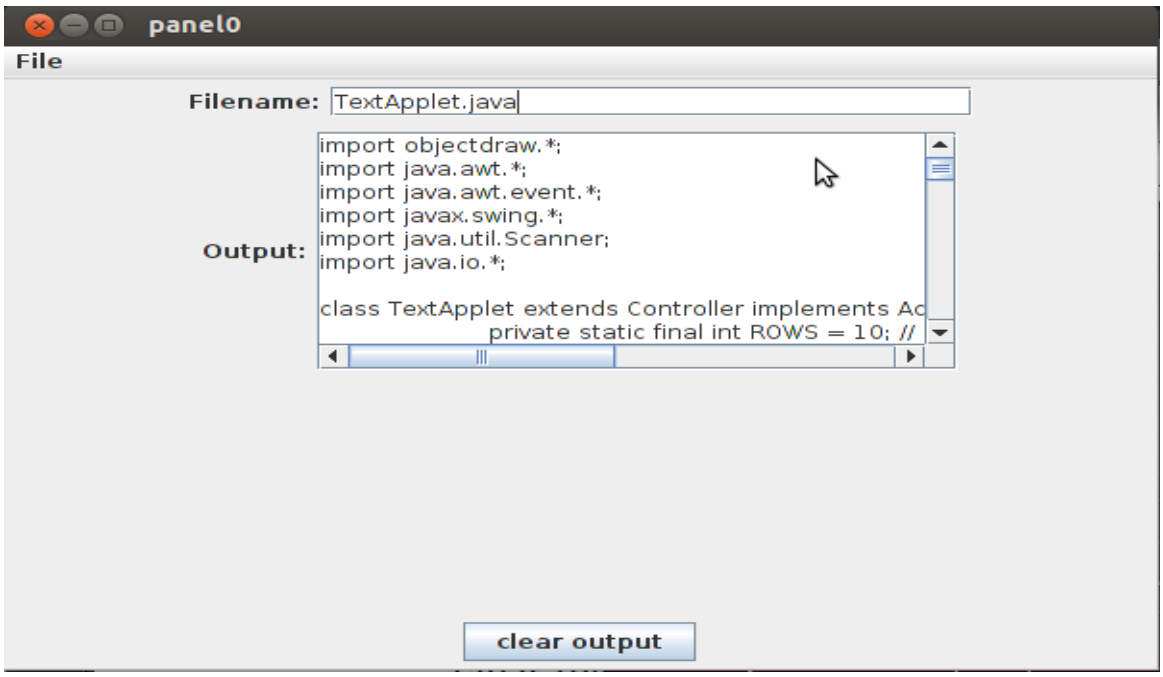
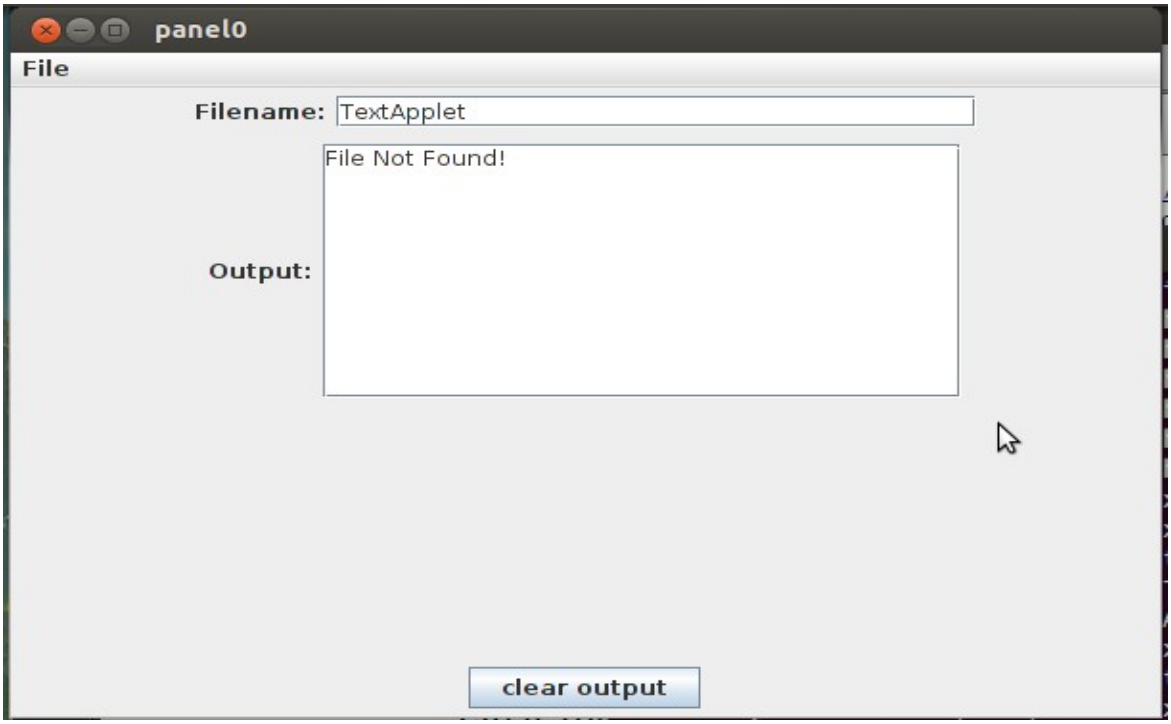
Homework/Program #5 Solutions

