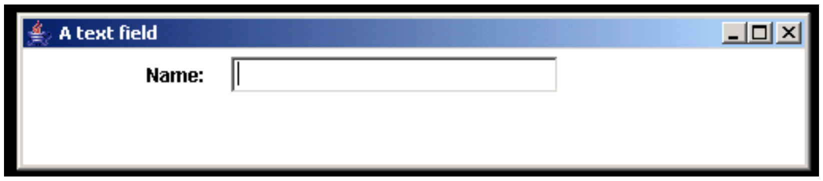
Class 21 Notes

Text Fields

A *text field* holds one line of text (a String) and can be used for input or output.



A label ("name") and a text field in a frame (The label is separate)

Here are the basics:
Don't skip over this

Class: JTextField

Generates: ActionEvent (like JButton) **when cursor is in the text field and a user presses "Enter."**

Listener: Must implement the ActionListener interface (like a JButton)

Listener method to implement: void actionPerformed(ActionEvent e) (like a JButton)

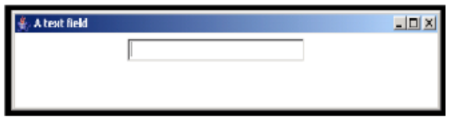
Register a listener : void addActionListener(ActionListener a) (Like a JButton)

Important: When you click on a button an An ActionEvent object is created. When you type something in a text field and hit "Enter" an ActionEvent is created

Constructors:

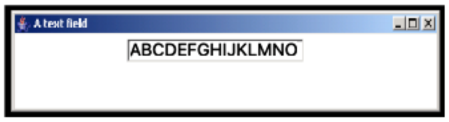
- JTextField(int numColumns)
  numColumns is the number of visible characters.

  JTextField line = new JtextField(25);
  // a text field where 25 characters are visible

- JTextField(String text)
  creates a JTextField object and initializes the text to text,

  JTextField line = new JTextField ("ABCDEFGHIJKLMNO");

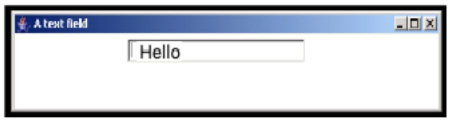
- JTextField(String text, int numColumns) creates a JTextField object with numColumns columns and initial text, text.

Methods:

The two methods in red are very important. We can retrieve data from a text field or place it in a text field using getText() and setText() are

- **void setText(String text)**
  places text in the text field.

```
JTextField line = new JTextField( 25)
line.setText("Hello")
```



Hello

- **String getText()**
  returns the text in a text field It is one String

```
JTextField line = new JTextField( "Hello")
 String s = line.getText(); // s is the string "Hello"
```

- void setEditable(boolean editable)
  if editable is false the string in the text field is read-only, i.e., it cannot be changed. The text field is grayed-out

- void setColums(int numColums)
  sets the number of characters that are displayed by the text field.

- int getColumns()
  returns the number of characters that are displayed by the text field.

- void setFont(Font font)
  set the font to font.

  ```
  TextField line = new JTextField( 25);
  line.setFont(new Font("Arial", Font.BOLD, 14));
  ```

- void setHorizontalAlignment (int alignment)
  alignment is JTextField.LEFT, JTextField.RIGHT, or JTextField.CENTER.
  The default is LEFT.

A Loan Calculator – JTextField in Action

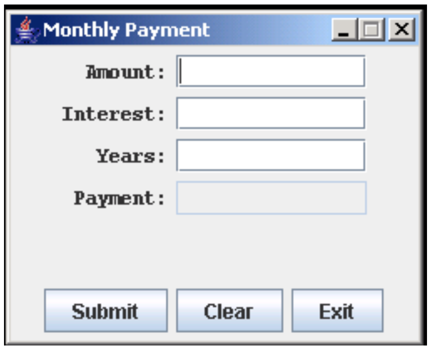
Example

Design an application that accepts the

- amount of a loan,
- the duration (in years) of the loan, and
- the yearly interest rate

and calculates the monthly payment.

Use three text fields for input, and an additional text field for output (payment).
Notice there is a label next to each text field.



