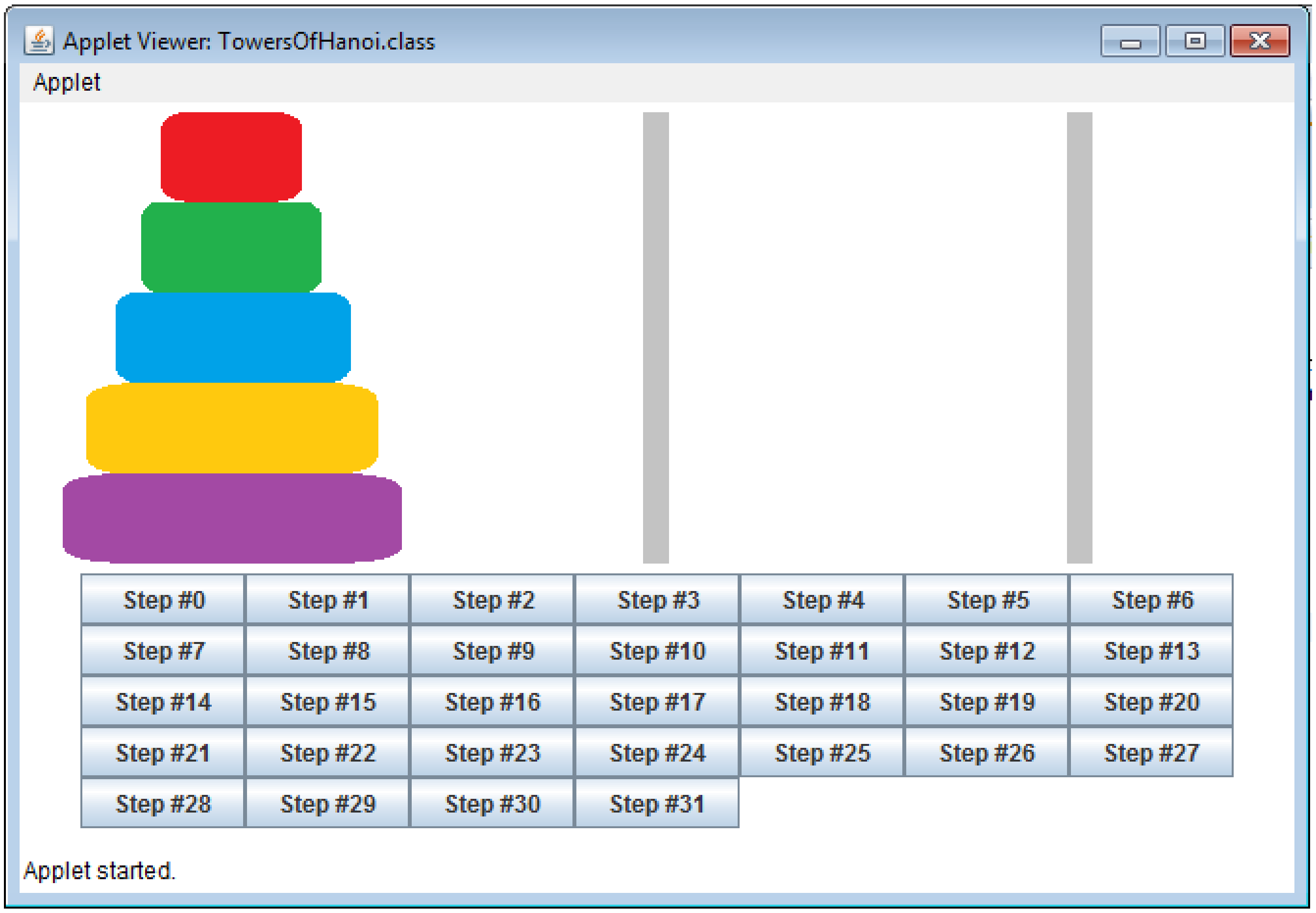
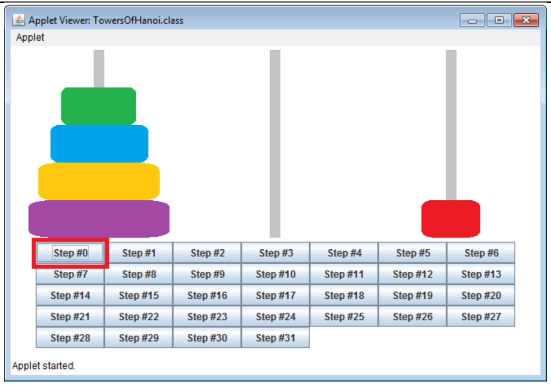
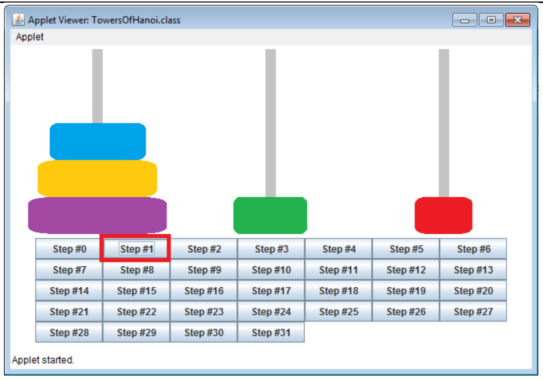
Towers of Hanoi



