Flyweight Design Pattern Tutorial

Posted by Derek Banas on Oct 12, 2012 in Java Video Tutorial | 8 comments

Welcome to my Flyweight Design Pattern Tutorial! The flyweight design pattern is used to dramatically increase the speed of your code when you are using many similar objects.

To reduce memory usage the flyweight design pattern shares Objects that are the same rather than creating new ones. In this tutorial, I'll create 100,000 rectangles and show you the difference in speed versus creating unique rectangle objects versus the flyweight design. The code follows to help you learn.

If you like videos like this it helps to tell Google [googleplusone]

Share if you know anyone who would be interested

Code from the Video

FLYWEIGHTTEST.JAVA

```
001 // The Flyweight design pattern is used when you need to
002 // create a large number of similar objects
003
004 // To reduce memory this pattern shares Objects that are
005 // the same rather than creating new ones
006
007 import javax.swing.*;
008
009 import java.awt.BorderLayout;
010 import java.awt.Color;
011 import java.awt.Graphics;
012
013 import java.awt.event.ActionEvent;
014 import java.awt.event.ActionListener;
015 import java.util.Random;
016
017 public class FlyWeightTest extends JFrame{
018
019     private static final long serialVersionUID = 1L;
020
021     JButton startDrawing;
022
023     int windowWidth = 1750;
024     int windowHeight = 1000;
025
026     // A new rectangle is created only if a new color is needed
```

```
027
028     Color[] shapeColor = {Color.orange, Color.red, Color.yellow,
029             Color.blue, Color.pink, Color.cyan, Color.magenta,
030             Color.black, Color.gray};
031
032     public static void main(String[] args){
033
034         new FlyWeightTest();
035
036     }
037
038     public FlyWeightTest(){
039
040         // Create the frame, position it and handle closing it
041
042         this.setSize(windowWidth,windowHeight);
043         this.setLocationRelativeTo(null);
044         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
045         this.setTitle("Flyweight Test");
046
047         JPanel contentPane = new JPanel();
048
049         contentPane.setLayout(new BorderLayout());
050
051         final JPanel  drawingPanel  = new JPanel();
052
053         startDrawing = new JButton("Button 1");
054
055         contentPane.add(drawingPanel,  BorderLayout.CENTER);
056
057         contentPane.add(startDrawing, BorderLayout.SOUTH);
058
059         startDrawing.addActionListener(new ActionListener() {
060
061             public void actionPerformed(ActionEvent event) {
062                 Graphics g = drawingPanel.getGraphics();
063
064
065                 long startTime = System.currentTimeMillis();
066
067
068                 for(int i=0; i < 100000; ++i) {
069
070                     //
071                     // Uses rectangles stored in the HashMap to
072                     // speed up the program
073
074                     MyRect rect = RectFactory.getRect(getRandColor());
075                     rect.draw(g, getRandX(), getRandY(),
076                             getRandX(), getRandY());
077
078                     //
079                     /*
080                     MyRect rect = new MyRect(getRandColor(), getRandX(),
```

```
getRandY(), getRandX(), getRandY());
                    rect.draw(g);
                    */


                    //
                    /*
                    g.setColor(getRandColor());
                    g.fillRect(getRandX(), getRandY(), getRandX(),
getRandY());
                    */


                }

                long endTime = System.currentTimeMillis();

                System.out.println("That took " + (endTime - startTime) +
" milliseconds");

            }
          });

        this.add(contentPane);

        this.setVisible(true);

    }

    // Picks random x & y coordinates

    private int getRandX(){ return (int)(Math.random()*windowWidth); }

    private int getRandY(){ return (int)(Math.random()*windowHeight); }

    // Picks a random Color from the 9 available

    private Color getRandColor(){
        Random randomGenerator = new Random();

        int randInt = randomGenerator.nextInt(9);

        return shapeColor[randInt];

    }

}
```