A loan calculator GUI

- We will use a static method **getPayment (..)** (see below) that accepts the amount, the interest rate, and the years and returns the monthly payment. This is just a regular static method that I wrote.
- The formula for getting a payment is not important. Just take it as a black box that calculates your payment.
- getPayment(…)is a static method of the LoanPayment class so you call it as LoanPayment(50000, 5.25, 20). Here, it will give you the payment on a twenty year loan of $50,000 at 5.25%.

Here is the method – but the formula for computing the payment is not important for our purposes. Just think of it as you would think of Math.sqrt(..) or Math.sin(..) or SelectionSort.sort(..)etc. A useful method that any program can use.

```
public class LoanPayment
{
   public static  double getPayment( double amount, double interest, double years)
  {
     double payment =
                    amount*((interest/1200.0)/(1 - Math.pow(1 + interest/1200.0 , -years*12)));
     return(Math.round(payment*100))/100.00; // rounds to 2 decimal places
  }
}
```

So now we build the LoanCalculator class – it will use getPayment()

Set up the GUI first –

Look at the picture –

four labels and four text fields

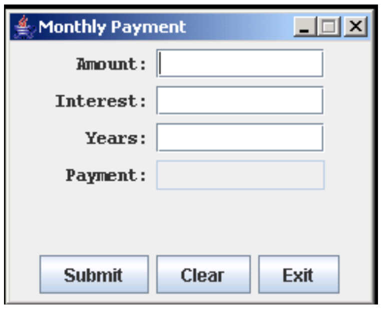
three buttons in the SOUTH

NOTE: The text fields and buttons must be declared as global and not in the constructor.
That is not necessary for the labels or the panels, since they are not active.

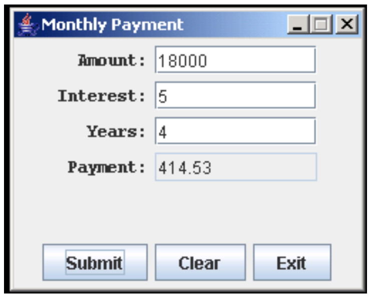
So here is the code that sets up the GUI. It seems like a lot but it is really pretty easy.

```
////////////// Loan Calculator Class //////////////

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

public class LoanCalculator extends JFrame  // look at the picturw
{
   // the global variables
    private JTextField amountField;
    private JTextField interestField;
    private JTextField yearsField;
    private JTextField paymentField;
    private JButton submitButton;
    private JButton clearButton;
    private JButton exitButton;

    public LoanCalculator()                    // constructor
    {
        super("Monthly Payment");
        setBounds(0,0,260,200);               // I had to play around with the size to make it look nice
        JPanel panel = new JPanel();       // for text fields and labels

        // make a label for each text field – setting the font
        //add  the labels and text fields to the panel

        JLabel amountLabel = new JLabel();
        amountLabel.setFont(new Font("Courier", Font.BOLD, 12));
        amountLabel.setText("  Amount:");
        amountField = new JTextField(10);
        panel.add(amountLabel);                              // place the  label in the panel
        panel.add(amountField);                                // place the text field in the panel

        JLabel interestLabel = new JLabel();
        interestLabel.setFont(new Font("Courier", Font.BOLD, 12));
        interestLabel.setText("Interest:");
        interestField = new JTextField(10);
        panel.add(interestLabel);
        panel.add(interestField);

        JLabel yearsLabel = new JLabel();
        yearsLabel.setFont(new Font("Courier", Font.BOLD, 12));
        yearsLabel.setText("   Years:");
        yearsField = new JTextField(10);
        panel.add(yearsLabel);
        panel.add(yearsField);

        JLabel paymentLabel = new JLabel();
        paymentLabel.setFont(new Font("Courier", Font.BOLD, 12));
        paymentLabel.setText(" Payment:");
        paymentField = new JTextField(10);
        panel.add(paymentLabel);
        panel.add(paymentField);
        paymentField.setEditable(false);            // read-only

        add(panel, BorderLayout.CENTER);                   // add the panel to the frame

        // add three buttons to the bottom of the frame

        JPanel buttonPanel = new JPanel(); // holds the buttons
        submitButton = new JButton("Submit"); // calculates
        exitButton = new JButton("Exit"); // ends application
        clearButton = new JButton("Clear"); // clears all fields
        buttonPanel.add(submitButton);  // add buttons to buttonPanel
        buttonPanel.add(clearButton);
        buttonPanel.add(exitButton);
        add(buttonPanel, BorderLayout.SOUTH);                 // add buttonPanel to bottom of frame


        setResizable(false);
        setVisible(true);

    }


  public static void main(String[] args)
  {
     LoanCalculator frame = new LoanCalculator();
  }

}
```