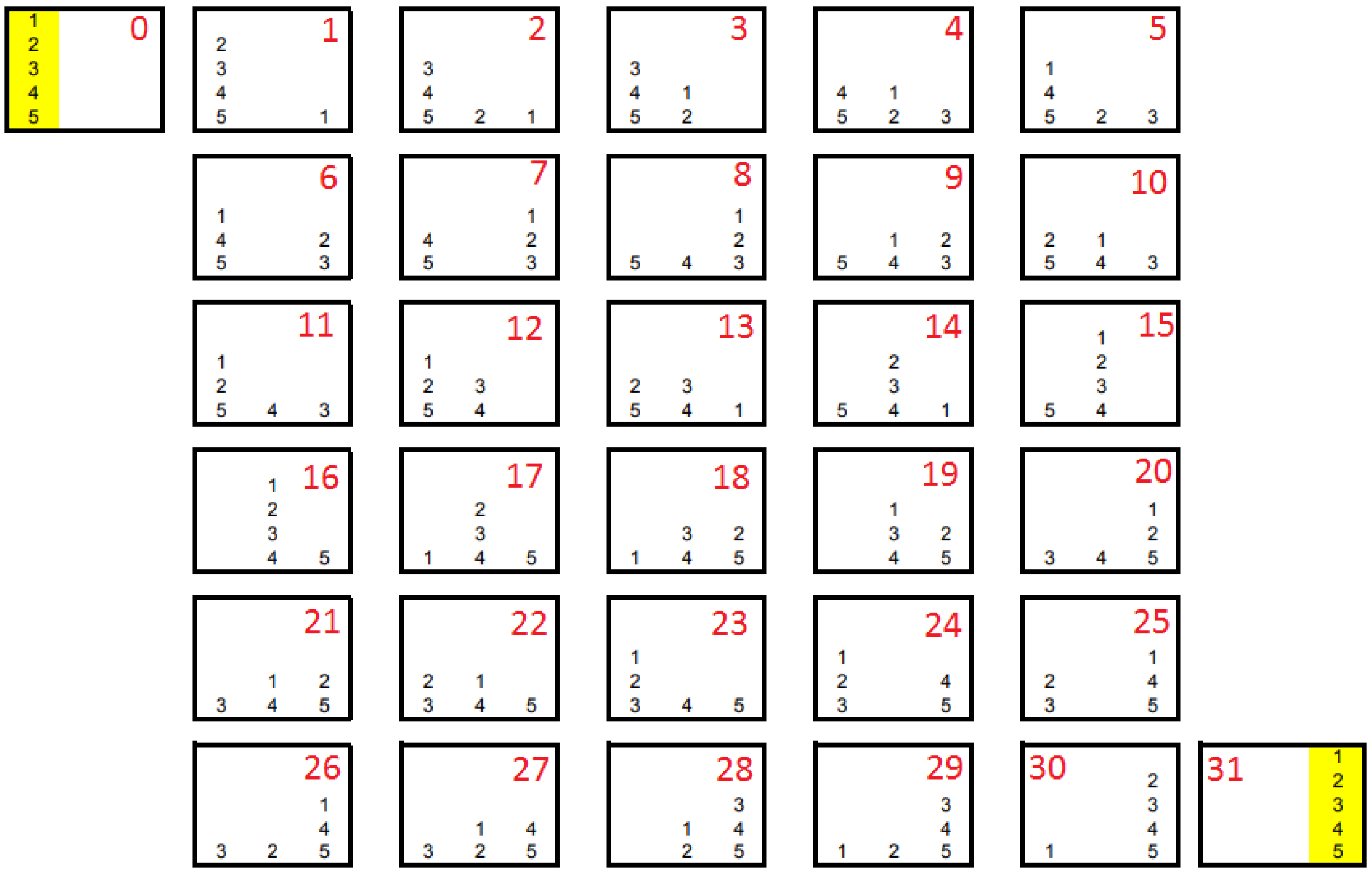
Applet started.

Recall that the goal of the Towers of Hanoi is to move all the disks from the leftmost peg to the rightmost peg, adhering to the following rules: Move only one disk at a time. A larger disk may not be placed on top of a smaller disk.

Code the Towers of Hanoi Game so that each button shows one step towards the final solution.

