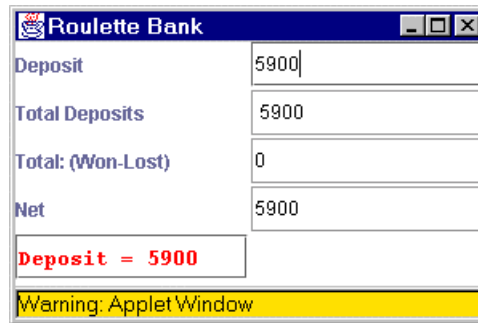


## Lecture 16

### The Bank

The first version of the bank will look like this:



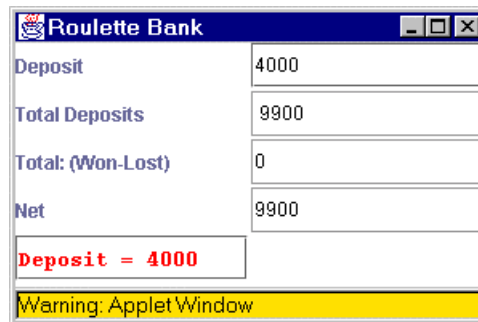
The screenshot shows a window titled "Roulette Bank" with a blue title bar. Inside the window, there is a table with four rows: "Deposit" with value 5900, "Total Deposits" with value 5900, "Total: (Won-Lost)" with value 0, and "Net" with value 5900. Below the table, there is a red text label "Deposit = 5900". At the bottom of the window, there is a yellow warning bar that says "Warning: Applet Window".

Deposit	5900
Total Deposits	5900
Total: (Won-Lost)	0
Net	5900

Deposit = 5900

Warning: Applet Window

The user will be able to make a deposit, and the result will be reflected in several fields. In addition if the user makes a second deposit all the relevant fields will be updated. If, after making the above deposit, the user adds an additional 4000 to his account the window will be updated:



The screenshot shows a window titled "Roulette Bank" with a blue title bar. Inside the window, there is a table with four rows: "Deposit" with value 4000, "Total Deposits" with value 9900, "Total: (Won-Lost)" with value 0, and "Net" with value 9900. Below the table, there is a red text label "Deposit = 4000". At the bottom of the window, there is a yellow warning bar that says "Warning: Applet Window".

Deposit	4000
Total Deposits	9900
Total: (Won-Lost)	0
Net	9900

Deposit = 4000

Warning: Applet Window

This construction of the class is straightforward. We need to add a variety of labels and fields to a container, construct several set() methods, and add an anonymous action listener to the first field.

The variables are all protected because, later on, we will want to extend the Bank class.

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

public class Bank1 extends JFrame
{
    protected Container c;
    protected GridLayout grid;
    protected JLabel label1, label2, label3, label4;
    protected JTextField field1, field2, field3, field4,
        display;
```

```

protected int deposit, total = 0, win = 0,
           lose = 0, net;

public Bank1()
{
    super("Roulette Bank");
    grid = new GridLayout(5,2,2,2);
    c = getContentPane();
    c.setLayout(grid);
    c.setBackground(Color.white);
    setFont(new Font("Serif", Font.BOLD, 20));

    label1 = new JLabel("Deposit");
    c.add(label1);
    field1 = new JTextField(3);
    field1.addActionListener
    (
        new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                deposit =
                Integer.parseInt(e.getActionCommand());
                total = total+deposit;
                display.setText("Deposit = "+deposit);
                field2.setText(" " + total);
                setNet();
                String s = Integer.toString(net);
                field4.setText(s);
            }
        }
    );
    c.add(field1);

    label2 = new JLabel("Total Deposits");
    c.add(label2);
    field2 = new JTextField(3);
    field2.setBackground(Color.white);
    field2.setText(Integer.toString(total));
    field2.setEditable(false);
    c.add(field2);

    label3 = new JLabel("Total: (Won-Lost)");
    c.add(label3);
    field3 = new JTextField(3);
    field3.setBackground(Color.white);
    field3.setEditable(false);
    field3.setText(Integer.toString(win-lose));
    c.add(field3);

    label4 = new JLabel("Net");
    c.add(label4);
    field4 = new JTextField(3);
    field4.setBackground(Color.white);
    setNet();
}

```

