S. Santhosh kumar.....
7305118197.....

//Question 1:.....

```
import java.awt.*;
public class AwtApp extends Frame {

AwtApp(){
Label firstName = new Label("First Name");
firstName.setBounds(20, 50, 80, 20);

Label lastName = new Label("Last Name");
lastName.setBounds(20, 80, 80, 20);

Label dob = new Label("Date of Birth");
dob.setBounds(20, 110, 80, 20);

TextField firstNameTF = new TextField();
firstNameTF.setBounds(120, 50, 100, 20);

lblGender=new Label("Gender");
lblGender.setBounds(250,250,150,30);
lblGender.setFont(labelFont);
lblGender.setForeground(Color.WHITE);
□

cbg = new CheckboxGroup();

checkMale = new Checkbox("Male",cbg,true);
checkMale.setBounds(400,250, 100, 30);
checkMale.setFont(labelFont);
checkMale.setForeground(Color.WHITE);
□□

checkFemale = new Checkbox("Female",cbg,false);
checkFemale.setBounds(500,250, 100, 30);
checkFemale.setFont(labelFont);
checkFemale.setForeground(Color.WHITE);
□□

TextField lastNameTF = new TextField();
lastNameTF.setBounds(120, 80, 100, 20);

TextField dobTF = new TextField();
dobTF.setBounds(120, 110, 100, 20);

Button sbmt = new Button("Submit");
sbmt.setBounds(20, 160, 100, 30);

Button reset = new Button("Reset");
reset.setBounds(120,160,100,30);

add(firstName);
add(lastName);
add(firstNameTF);
add(lastNameTF);
add(dob);
add(dobTF);
```

```
add(lblGender);
add(checkMale);
add(checkFemale);

add(sbmt);


setSize(300,300);
setLayout(null);
setVisible(true);
}
public static void main(String[] args) {
// TODO Auto-generated method stub
AwtApp awt = new AwtApp();
}
}


//Question 2:.....

import java.awt.BorderLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class Calculator implements ActionListener {
  private static JTextField inputBox;

  Calculator(){}
  public static void main(String[] args) {
    createWindow();
  }

  private static void createWindow() {
    JFrame frame = new JFrame("Calculator");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    createUI(frame);
    frame.setSize(200, 200);
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);
  }

  private static void createUI(JFrame frame) {
    JPanel panel = new JPanel();
    Calculator calculator = new Calculator();
    GridBagLayout layout = new GridBagLayout();
    GridBagConstraints gbc = new GridBagConstraints();
    panel.setLayout(layout);

    inputBox = new JTextField(10);
    inputBox.setEditable(false);

    JButton button0 = new JButton("0");JButton button1 = new JButton("1");
```

```
    JButton button2 = new JButton("2");JButton button3 = new JButton("3");
    JButton button4 = new JButton("4");JButton button5 = new JButton("5");
    JButton button6 = new JButton("6");JButton button7 = new JButton("7");
    JButton button8 = new JButton("8");JButton button9 = new JButton("9");

    JButton buttonPlus = new JButton("+");JButton buttonMinus = new JButton("-");
    JButton buttonDivide = new JButton("/");JButton buttonMultiply = new JButton("x");
    JButton buttonClear = new JButton("C");JButton buttonDot = new JButton(".");
    JButton buttonEquals = new JButton("=");

    button1.addActionListener(calculator);button2.addActionListener(calculator);
    button3.addActionListener(calculator);button4.addActionListener(calculator);
    button5.addActionListener(calculator);button6.addActionListener(calculator);
    button7.addActionListener(calculator);button8.addActionListener(calculator);
    button9.addActionListener(calculator);button0.addActionListener(calculator);

    buttonPlus.addActionListener(calculator);buttonMinus.addActionListener(calculator);
    buttonDivide.addActionListener(calculator);buttonMultiply.addActionListener(calculator);
    buttonClear.addActionListener(calculator);buttonDot.addActionListener(calculator);
    buttonEquals.addActionListener(calculator);

    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.gridx = 0; gbc.gridy = 0; panel.add(button1, gbc);
    gbc.gridx = 1; gbc.gridy = 0; panel.add(button2, gbc);
    gbc.gridx = 2; gbc.gridy = 0; panel.add(button3, gbc);
    gbc.gridx = 3; gbc.gridy = 0; panel.add(buttonPlus, gbc);
    gbc.gridx = 0; gbc.gridy = 1; panel.add(button4, gbc);
    gbc.gridx = 1; gbc.gridy = 1; panel.add(button5, gbc);