Problem #1 (20 points)
Using the standard Java Scanner class.
Look at http://natch3z.blogspot.com/2008/11/read-text-file-using-javautilscanner.html as an exampleof using the Scanner class and opening a file for reading. Modify the TextApplet example on page 296 in your book to do the following:
1. Change the applet to a program using the usual technique.
2. The text input box should be relabeled "Filename"
3. When you type a valid filename into the box and hit enter on your keyboard, your program should read the file indicated and display its contents in the JTextArea (scrollable text area next to "Output") using Scanner.
4. If the file is not found (this is what is called "throwing an exception"), the JTextArea should clear the contents and display "File Not Found!". This clear and display logic would replace the e.printStackTrace() in the blogspot example. Exceptions have not yet been covered in class,you should simply mimic the try { } except { } logic of the blogSpot example.
5. Pressing the "Clear Output" button should clear the JTextArea.

```
import objectdraw.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Scanner;
import java.io.*;

class TextApplet extends Controller implements ActionListener {
    private static final int ROWS = 10; // rows in TextArea
    private static final int COLS = 30; // cols in text field & area
    private JTextField Filename; // Input field
    private JTextArea output; // output area
    private JButton clear; // button to clear output
    public void begin(){
        Container contentPane = getContentPane();
        JPanel topPanel = new JPanel(); // prepare text field & label
        JLabel inLabel = new JLabel("Filename:");
        Filename = new JTextField(COLS);
        Filename.addActionListener(this);

        topPanel.add(inLabel);
        topPanel.add(Filename);
        contentPane.add(topPanel,BorderLayout.NORTH);

        JPanel centerPanel = new JPanel(); // prepare text area & label
        JLabel outLabel = new JLabel("Output:");
        output = new JTextArea(ROWS, COLS);
        output.setEditable(false);

        centerPanel.add(outLabel);
        centerPanel.add(new JScrollPane(output));
        contentPane.add(centerPanel,BorderLayout.CENTER);
```

```
            JPanel bottomPanel = new JPanel(); // create button
            clear = new JButton("clear output");
            clear.addActionListener(this);
            bottomPanel.add(clear);
            contentPane.add(bottomPanel,BorderLayout.SOUTH);
            validate();
        }
// add text to area if user hits return, else erase text area
        public void actionPerformed(ActionEvent evt){
            if (evt.getSource() == Filename){
              try {
                File file = new File(Filename.getText());
                Scanner scanner = new Scanner(file);
                scanner.useDelimiter(System.getProperty("line.separator"));
                while (scanner.hasNext()) {
                output.append(scanner.next()+"\n");
                }
                scanner.close();}
             catch (FileNotFoundException e) {
                output.setText("");
                  output.setText("File Not Found!");
                }
            } else { // clear button pressed
             output.setText("");
            }
        }
        public static void main(String args[]){
            new TextApplet().startController(600,400);
        }
}
```