```

        field4.setText(Integer.toString(net));
        field4.setEditable(false);
        c.add(field4);

        display = new JTextField(3);
        display.setFont(new Font("Monospaced", Font.BOLD, 14));
        display.setForeground(new Color(255,0,0));
        display.setForeground(Color.red);
        c.add(display);

        setSize(300, 200);
        setLocation(25, 25);
        show();

} // end public Bank()

    public void setDeposit(int d)
    {
        deposit = d;
    }

    public void setWin(int w)
    {
        win = win+w;
    }

    public void setLose(int l)
    {
        lose = lose +l;
    }
    public void setNet()
    {
        net = total + win -lose;
    }
    public int getNet()
    {
        return net;
    }

} // end class Bank1

```

### Tokens, String Tokenizer

The bank class has a defect in that if the user enters anything but an integer in the deposit field the program will crash. We will add a method, whose signature is `protected boolean isPosInt(String s)`, to rectify this limitation.

The method employs a java class called `StringTokenizer` whose function is to remove all "white space" from a string. That is, given a string it remove all blank spaces, tab spaces, and carriage returns from a string.

The next application `Tokens` illustrates this.

```

import java.util.*;

public class Tokens
{
    public static void main(String args[])
    {
        String s = "    Over hills,  Over dales \n";
        s += "    We shall hit the dusty trails  ";

        System.out.println("\n" + s);

        StringTokenizer tokens =
        new StringTokenizer(s);

        int N = tokens.countTokens();

        System.out.println("\nNumber of tokens = "+ N);

        while(tokens.hasMoreTokens())
            System.out.println(tokens.nextToken());

        System.exit(0);
    }
}
} // end    class Tokens

```

The output is:

```

MS-DOS Command Prompt
Z:\11Pic20\CrapGame>java Tokens

    Over hills,  Over dales
    We shall hit the dusty trails

Number of tokens = 10
Over
hills,
Over
dales
We
shall
hit
the
dusty
trails

Z:\11Pic20\CrapGame>

```

Further details about StringTokenizer can be found on page 495 of your text.

## Characters

A user may type in an integer with leading or trailing white spaces; tokenizing the entry will rid the string of these extraneous characters. The job of deciding whether the one token that remains is done by examining its' characters.

A character is a primitive element in java, like int, double, or boolean. In addition, one refers to it by using single quote marks: '0' is the character 0.

There are two built-in methods we shall also need. If t is a string the t.charAt(i) is the i<sup>th</sup> character of t. in addition, isDigit(char c) is a static method of the class Character that determines if a given character c is '0', '1', .. or '9'.

We can now write the boolean method isPosInt.

```
protected boolean isPosInt(String s)
{
    boolean test = true;

    if ( s.equals("")) test = false;

    else
    {
        StringTokenizer tokens =
            new StringTokenizer(s);

        int m = tokens.countTokens();

        if (m > 1) test = false;
        else
        {
            String t = tokens.nextToken();

            char[] ch = new char[t.length()];

            for (int i = 0; i < t.length(); i++)
                ch[i] = t.charAt(i);

            if (!((Character.isDigit(ch[0])&& ch[0] != '0'))
                test = false;
            else
                for (int i = 1; i < t.length(); i++)
                    if (! Character.isDigit(ch[i])) test = false;

        }
        // end else
    }
    return test;
} // end    isPosInt(String s)
```

For the sake of reference, we list the complete program, so far, as Bank2.java. The changes due to the error checking for the Deposit entry are printed in red.