    gbc.gridx = 2; gbc.gridy = 1; panel.add(button6, gbc);
    gbc.gridx = 3; gbc.gridy = 1; panel.add(buttonMinus, gbc);
    gbc.gridx = 0; gbc.gridy = 2; panel.add(button7, gbc);
    gbc.gridx = 1; gbc.gridy = 2; panel.add(button8, gbc);
    gbc.gridx = 2; gbc.gridy = 2; panel.add(button9, gbc);
    gbc.gridx = 3; gbc.gridy = 2; panel.add(buttonDivide, gbc);
    gbc.gridx = 0; gbc.gridy = 3; panel.add(buttonDot, gbc);
    gbc.gridx = 1; gbc.gridy = 3; panel.add(button0, gbc);
    gbc.gridx = 2; gbc.gridy = 3; panel.add(buttonClear, gbc);
    gbc.gridx = 3; gbc.gridy = 3; panel.add(buttonMultiply, gbc);
    gbc.gridwidth = 3;

    gbc.gridx = 0; gbc.gridy = 4; panel.add(inputBox, gbc);
    gbc.gridx = 3; gbc.gridy = 4; panel.add(buttonEquals, gbc);
    frame.getContentPane().add(panel, BorderLayout.CENTER);
  }

  public void actionPerformed(ActionEvent e) {
    String command = e.getActionCommand();
    if (command.charAt(0) == 'C') {
      inputBox.setText("");
    }else if (command.charAt(0) == '=') {
      inputBox.setText(evaluate(inputBox.getText()));
    }else {
      inputBox.setText(inputBox.getText() + command);
    }
  }

  public static String evaluate(String expression) {
    char[] arr = expression.toCharArray();
    String operand1 = "";String operand2 = "";String operator = "";
    double result = 0;
```

```
    for (int i = 0; i < arr.length; i++) {
      if (arr[i] >= '0' && arr[i] <= '9' || arr[i] == '.') {
        if(operator.isEmpty()){
          operand1 += arr[i];
        }else{
          operand2 += arr[i];
        }
      }

      if(arr[i] == '+' || arr[i] == '-' || arr[i] == '/' || arr[i] == '*') {
        operator += arr[i];
      }
    }

    if (operator.equals("+"))
      result = (Double.parseDouble(operand1) + Double.parseDouble(operand2));
    else if (operator.equals("-"))
      result = (Double.parseDouble(operand1) - Double.parseDouble(operand2));
    else if (operator.equals("/"))
      result = (Double.parseDouble(operand1) / Double.parseDouble(operand2));
    else
      result = (Double.parseDouble(operand1) * Double.parseDouble(operand2));
    return operand1 + operator + operand2 + "=" +result;
  }
}

//Question 3:.....


import java.io.*;
import java.util.Date;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;


class FileOperation
{
Notepad npd;

boolean saved;
boolean newFileFlag;
String fileName;
String applicationTitle="Notepad - JavaTpoint";

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.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);


}

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;
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("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);//replace
}

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="Your Javapad";
}

//Question 4:.....

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

public class Banking implements ActionListener {
  private JFrame mainFrame;
  private JTextField inputField;
  private JLabel balanceLabel;
  private int balance = 0;

  public Banking() {

    mainFrame = new JFrame("Simple Bank Application");
    mainFrame.setVisible(true);
    mainFrame.setSize(400, 400);
    mainFrame.setFont(new Font("Arial",Font.BOLD,18));
    mainFrame.setLayout(new FlowLayout());

    inputField = new JTextField(10);
    balanceLabel = new JLabel("Current balance: " + balance);
    JButton depositButton = new JButton("Deposit");
    JButton withdrawButton = new JButton("Withdraw");

    mainFrame.add(inputField);
    mainFrame.add(depositButton);
    mainFrame.add(withdrawButton);
    mainFrame.add(balanceLabel);

    depositButton.addActionListener(this);
    withdrawButton.addActionListener(this);

    mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }

  @Override
  public void actionPerformed(ActionEvent e) {
    int amount = Integer.parseInt(inputField.getText());
    if (e.getActionCommand().equals("Deposit")) {
      balance += amount;
    } else {
      balance -= amount;
    }
    balanceLabel.setText("Current balance: " + balance);
    inputField.setText("");
  }

  public static void main(String[] args) {
    new Banking();
  }
```

```
}
```