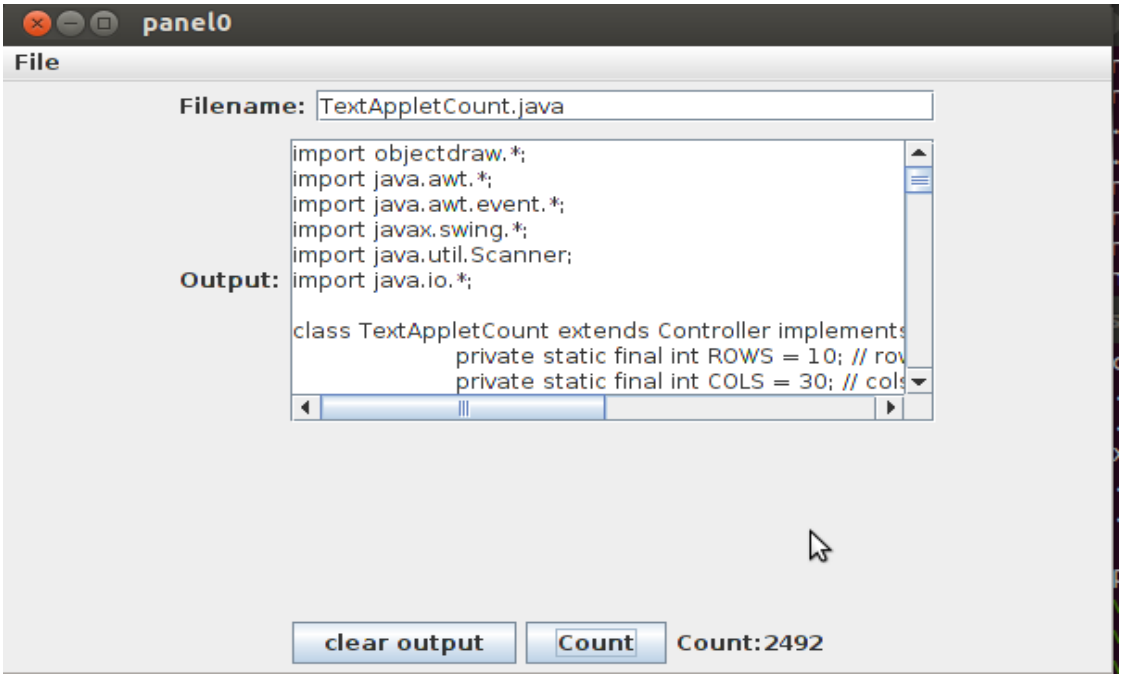
Problem #2 (10 points)
Modify your code from Problem #1 to (It is highly suggested to make it a different Program)
1. Add an additional button labeled “Count” placed next to the “clear output” button
2. Add an additional JLabel That will display the number of characters of the text that is currently displayed in the JTextArea. This label should be updated when the “Count” button is pressed by the user. You should use a panel with a FlowLayout for the buttons and label.

Include code
Include screen output when what is displayed is the java source code that answers this problem (i.e. your program). Your screenshot should be taken after the ”Count” button has been pressed.

```
import objectdraw.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Scanner;
import java.io.*;

class TextAppletCount extends Controller implements ActionListener {
    private static final int ROWS = 10; // rows in TextArea
    private static final int COLS = 30; // cols in text field & area
    private JTextField Filename; // Input field
    private JTextArea output; // output area
    private JButton clear; // button to clear output
    private JButton count;
    private JLabel countLabel;
    public void begin(){
        Container contentPane = getContentPane();
        JPanel topPanel = new JPanel(); // prepare text field & label
```

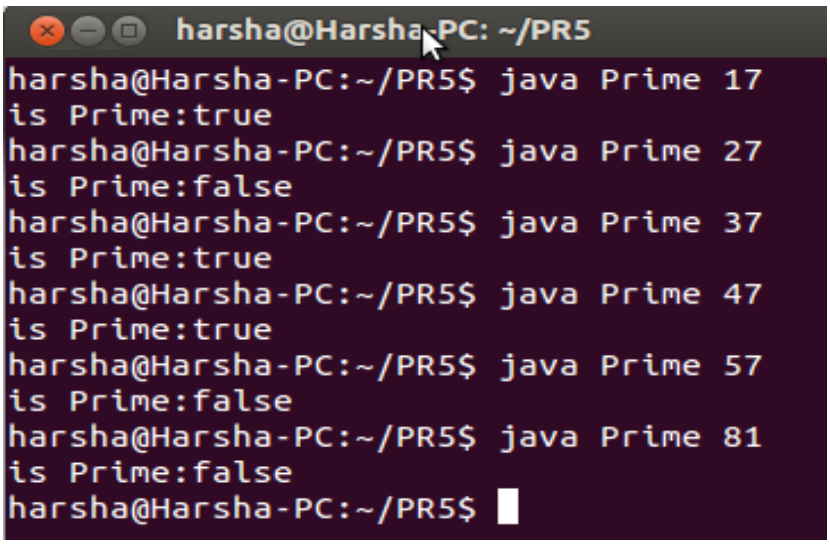
```
        JLabel inLabel = new JLabel("Filename:");
        Filename = new JTextField(COLS);
        Filename.addActionListener(this);

        topPanel.add(inLabel);
        topPanel.add(Filename);
        contentPane.add(topPanel,BorderLayout.NORTH);

        JPanel centerPanel = new JPanel(); // prepare text area & label
        JLabel outLabel = new JLabel("Output:");
        output = new JTextArea(ROWS, COLS);
        output.setEditable(false);

        centerPanel.add(outLabel);
        centerPanel.add(new JScrollPane(output));
        contentPane.add(centerPanel,BorderLayout.CENTER);

        JPanel bottomPanel = new JPanel(); // create button
        clear = new JButton("clear output");
        count = new JButton("Count");
          countLabel=new JLabel();
        clear.addActionListener(this);
          count.addActionListener(this);
          bottomPanel.setLayout(new FlowLayout());
        bottomPanel.add(clear);
          bottomPanel.add(count);
          bottomPanel.add(countLabel);
        contentPane.add(bottomPanel,BorderLayout.SOUTH);
        validate();
    }
// add text to area if user hits return, else erase text area
    public void actionPerformed(ActionEvent evt){
        if (evt.getSource() == Filename){
           try {
              File file = new File(Filename.getText());
              Scanner scanner = new Scanner(file);
              scanner.useDelimiter(System.getProperty("line.separator"));
              while (scanner.hasNext()) {
              output.append(scanner.next()+"\n");
              }
              scanner.close();}
          catch (FileNotFoundException e) {
              output.setText("");
                output.setText("File Not Found!");
             }
        } else if(evt.getSource()== count){
          countLabel.setText("Count:"+output.getDocument().getLength());
         }
        else { // clear button pressed
          output.setText("");
        }
    }
```

```
    public static void main(String args[]){
        new TextAppletCount().startController(600,400);
    }
}
```