```

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

public class Bank2 extends JFrame

{
    protected Container c;
    protected GridLayout grid;
    protected JLabel label1, label2, label3, label4;
    protected JTextField field1, field2, field3, field4,
        display;

    protected int deposit, total = 0, win = 0,
        lose = 0, net;

    public Bank2()
    {
        super("Roulette Bank");
        grid = new GridLayout(5,2,2,2);
        c = getContentPane();
        c.setLayout(grid);
        c.setBackground(Color.white);
        setFont(new Font("Serif", Font.BOLD, 20));

        label1 = new JLabel("Deposit");
        c.add(label1);
        field1 = new JTextField(3);
        field1.addActionListener
        (
            new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    if( isPosInt(e.getActionCommand()) )
                    {
                        deposit =
                            Integer.parseInt(e.getActionCommand());
                        total = total+deposit;
                        display.setText("Deposit = "+deposit);
                        field2.setText(" " + total);
                        setNet();
                        String s = Integer.toString(net);
                        field4.setText(s);
                    }
                    else
                        display.setText("Must enter an\n integer > 0");
                }
            }
        );
        c.add(field1);

        label2 = new JLabel("Total Deposits");
        c.add(label2);
        field2 = new JTextField(3);

```

```

        field2.setBackground(Color.white);
        field2.setText(Integer.toString(total));
        field2.setEditable(false);
        c.add(field2);

        label3 = new JLabel("Total: (Won-Lost)");
        c.add(label3);
        field3 = new JTextField(3);
        field3.setBackground(Color.white);
        field3.setEditable(false);
        field3.setText(Integer.toString(win-lose));
        c.add(field3);

        label4 = new JLabel("Net");
        c.add(label4);
        field4 = new JTextField(3);
        field4.setBackground(Color.white);
        setNet();
        field4.setText(Integer.toString(net));
        field4.setEditable(false);
        c.add(field4);

        display = new JTextField(3);
        display.setFont(new Font("Monospaced", Font.BOLD, 14));
        display.setForeground(new Color(255,0,0));
        display.setForeground(Color.red);
        c.add(display);

        setSize(300, 300);
        setLocation(25, 25);
        show();

} // end public Bank()

public void setDeposit(int d)
{
    deposit = d;
}

public String getField1()
{
    return field1.getText();
}

public void setWin(int w)
{
    win = win+w;
}

public void setLose(int l)
{
    lose = lose +l;
}
public void setNet()
{
    net = total + win -lose;
}

```

```

    }
    public int getNet()
    {
    return net;
    }

    protected boolean isPosInt(String s)
    {
    boolean test = true;

    if ( s.equals("")) test = false;

    else
    {
    StringTokenizer tokens =
        new StringTokenizer(s);

    int m = tokens.countTokens();

    if (m > 1) test = false;
    else
    {
        String t = tokens.nextToken();

        char[] ch = new char[t.length()];

        for (int i = 0; i < t.length(); i++)
            ch[i] = t.charAt(i);

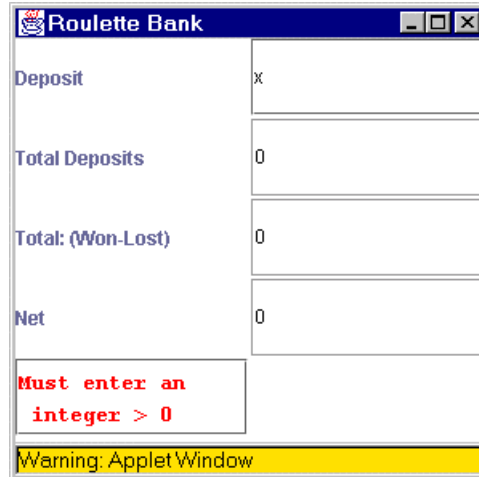
        if (!((Character.isDigit(ch[0])&& ch[0] != '0')))
            test = false;
        else
            for (int i = 1; i < t.length(); i++)
                if (! Character.isDigit(ch[i])) test = false;

        }
        // end else
    }
    return test;
} // end    isPosInt(String s)

} // end class Bank2

```

The output follows. You can check that entering anything but a positive integer produces an error message.



### Extending Bank to Craps

At this stage we have two classes, Bank2 and Craps3, but there is no connection between them. If we make Craps3 an extension of Bank2 by changing the lead line of Craps3 to