 `s3.push(s1.pop());`	 `s2.push(s1.pop());`

Here are each of the steps to help you with the actionPerformed:



Starter Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
import java.util.*;
public class TowersOfHanoi extends Applet implements ActionListener
{
    JLabel stackPic[] = new JLabel [15];
    Stack s1 = new Stack ();
    Stack s2 = new Stack ();
    Stack s3 = new Stack ();
    public void init ()
    {

        resize (650, 380);
        Panel p = new Panel (new GridLayout (5, 3));
        for (int i = 0 ; i < 15 ; i++)
        {
            stackPic [i] = new JLabel (createImageIcon ("0.png"));
            stackPic [i].setPreferredSize (new Dimension (216, 46));
            p.add (stackPic [i]);
        }
        add (p);
        JButton b[] = new JButton [32];
        Panel p2 = new Panel (new GridLayout (5,7));
        int num = 1;
        //each button has one step
        for (int i = 0 ; i < 32 ; i ++)
        {
               b [i] = new JButton ("Step #" + i);
               b [i].setActionCommand ("" + i);
               b [i].addActionListener (this);
               p2.add (b [i]);
        }
        add (p2);

        s1.push("5");
        s1.push("4");
        s1.push("3");
        s1.push("2");
        s1.push("1");
        viewAllStacks ();

    }
    public void actionPerformed (ActionEvent e)
    {
        if (e.getActionCommand ().equals ("0")) {
               s3.push(s1.pop());
        }
        else if (e.getActionCommand ().equals ("1")) {
               s2.push(s1.pop());
        }
        else if (e.getActionCommand ().equals ("2")) {
               s2.push(s3.pop());
        }
        else if (e.getActionCommand ().equals ("3")) {
               s3.push(s1.pop());
        }
        else if (e.getActionCommand ().equals ("4")) {
        }
        else if (e.getActionCommand ().equals ("5")) {
        }
        else if (e.getActionCommand ().equals ("6")) {
        }
        else if (e.getActionCommand ().equals ("7")) {
        }
        else if (e.getActionCommand ().equals ("8")) {
        }
        else if (e.getActionCommand ().equals ("9")) {
        }
        else if (e.getActionCommand ().equals ("10")) {
        }
        else if (e.getActionCommand ().equals ("11")) {
        }
        else if (e.getActionCommand ().equals ("12")) {
        }
        else if (e.getActionCommand ().equals ("13")) {
        }
        else if (e.getActionCommand ().equals ("14")) {
        }
        else if (e.getActionCommand ().equals ("15")) {
        }
        else if (e.getActionCommand ().equals ("16")) {
        }
        else if (e.getActionCommand ().equals ("17")) {
        }
        else if (e.getActionCommand ().equals ("18")) {
        }
        else if (e.getActionCommand ().equals ("19")) {
        }
        else if (e.getActionCommand ().equals ("20")) {
        }
        else if (e.getActionCommand ().equals ("21")) {
        }
        else if (e.getActionCommand ().equals ("22")) {
        }
        else if (e.getActionCommand ().equals ("23")) {
        }
        else if (e.getActionCommand ().equals ("24")) {
        }
        else if (e.getActionCommand ().equals ("25")) {
        }
        else if (e.getActionCommand ().equals ("26")) {
        }
        else if (e.getActionCommand ().equals ("27")) {
        }
        else if (e.getActionCommand ().equals ("28")) {
        }
        else if (e.getActionCommand ().equals ("29")) {
        }
        else if (e.getActionCommand ().equals ("30")) {
        }
        else if (e.getActionCommand ().equals ("31")) {
        }
        else if (e.getActionCommand ().equals ("32")) {
        }
        viewAllStacks ();
}
public void clearAllStacks ()
{
        s1.clear ();
        s2.clear ();
        s3.clear ();
}
public void viewAllStacks ()
{
        for (int i = 0 ; i < 15 ; i++)
                stackPic [i].setIcon (createImageIcon ("0.png"));
        drawStack (s1, 12);
        drawStack (s2, 13);
        drawStack (s3, 14);
}
```

```java
        public void drawStack (Stack s, int start)
        {
                Stack s4 = new Stack ();
                while (!s.isEmpty ())
                        s4.push (s.pop ());
                while (!s4.isEmpty ())
                {
                        String value = (String) s4.pop ();
                        stackPic [start].setIcon (createImageIcon (value + ".png"));
                        s.push (value);
                        start -= 3;
                }
        }
        protected static ImageIcon createImageIcon (String path)
        {
                java.net.URL imgURL = TowersOfHanoi.class.getResource (path);
                if (imgURL != null)
                {
                        return new ImageIcon (imgURL);
                }
                else
                {
                        System.err.println ("Couldn't find file: " + path);
                        return null;
                }
        }
} //end applet
```