OK Here is the GUI



We now have to make it work.
Assume that the user has filled in the amount, Interest and Years.
The user clicks Submit and the program fills in the payment.
In this picture the user entered $18000, 5%, for 4 years, clicked Submit, and the payment field was filled in with 413.53

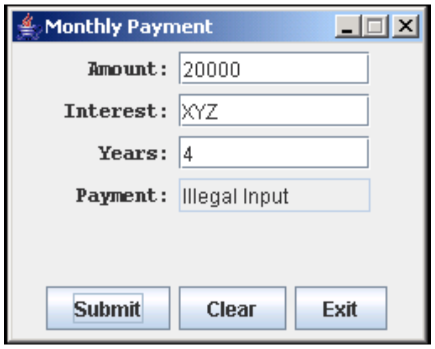


Amount: 18000
Interest: 5
Years: 4
Payment: 414.53

OK..so how does the Submit button work?
WE DO NOT RESPOND TO EVENTS OF THE TEXT FIELD. WE RESPOND TO BUTTON EVENTS . WE JUST RETRIEVE DATA FROM THE TEXT FIELD
This is how the listener works for the Submit button

- **Using the getText() method the amount, interest , and years are retrieved from the text boxes.**
- **These values are Strings so they must be converted to doubles (use Double.parseDouble())**
- **Then these three values are passed to the LoanPayment.getPayment(..) method**
- **The LoanPayment.getPayment(..) method returns the payment – a double**
- **And the payment (converted to a String), is put into the Payment text field using setText()**

There is one slight hitch…The data in the text boxes are Strings and must be converted to doubles by parseDouble(..). Suppose some clown enters rate of "XYZ" The program would crash. Very bad!!! So that has to be taken into account. As in the next picture..no crash, just a message-- "Illegal input"
.



Amount: 20000
Interest: XYZ
Years: 4
Payment: Illegal Input

***LoanCalculator*: with Illegal data**

We will use a try-catch to handle bad data. Remember that?
Now here is the program with the listener class for the buttons.
Remember : The listener is a private inner class and we must register the buttons with the listener.
The GUI stuff that we did above is printed in gray. The listener and the registration are in bold.
That is all that has been added – the listener and the registration

////////////// Loan Calculator Class //////////////

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

public class LoanCalculator extends JFrame
{
    private JTextField amountField;
    private JTextField interestField;
    private JTextField yearsField;
    private JTextField paymentField;
    private JButton submitButton;
    private JButton clearButton;
    private JButton exitButton;

    public LoanCalculator()                          // constructor
    {
        super("Monthly Payment");
        setBounds(0,0,250,200);
        JPanel panel = new JPanel();               // for text fields and labels

        // make a label for each text field
        //add  the labels and text fields to the panel

        JLabel amountLabel = new JLabel();
        amountLabel.setFont(new Font("Courier", Font.BOLD, 12));
        amountLabel.setText("  Amount:");
        amountField = new JTextField(10);
        panel.add(amountLabel);                    // place the  label in the panel
        panel.add(amountField);                              // place the text field in the panel

        JLabel interestLabel = new JLabel();
        interestLabel.setFont(new Font("Courier", Font.BOLD, 12));
        interestLabel.setText("Interest:");
        interestField = new JTextField(10);
        panel.add(interestLabel);
        panel.add(interestField);

        JLabel yearsLabel = new JLabel();
        yearsLabel.setFont(new Font("Courier", Font.BOLD, 12));
        yearsLabel.setText("   Years:");
        yearsField = new JTextField(10);
        panel.add(yearsLabel);
        panel.add(yearsField);

