

Lecture 15

The Game of "Craps"

In the game of "craps" a player throws a pair of dice. If the sum on the faces of the pair of dice after the first toss is 7 or 11 the player wins; if the sum on the first throw is 2, 3, or 12 he loses. If he tosses a 4,5,6,8,9, or 10 then this sum is called the his "point" and he continues rolling until he makes his point, and wins, or tosses a 7, and loses.

The final product we shall develop, as Lab 7, is on the web.

We will approach the development by building the parts, trying to combine the parts into one piece, finding difficulties in carrying out the combinations, and then resolving these problems.

Simulating the Game

We begin by constructing a class `Craps` for simulating the game. The "Bank" will be built later. The class will employ `JLabels`, `JTextFields`, a `JButton`, and will also implement the `ActionListener`.

The skeleton of `Craps` is as follows:

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

public class Craps extends JFrame implements ActionListener
{
    final int WON = 0, LOST = 1, CONTINUE = 2;

    boolean firstRoll = true;
    int sumOfDice = 0;
    int myPoint = 0;
    int gameStatus = CONTINUE;

    JLabel die1Label, die2Label, sumLabel, pointLabel,
           pointLabel, setBetLabel;

    JTextField firstDie, secondDie, sum, point, setBet,
              display;

    JButton roll;

    GridLayout grid;

    public Craps()
    {
        } // end Craps()

    public void actionPerformed(ActionEvent e)
```

```

    {
        play();

    } // end actionPerformed(ActionEvent e)

    public void play()
    {
    } // end play()

    public int rollDice()
    {
        return 1;
    } // end rollDice()

} // end Craps

```

This is the bare structure of the class. The two major parts to be written are the constructor `Craps()` and the method `play()`. Incidentally, the class as it now stands will compile; it will not compile without the `return 1` in `rollDice()`.

The constructor

This segment consists of building the constructor `Craps2()`. (The names will go from `Craps` to `Craps2`, to `Craps3`, etc in order to have a record of the different stages of the development of the program) As you can see, in the red print below, building the constructor consists of creating `JLabels` and `JTextFields` and adding them to the container.

```

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

public class Craps2 extends JFrame implements ActionListener
{
    final int WON = 0, LOST = 1, CONTINUE = 2;

    boolean firstRoll = true;
    int sumOfDice = 0;
    int myPoint = 0;
    int gameStatus = CONTINUE;

    JLabel die1Label, die2Label, sumLabel, pointLabel,
        setBetLabel;

    JTextField firstDie, secondDie, sum, point, setBet,
        display;

    JButton roll;

    GridLayout grid;

```

```

public Craps2()
{
    Container c = getContentPane();
    grid = new GridLayout(6,2,2,2);
    c.setLayout(grid);

    setBetLabel = new JLabel("  Set your bet");
    c.add(setBetLabel);
    setBet = new JTextField();
    setBet.addActionListener(this);
    c.add(setBet);

    die1Label = new JLabel("  Die1");
    c.add(die1Label);
    firstDie = new JTextField();
    firstDie.setEditable(false);
    c.add(firstDie);

    die2Label = new JLabel("  Die2");
    c.add(die2Label);
    secondDie = new JTextField();
    secondDie.setEditable(false);
    c.add(secondDie);

    sumLabel = new JLabel("  Sum is");
    c.add(sumLabel);
    sum = new JTextField();
    sum.setEditable(false);
    c.add(sum);

    pointLabel = new JLabel("  Point is");
    c.add(pointLabel);
    point = new JTextField();
    point.setEditable(false);
    c.add(point);

    roll = new JButton("  Roll Dice");
    roll.addActionListener(this);
    c.add(roll);

    display = new JTextField();
    c.add(display);

    setSize(200,300);
    setLocation(300,200);
    show();          // <---- Be sure to show your work
} // end Craps2()

```

```

public void actionPerformed(ActionEvent e) //2 the action
{
    play();

} //end actionPerformed(ActionEvent e)

public void play() //3 implementing the action
{
} // end play()

public int rollDice() //4 the simplest part
{
    return 1;
} // end rollDice()
} // end Craps2

```

To run this class as an applet use the simple applet below:

```

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

public class TestCraps2 extends JApplet
{
    Craps2 game;

    public void init()
    {
        game = new Craps2();
    }
}

```

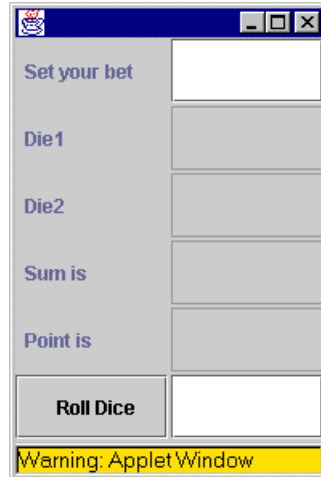
Of course you will need the following file, `TestCraps2.html`, to run the applet.

