,this);
createMenuItem(fileSave,KeyEvent.VK_S,fileMenu,KeyEvent
    ↪ .VK_S,this);
```

```
createMenuItem(fileSaveAs ,KeyEvent.VK_A,fileMenu ,this);
fileMenu.addSeparator();
temp=createMenuItem(filePageSetup ,KeyEvent.VK_U,
    ↪ fileMenu ,this);
temp.setEnabled(false);
createMenuItem(filePrint ,KeyEvent.VK_P,fileMenu ,
    ↪ KeyEvent.VK_P,this);
fileMenu.addSeparator();
createMenuItem(fileExit ,KeyEvent.VK_X,fileMenu ,this);
temp=createMenuItem(editUndo ,KeyEvent.VK_U,editMenu ,
    ↪ KeyEvent.VK_Z,this);
temp.setEnabled(false);
editMenu.addSeparator();
cutItem=createMenuItem(editCut ,KeyEvent.VK_T,editMenu ,
    ↪ KeyEvent.VK_X,this);
copyItem=createMenuItem(editCopy ,KeyEvent.VK_C,editMenu
    ↪ ,KeyEvent.VK_C,this);
createMenuItem(editPaste ,KeyEvent.VK_P,editMenu ,
    ↪ KeyEvent.VK_V,this);
deleteItem=createMenuItem(editDelete ,KeyEvent.VK_L,
    ↪ editMenu ,this);
deleteItem.setAccelerator(KeyStroke.getKeyStroke(
    ↪ KeyEvent.VK_DELETE,0));
editMenu.addSeparator();
findItem=createMenuItem(editFind ,KeyEvent.VK_F,editMenu
    ↪ ,KeyEvent.VK_F,this);
findNextItem=createMenuItem(editFindNext ,KeyEvent.VK_N,
    ↪ editMenu ,this);
findNextItem.setAccelerator(KeyStroke.getKeyStroke(
    ↪ KeyEvent.VK_F3,0));
replaceItem=createMenuItem(editReplace ,KeyEvent.VK_R,
    ↪ editMenu ,KeyEvent.VK_H,this);
gotoItem=createMenuItem(editGoTo ,KeyEvent.VK_G,editMenu
    ↪ ,KeyEvent.VK_G,this);
editMenu.addSeparator();
selectAllItem=createMenuItem(editSelectAll ,KeyEvent.
    ↪ VK_A,editMenu ,KeyEvent.VK_A,this);
createMenuItem(editTimeDate ,KeyEvent.VK_D,editMenu ,this
    ↪ ).setAccelerator(KeyStroke.getKeyStroke(KeyEvent.
    ↪ VK_F5,0));
```

```
createCheckBoxMenuItem(formatWordWrap,KeyEvent.VK_W,
    ↪ formatMenu,this);
createMenuItem(formatFont,KeyEvent.VK_F,formatMenu,this
    ↪ );
formatMenu.addSeparator();
createMenuItem(formatForeground,KeyEvent.VK_T,
    ↪ formatMenu,this);
createMenuItem(formatBackground,KeyEvent.VK_P,
    ↪ formatMenu,this);
createCheckBoxMenuItem(viewStatusBar,KeyEvent.VK_S,
    ↪ viewMenu,this).setSelected(true);
LookAndFeelMenu.createLookAndFeelMenuItem(viewMenu,this
    ↪ .f);
temp=createMenuItem(helpHelpTopic,KeyEvent.VK_H,
    ↪ helpMenu,this);
temp.setEnabled(false);
helpMenu.addSeparator();
createMenuItem(helpAboutNotepad,KeyEvent.VK_A,helpMenu,
    ↪ this);
MenuListener editMenuListener=new MenuListener()
{
    public void menuSelected(MenuEvent evvvv)
        {
        if(Notepad.this.ta.getText().length()==0)
        {
        findItem.setEnabled(false);
        findNextItem.setEnabled(false);
        replaceItem.setEnabled(false);
        selectAllItem.setEnabled(false);
        gotoItem.setEnabled(false);
        }
        else
        {
        findItem.setEnabled(true);
        findNextItem.setEnabled(true);
        replaceItem.setEnabled(true);
        selectAllItem.setEnabled(true);
        gotoItem.setEnabled(true);
        }
        if(Notepad.this.ta.getSelectionStart()==ta.
```

```
            ↪ getSelectionEnd())
        {
        cutItem.setEnabled(false);
        copyItem.setEnabled(false);
        deleteItem.setEnabled(false);
        }
        else
        {
        cutItem.setEnabled(true);
        copyItem.setEnabled(true);
        deleteItem.setEnabled(true);
        }
        }
    public void menuDeselected(MenuEvent evvvv){}
    public void menuCanceled(MenuEvent evvvv){}
};
editMenu.addMenuListener(editMenuListener);
f.setJMenuBar(mb);
}
public static void main(String[] s)
{
new Notepad();
}
}
interface MenuConstants
{
final String fileText="File";
final String editText="Edit";
final String formatText="Format";
final String viewText="View";
final String helpText="Help";
final String fileNew="New";
final String fileOpen="Open...";
final String fileSave="Save";
final String fileSaveAs="Save As...";
final String filePageSetup="Page Setup...";
final String filePrint="Print";
final String fileExit="Exit";
final String editUndo="Undo";
final String editCut="Cut";
```

```
final String editCopy="Copy";
final String editPaste="Paste";
final String editDelete="Delete";
final String editFind="Find...";
final String editFindNext="Find Next";
final String editReplace="Replace";
final String editGoTo="Go To...";
final String editSelectAll="Select All";
final String editTimeDate="Time/Date";
final String formatWordWrap="Word Wrap";
final String formatFont="Font...";
final String formatForeground="Set Text color...";
final String formatBackground="Set Pad color...";
final String viewStatusBar="Status Bar";
final String helpHelpTopic="Help Topic";
final String helpAboutNotepad="About Javapad";
final String aboutText=
        "<html><big>Your Javapad</big><hr><hr>"
        +"<p align=right>Prepared by a Ducatian!"
        +"<hr><p align=left>I Used jdk1.5 to compile
            ↪ the source code.<br><br>"
        +"<strong>Thanx 4 using Javapad</strong><br>"
        +"Ur Comments as well as bug reports r very
            ↪ welcome at<p align=center>"
        +"<hr><em><big>radialgoal@gmail.com</big></em>
            ↪ hr><html>";
}
```