        JLabel paymentLabel = new JLabel();
        paymentLabel.setFont(new Font("Courier", Font.BOLD, 12));
        paymentLabel.setText(" Payment:");
        paymentField = new JTextField(10);
        panel.add(paymentLabel);
        panel.add(paymentField);
        paymentField.setEditable(false);                     // read-only

        add(panel, BorderLayout.CENTER);     // add the panel to the frame

        // add three buttons to the bottom of the frame

        JPanel buttonPanel = new JPanel(); // holds the buttons
        submitButton = new JButton("Submit"); // calculates
        exitButton = new JButton("Exit"); // ends application
        clearButton = new JButton("Clear"); // clears all fields
        buttonPanel.add(submitButton);  // add buttons to buttonPanel
        buttonPanel.add(clearButton);
        buttonPanel.add(exitButton);
        add(buttonPanel, BorderLayout.SOUTH);          // add buttonPanel to bottom of frame

        // register a listener with each button

        submitButton.addActionListener(new ButtonListener());
        clearButton.addActionListener(new ButtonListener());
        exitButton.addActionListener(new ButtonListener());

        setResizable(false);
        setVisible(true);
    }

    private class ButtonListener implements ActionListener         // interface
    {
        public void actionPerformed(ActionEvent e)                  // method of ActionListener
        {
            if ( e.getSource() == submitButton)                       // calculates payment
```

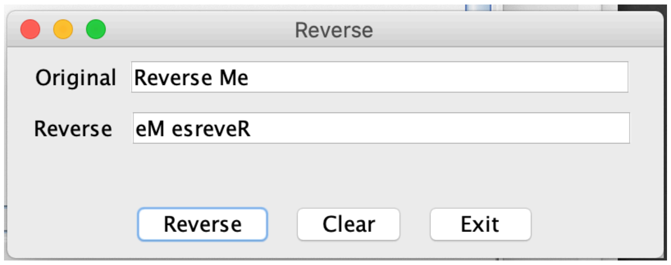
```
              try                // DoubleParseDouble() throws  NumberFormatException
              {
                 // retrieve data from the text fields; the data are Strings
                 // use Double.parseDouble(..) to convert the strings to numbers
                 double amount = Double.parseDouble(amountField.getText());
                 double interest = Double.parseDouble(interestField.getText());
                 double years = Double.parseDouble(yearsField.getText());

                 // Pass these values to the getPayment(..) method; it returns payment
                 double payment = LoanPayment.getPayment(amount, interest, years);

                 //  Use setText()  to put the payment in the payment text field
                // the text field requires a String,  so add “”  making a String a String

                 paymentField.setText(payment+"");
              }
              catch( NumberFormatException ex)      // if a text field has bad data
              {
                 paymentField.setText("Illegal Input");
              }

           else if (e.getSource() == clearButton)    //  clearButton clears all fields
           {
              amountField.setText("");
              interestField.setText("");
              yearsField.setText("");
              paymentField.setText("");
           }
           else
              System.exit(0);  // exit button
     }
   }

   public static void main(String[] args)
   {
      LoanCalculator frame = new LoanCalculator();
   }

 }

.
```

Example

Write an application with a GUI that displays a button labeled Reverse and two text fields. The first text field accepts a string, and the second displays the string in reverse. The reverse string should be displayed either when the cursor is in the first text field and the Enter key is pressed, or when the reverse button is clicked. That is, your listener must handle events generated by both the text field or the button.

Here is a screenshot

BTW I made the frame (400 x 150) and the text fields size 25.

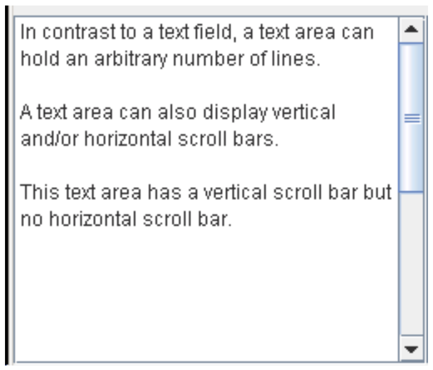