Problem #3 (10 points)
JAEA Exercise 13.6.1
Modify so that the corrected isPrime method is invoked as part of main program called Prime. Prime takes an integer argument from the command line and properly determines if anumber is prime.
Include code
Include output of running Prime with input values of 17, 27, 37, 47, 57, 81

```
class Prime
{
    public static boolean isPrime(int n)
    {
        int divisor =n-1;
        boolean evenlyDivisible=false;
        while(divisor>1 && !evenlyDivisible){
            evenlyDivisible=((n%divisor)==0);
            divisor--;
        }
    return !evenlyDivisible;
    }

    public static void main(String args[])
    {
    if(args.length==1)
        System.out.println("is Prime:"+isPrime(Integer.parseInt(args[0])));
    else
        System.out.println("Incorrect Input!");
    }
}
```



Problem #4 (10 points) Fun with for loops

In each of the following, create a valid for(; ;) statement to achieve the desired results of the following two-line code segment:
for (; ;)
System.out.println(i);
example: print out integers from 1 to 10
ans: for (i = 1; i <= 10; i++)

(a) print out integers from 10 to 1
(b) print out 2k k=0,1,2,3, ... 20
(c) print out every seventh integer starting at 14 ending at 98
(d) print out the numbers from 1 to 41 that are NOT evenly divisible by 3
(e) print out the numbers 1,3,6,10,15,21,28,36,45,55 (Hint: int j = i = 1; is a legal statement that initializes both i and j to 1)

(a) for(i=10;i>=1;i--)
(b) for(i=1;i<=Math.pow(2,20);i=i*2) **alternate solution** for (i = 1; i <= 1 << 20; i*=2)
(c) for(i=14;i<=98;i=i+7)
(d) for(i=1;i<=41;i=i+i%3)
(e) for(int j=i=1;i<=55;j=j+1,i=i+j) **alternate solution** for(int j=i=1; j++ <= 10; i += j)

Problem #6 (10 points)
(a) what is the fundamental difference between do {... } while(); and a while() { } loops?
do {... } while() loop -- The condition is checked each time after executing the loop body.
While() { } loop – The condition is checked each time before executing the loop body.
A do {...} while() loop always executes the body of the loop at least once. A while() loop may never execute the body of the loop;

(b) Write a pair of nested for loops that print the multiplication tables for 1 through 10

```
for (int i=1;i<=10;i++)
{
    for(int j=1;j<=10;j++)
    {
        System.out.print(i*j+" ");
    }
```

```
    System.out.println();
}
```

(c) Look up the definition of the break; statement. What does the following code print out? (attempt this without compiling the code!)

```
int n = 0;
for (int i = 1; i <= 5; i++)
{
System.out.print( i + ": ");
for (int j = 1 ; j <= 5 ; j++)
{
n++;
if (j * i >= 15)
break;
System.out.print(j * i + " ");
}
System.out.println("");
}
System.out.println("Iterations: " + n);
```

```
1: 1 2 3 4 5
2: 2 4 6 8 10
3: 3 6 9 12
4: 4 8 12
5: 5 10
Iterations: 22
```

(d) Look up the definition of the continue; statement. What does the following code print out? (attempt this without compiling the code!)

```
int n = 0;
for (int i = 1; i <= 5; i++)
{
System.out.print( i + ": ");
for (int j = 1 ; j <= 5 ; j++)
{
n++;
if (j * i >= 15)
continue;
System.out.print(j * i + " ");
}
System.out.println("");
}
System.out.println("Iterations: " + n);
```

```
1: 1 2 3 4 5
2: 2 4 6 8 10
3: 3 6 9 12
4: 4 8 12
5: 5 10
Iterations: 25
```

(e) What is different about the two outputs? Explain why one line of code results in very similar but not identical output?

Number of iterations in case of break is 22 and incase of continue is 25
outputs are similar but not identical because

a) break will exit the loop
b) continue will skip the statements following continue and move to next iteration.