```

<html>
<applet code="TestCraps2.class" height = 50 width = 500>
</applet>
</html>

```

When is `TestCraps2` compiled and viewed by means of the appletviewer the following appears on the screen:



The background colors above are the ones that appear when no special efforts are made to produce particular colors. In general, labels and fields that cannot be edited will be gray; fields that can be edited will be white.

So, all we have so far is the grid that will hold our results.

Playing the game

For the time being we shall ignore the question of the size of the bet and concentrate on playing the game. Consequently the `actionPerformed()` method will have the simple form:

```
public void actionPerformed(ActionEvent e)
{
    play();
} //end actionPerformed(ActionEvent e)
```

Of course, writing `play()` is the major problem at this point. We start with a simple method for simulating one toss of a pair of dice:

```
public int rollDice()
{
    int die1, die2, workSum;
    die1 = 1 + (int)(6*Math.random());
    die2 = 1 + (int)(6*Math.random());
    workSum = die1 + die2;

    firstDie.setText(Integer.toString(die1));
    secondDie.setText(Integer.toString(die2));
    sum.setText(Integer.toString(workSum));

    return workSum;
} // end rollDice()
```

There is nothing new here.

The play() method does contain a new idea, mainly switch(sumOfDice) ,which is a variation of the if-then construction.

```
public void play()
{
    if(firstRoll)
    {
        sumOfDice = rollDice();

        switch(sumOfDice)
        {
            case 7: case 11:
                gameStatus = WON;
                point.setText("");
                break;

            case 2: case 3: case 12:
                gameStatus = LOST;
                point.setText("");
                break;

            default:
                gameStatus = CONTINUE;
                myPoint = sumOfDice;
                point.setText(Integer.toString(myPoint));
                firstRoll = false;
                break;

        } // end switch
    } // end if(firstRoll)

    else
    {
        sumOfDice = rollDice();

        if(sumOfDice == myPoint) gameStatus = WON;

        if(sumOfDice == 7) gameStatus = LOST;

    } //end else

    if(gameStatus == CONTINUE)
        display.setText("Roll Again");

    if(gameStatus == WON)
    {
        display.setText(" Win!. Roll\n to play again");
        firstRoll = true;
    }

    if(gameStatus == LOST)
    {
        display.setText(" Player Loses\n Play again");
        firstRoll = true;
    }

} // end play
```

In general `switch()` is an integer-valued method is a method whose parameter is also an integer. The fragment

```
        switch(sumOfDice)
        {
            case 7: case 11:
                gameStatus = WON;
                point.setText("");
                break;
```

means if `sumOfDice` is 7 or 11 then `gameStatus = WON`.

For the record, the complete program at this point, `Craps3`, is:

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

public class Craps3 extends JFrame implements ActionListener
{
    final int WON = 0, LOST = 1, CONTINUE = 2;

    boolean firstRoll = true;
    int sumOfDice = 0;
    int myPoint = 0;
    int gameStatus = CONTINUE;

    JLabel die1Label, die2Label, sumLabel, pointLabel, setBetLabel;
    JTextField firstDie, secondDie, sum, point, setBet, display;
    JButton roll;

    GridLayout grid;

    public Craps3()
    {
        Container c = getContentPane();
        grid = new GridLayout(6, 2, 2, 2); // 1
        c.setLayout(grid);

        setBetLabel = new JLabel(" Set your bet");
        c.add(setBetLabel);
        setBet = new JTextField();
        setBet.addActionListener(this); // 2
        c.add(setBet);

        die1Label = new JLabel(" Die1");
        c.add(die1Label);
        firstDie = new JTextField();
        firstDie.setEditable(false);
        c.add(firstDie);

        die2Label = new JLabel(" Die2");
        c.add(die2Label);
        secondDie = new JTextField();
        secondDie.setEditable(false);
```

```

c.add(secondDie);

sumLabel = new JLabel(" Sum is");
c.add(sumLabel);
sum = new JTextField();
    sum.setEditable(false);
c.add(sum);

pointLabel = new JLabel(" Point is");
c.add(pointLabel);
point = new JTextField();
    point.setEditable(false);
c.add(point);

    roll = new JButton(" Roll Dice");
    roll.addActionListener(this);
c.add(roll);

    display = new JTextField();

    c.add(display);

setSize(200,300);
setLocation(300,200);
show();
} // end Craps3()

public void actionPerformed(ActionEvent e)
{
play();

} //end actionPerformed(ActionEvent e)

public void play()
{
if(firstRoll)
{
    sumOfDice = rollDice();

switch(sumOfDice)
{
    case 7: case 11:
        gameStatus = WON;
        point.setText("");
        break;

    case 2: case 3: case 12:
        gameStatus = LOST;
        point.setText("");
        break;

    default:
        gameStatus = CONTINUE;
        myPoint = sumOfDice;
        point.setText(Integer.toString(myPoint));
}
}
}