```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Reverse extends JFrame
{
  private JTextField s1;
  private JTextField s2;
  private JButton reverse;
  private JButton clear;
  private JButton exit;

  public Reverse()
  {
    super("Reverse");
    setBounds (50,50, 400,150);
    JLabel orig = new JLabel("Original");
    JLabel rev  = new JLabel("Reverse ");
    s1 = new JTextField(25);
    s2 = new JTextField (25);
    JPanel p = new JPanel();

    p.add(orig);  //the label
    p.add(s1);  // the first text field
    p.add(rev); // second label
    p.add(s2); // second text field
    add(p);  // add panel to the frame in CENTER
    J
Panel b = new JPanel(); f//or the buttons
 reverse = new JButton("Reverse");
 b.add(reverse);
 clear = new JButton("Clear");
 b.add(clear);
 exit = new JButton("Exit");
 b.add(exit);
 add(b, BorderLayout.SOUTH);   // add panel to frame


 // register the listeners
 reverse.addActionListener(new Handler());
 exit.addActionListener(new Handler());
 clear.addActionListener(new Handler());
 s1.addActionListener(new Handler());


 setVisible(true);
}

private class Handler implements ActionListener
{
  public void actionPerformed(ActionEvent e)
  {
    if (e.getSource() == reverse || e.getSource() == s1)// the button or the enter key in the text field
    {
      String s = s1.getText();  // the string in the text field
      StringBuilder sb = new StringBuilder(s);  // make a StringBuilder
      sb = sb.reverse();
      s2.setText(sb.toString());  // place the reversed string in the second text field
    }
    else if (e.getSource() == clear)
    {
     s1.setText("");
     s2.setText("");
    }
    else
     System.exit(0);
  }
}
public static void main(String[] args)
{
   Reverse r = new Reverse();
}
}
```

Text Areas

A text field holds a single line of data; a *text area* holds multiple lines.

The number of lines and the length of each line of a text area are defined in the constructor. Moreover, a text area can also display horizontal and vertical *scroll bars*, if desired.



In contrast to a text field, a text area can hold an arbitrary number of lines.

A text area can also display vertical and/or horizontal scroll bars.

This text area has a vertical scroll bar but no horizontal scroll bar.

A text area with a vertical scrollbar

Here are the basics:

Class: JTextArea
Generates: ActionEvent when a user presses "Enter" – like a text field
Listener: Must implement ActionListener.
Listener method to implement: void actionPerformed(ActionEvent e)
Register a listener : void addActionListener(ActionListener a)
Constructors:

- public JTextArea()
  instantiates a JTextArea object that displays no initial text.

- public JTextArea(String text)
  instantiates a JTextArea object that displays the string text.

- public JTextArea(**int rows, int cols**)
  instantiates a JTextArea object with rows rows and cols columns and displays no initial text.

- public JTextArea(**String text, int rows, int cols**)
  instantiates a JTextArea object with rows rows and cols columns and displays the string text.

The text Area holds one big string. Everything in a text area is just one String

The methods of JTextField are also applicable to JTextArea. Additionally, the following methods are also available:

- void append(String text)
  appends text to the end of a text area.
  area.append("Hello");

- void insert (String text, int place)
  inserts text at position place. Remember the text area holds one String
  area.insert("Hello", 23) // positions start with 0

- void replaceRange(String text, int start, int end)
  replaces the characters from position start to position end with text.
  area.replace("Hello" 10, 20); // replaces the characters from 10 to 19 with "hello"

- void setLineWrap(boolean wrap)
  If wrap is set to true, lines that exceed the allocated number of columns of a text area will wrap to the next line. The default is false.
  area.setLineWrap(true);

- void setRows(int rows)
  sets the number of rows of a text area to rows.

- int getRows()
  returns the number of visible rows.

In addition to the text area methods, JTextArea and JTextField inherit the following methods familiar to anyone who has used a word processor or text editor. These methods are inherited from JTextComponent.

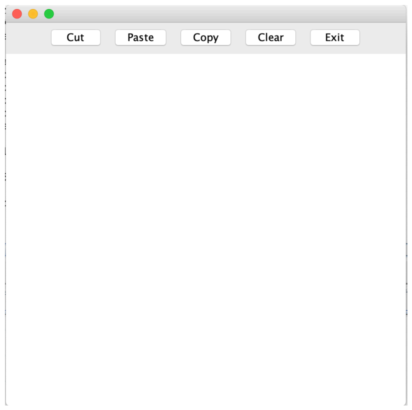
- void copy()
  copies selected text to the system clipboard. Text is selected as you normally select text using an editor or a word processor.

- void cut()
  removes the selected text from the text area (field) and moves the text to the system clipboard.

- void paste()
  places the contents of the system clipboard into the text area (field). If text in the component has been selected, that text is replaced. If text is not selected, the clipboard text is inserted at the position of the cursor.

- void selectAll()
  marks as selected all the text in the component.

OK…Let's set up the following GUI. A text area in CENTER and five buttons in the NORTH



Here is the code that does that:

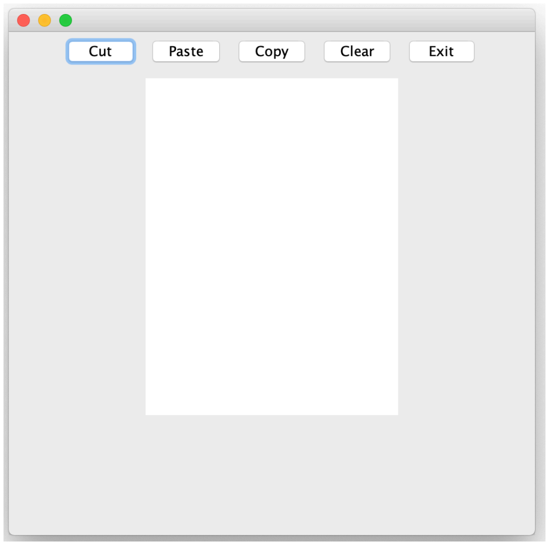
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TextAreas11 extends JFrame
{// global declarations
  private JTextArea area;
  private JButton cut;
  private JButton paste;
  private JButton copy
  private JButton clear;
  private JButton exit;

  public TextAreas11()
  {
    setBounds (50,50,500,500);

    area = new JTextArea();
    add(area);     // adds to CENTER by default fills the entire center

    JPanel buttonPanel = new JPanel();
    cut = new JButton("Cut");
    buttonPanel.add(cut);

    paste= new JButton("Paste");
    buttonPanel.add(paste);

    copy= new JButton("Copy");
    buttonPanel.add(copy);

    clear= new JButton("Clear");
    buttonPanel.add(clear);

    exit= new JButton("Exit");
    buttonPanel.add(exit);

    add(buttonPanel,BorderLayout.NORTH);
    setVisible(true);
  }

  public static void main(String[] args)
  {
      TextAreas11 ta = new TextAreas11();
    }
  }
```