```
public class Craps3 extends Bank2 implements ActionListener
```

and test this out by means of the driver

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

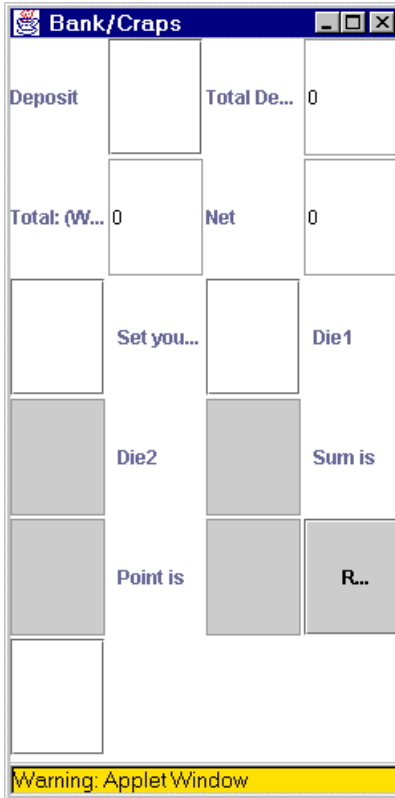
public class TestExCraps3 extends JApplet
{
    Craps3 game;

    public void init()
    {
        game = new Craps3();
    }
}
```

and the associated .html file

```
<html>
<applet code="TestExCraps3.class" height = 50 width = 500>
</applet>
</html>
```

we get the garbled output:



One way of solving this problem is as follows:

In the class Bank2

```
change the grid size to: grid = new GridLayout(11,2,2,2);
comment out:           //setSize(300, 600);
                       //setLocation(25, 25);
                       //show();
```

In the class Craps3

```
comment out:           //GridLayout grid;
change size           setSize(250,500);
and location:         setLocation(400,300);
```

This yields



This is not quite what is wanted; the labels and fields from the Craps3 class are out of phase. To get them in phase add an extra JTextField at the start of the class Craps3.

## Lab 7

Lab 7 starts at this point. The parts are given in lectures 15 and 16. Put them together and add the connecting tissue to get an applet that looks like and functions as the one on the class web site.

The specific conditions your program should meet are:

1. The output should be an 11 x 2 grid whose background colors, with one exception, are all white. The exception will be in the 5<sup>th</sup> row, 2<sup>th</sup> column, where the color will be gray.

2. The text fields in the second column and rows 2,3,4,5, 7,8,9,10 should not be editable; the text field in row 5, column 1 should also be not editable. The text field in row 11, column 2 may be either.

3. The text fields associated with "Deposit" and "Set your bet" should accept only positive integers. These entries may have leading and trailing white spaces. If an inadmissible entry is made the program should set out an error message.

### Input Errors, Error Messages

If a positive integer is not entered for Deposit field (row 1, column 2) then "Must enter an integer > 0" should appear in row 5, column 1.

If a positive integer is not entered for the Set your bet field (row 6, column 2) then the message "Input error reset bet" should appear at row 5, column 2. Furthermore, if this type of error occurs the "Roll Dice" button must be disabled.

If the size of the bet is greater than the net an error message should appear at row 5, column 2. Furthermore, if this type of error occurs the "Roll Dice" button must be disabled.

If an input error is made in the Deposit field in the course of the game, but the net is still greater than the bet the game continues.

4. Once the size of a bet has been set it need not, but may, be changed at the start of any game.

5. Once a particular game is under way the bet cannot be changed. It can be changed once a game is won or lost and before the first roll of any game.

### Names of Files to be Submitted

You may (and should) use different names for your classes at different stages of the programs development. However, once you have finished your work you should make the necessary definitions so that the class names are:

Bank.java, Bank.class, Bank\$1.class (this is produced automatically by the anonymous listener in Bank1)

Craps.java, Craps.class,

TestCraps.java, TestCraps.class, TestCraps.html

TestCraps.java is the driver for the applet. It is quite simple:

```
import javax.swing.*;

public class TestCraps extends JApplet
{
    Craps game;

    public void init()
    {
        game = new Craps();
    }
}
```

Lab 7 will be due Wednesday, February 28, at 5:30. It will count as a double lab.