```

```

        firstRoll = false;
        break;

    } // end switch
} // end if(firstRoll)

else
{
    sumOfDice = rollDice();

    if(sumOfDice == myPoint) gameStatus = WON;

    if(sumOfDice == 7) gameStatus = LOST;

} //end else

if(gameStatus == CONTINUE)
    display.setText("Roll Again");

if(gameStatus == WON)
{
    display.setText(" Win!. Roll\n to play again");
    firstRoll = true;
}

if(gameStatus == LOST)
{
    display.setText(" Player Loses\n Play again");
    firstRoll = true;
}

} // end play

public int rollDice()
{
    int die1, die2, workSum;
    die1 = 1 + (int)(6*Math.random());
    die2 = 1 + (int)(6*Math.random());
    workSum = die1 + die2;

    firstDie.setText(Integer.toString(die1));
    secondDie.setText(Integer.toString(die2));
    sum.setText(Integer.toString(workSum));

    return workSum;
} // end rollDice()

} // end Craps3

```

Two supplements are needed to test it. First, an applet:

```

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

public class TestCraps3 extends JApplet

```

```

{
    Craps3 game;

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

```

Then a TestCraps3.html file:

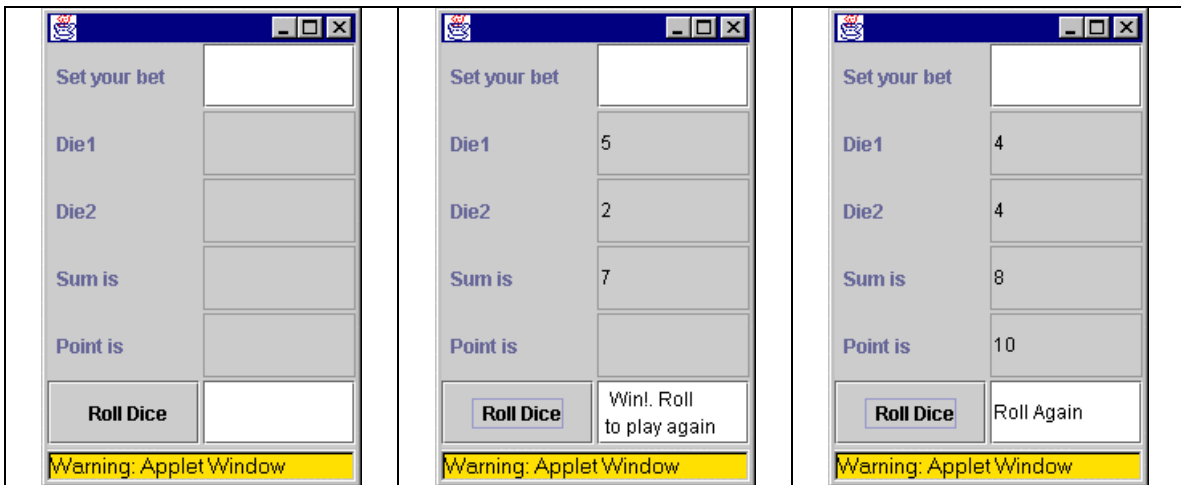
```

<html>
<applet code="TestCraps3.class" height = 50 width = 500>
</applet>
</html>

```

When the program is run a window like the one in the first panel below appears. The player then clicks on the Roll Dice button. If he makes a 7 on the first toss, he wins, as in the second window. If he tosses a 10 on the first roll, as in third panel, the game continues.

The setting of bets and accompanying bookkeeping have yet to be implemented.



Setting Colors and Fonts for Labels, TextFields, and Buttons

There are several ways to go about changing colors and fonts on components.

First, you can go to the beginning of the constructor and add the line `c.setBackground(Color.white)`:

```
public Craps3()
{
    Container c = getContentPane();
    grid = new GridLayout(6,2,2,2);
    c.setLayout(grid);
    c.setBackground(Color.white);
    .
}
```

When this is done most of the components will have a white background. Some, such as buttons and text fields that are not editable will have a blue-gray background.

Second, you can work with the individual components. In the case below the background has been set to white, the foreground (the text) has been set to red, and the font has been set to SansSerif, bold, 16 point.

```
pointLabel = new JLabel(" Point is");
c.add(pointLabel);
point = new JTextField();
point.setBackground(Color.white);
point.setForeground(Color.red);
point.setFont(new Font("SansSerif", Font.BOLD, 16));
point.setEditable(false);
c.add(point);
```