Notice the text area (above) fills the entire CENTER region of the frame. We added the text area directly to the CENTER of the frame

If you do not want that you can set the size of the text area and put it in a panel and then the frame with this code:

```
                    JPanel textPanel = new JPanel();
                    area = new JTextArea(20,20);
                    textPanel.add(area);
                    add(textPanel);
```

and this is the picture



Now making the buttons work is easy. I will not repeat the whole GUI code again

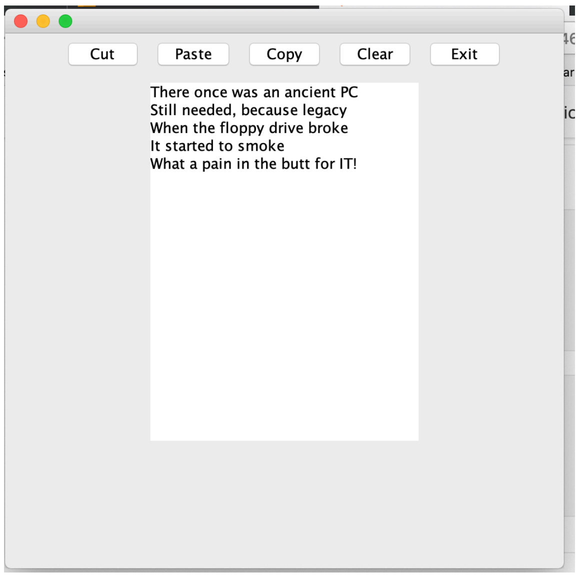
Here is just private listener class

```
private class ButtonListener implements ActionListener
  {
    public void actionPerformed(ActionEvent e)
    {
      if (e.getSource() == cut)
        area.cut();
      else if (e.getSource() == paste)
         area.paste();
      else if (e.getSource() ==copy)
         area.copy();
      else if (e.getSource() == clear)
          area.setText("");
      else System.exit(0);
    }
  }
```

And here is the registration that I added to the constructor

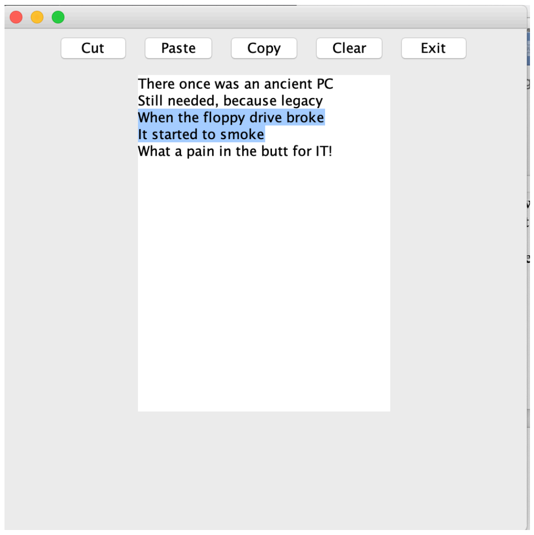
```
add(buttonPanel,BorderLayout.NORTH);
cut.addActionListener(new ButtonListener());
paste.addActionListener(new ButtonListener());
copy.addActionListener(new ButtonListener());
clear.addActionListener(new ButtonListener());
exit.addActionListener(new ButtonListener());
```

OK Now suppose I run the program and type or paste a limerick (which I did NOT make up) into the text area

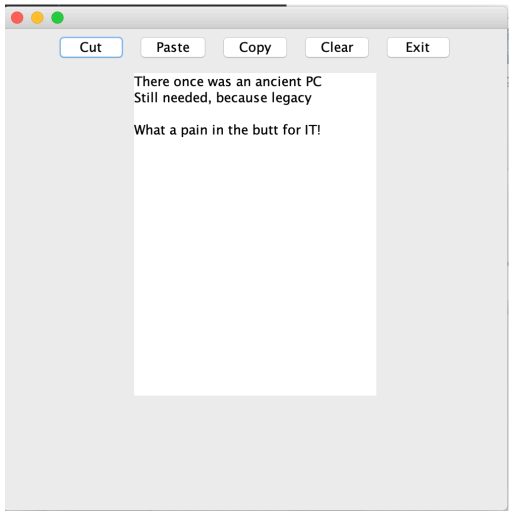


```
There once was an ancient PC
Still needed, because legacy
When the floppy drive broke
It started to smoke
What a pain in the butt for IT!
```

Now I select some text and click the cut button:

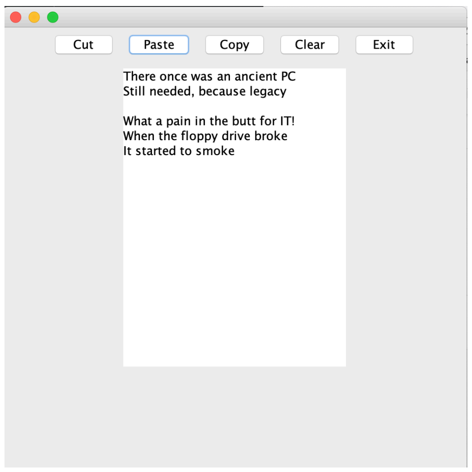


```
There once was an ancient PC
Still needed, because legacy
When the floppy drive broke
It started to smoke
What a pain in the butt for IT!
```



```
There once was an ancient PC
Still needed, because legacy

What a pain in the butt for IT!
```

Look! Now I will move the cursor to the end and click paste. There is the stuff I cut at the end
Like a mini editor.



```
There once was an ancient PC
Still needed, because legacy

What a pain in the butt for IT!
When the floppy drive broke
It started to smoke
```

See it is all very easy (I hope!).

Later we will see how to read and write a file into/from a text area and include scroll bars.
But this is enough for now

s