MYRECT.JAVA

```
import java.awt.*;
public class MyRect {
   private Color color = Color.black;
   private int x, y, x2, y2;
```

```
05
06    public MyRect(Color color) {
07
08        this.color = color;
09
10    }
11
12    public void draw(Graphics g, int upperX, int upperY, int lowerX, int
lowerY) {
13            g.setColor(color);
14            g.fillRect(upperX, upperY, lowerX, lowerY);
15    }
16
17    /* Original forces creation of a rectangle every time
18
19    public MyRect(Color color, int upperX, int upperY, int lowerX, int
lowerY) {
20        this.color = color;
21        this.x = upperX;
22        this.y = upperY;
23        this.x2 = lowerX;
24        this.y2 = lowerY;
25    }
26
27    public void draw(Graphics g) {
28        g.setColor(color);
29        g.fillRect(x, y, x2, y2);
30    }
31    */
32 }
```

RECTFACTORY.JAVA

```
01 // This factory only creates a new rectangle if it
02 // uses a color not previously used
03
04 // Intrinsic State: Color
05 // Extrinsic State: X & Y Values
06
07 import java.util.HashMap;
08 import java.awt.Color;
09 public class RectFactory {
10
11     // The HashMap uses the color as the key for every
12     // rectangle it will make up to 8 total
13
14     private static final HashMap<Color, MyRect> rectsByColor = new
HashMap<Color, MyRect>();
15
16     public static MyRect getRect(Color color) {
17         MyRect rect = (MyRect)rectsByColor.get(color);
18
19         // Checks if a rectangle with a specific
20         // color has been made. If not it makes a
```

```
21          // new one, otherwise it returns one made already
22
23          if(rect == null) {
24              rect = new MyRect(color);
25
26              // Add new rectangle to HashMap
27
28              rectsByColor.put(color, rect);
29
30          }
31          return rect;
32      }
33  }
```

8 Responses to “Flyweight Design Pattern Tutorial”

1.  *Anonymous* says:
March 7, 2013 at 11:29 am

   Very nice tutorials. Thanks you!

   Reply

   -  *Derek Banas* says:
   March 7, 2013 at 2:03 pm

     You’re very welcome 🙂

     Reply

2.  *boris* says:
March 29, 2013 at 1:09 am

   thank you so much for taking the time doing this

   Reply

   -  *Derek Banas* says:
   March 30, 2013 at 9:58 am

     You’re very welcome 🙂 Thank you for taking the time to show your appreciation!

     Reply

3.  *Emily Chen* says:
May 6, 2014 at 7:00 am

Great job! You are my on-line teacher now.

Reply

-  *Derek Banas* says:
  May 7, 2014 at 12:07 pm

  Thank you 🙂

  Reply

4.  *Ani* says:
   January 21, 2015 at 6:33 am

   You're awesome! Thanks to you I may have a chance to pass my final exam!

   Reply

   -  *Derek Banas* says:
     January 24, 2015 at 5:59 am

     Thank you 🙂 I wish you the best on your exam.

     Reply

Leave a Reply

Your email address will not be published.

Comment

Name

Email

Website

Submit Comment

Search

Search

Help Me Make Free Education

Donate Crypto

Social Networks

- Facebook
- YouTube
- Twitter
- LinkedIn

Buy me a Cup of Coffee
"Donations help me to keep the site running. One dollar is greatly appreciated." - (Pay Pal Secured)

My Facebook Page

Archives

- June 2018
- May 2018
- April 2018
- March 2018
- February 2018
- January 2018
- December 2017
- November 2017
- October 2017
- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017
- March 2017
- February 2017
- January 2017
- December 2016
- November 2016
- October 2016
- September 2016
- August 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- February 2016
- January 2016
- December 2015
- November 2015
- October 2015
- September 2015
- August 2015
- July 2015
- June 2015
- May 2015
- April 2015
- March 2015
- February 2015
- January 2015
- December 2014
- November 2014
- October 2014
- September 2014
- August 2014
- July 2014
- June 2014
- May 2014
- April 2014
- March 2014
- February 2014
- January 2014

- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- May 2013
- April 2013
- March 2013
- February 2013
- January 2013
- December 2012
- November 2012
- October 2012
- September 2012
- August 2012
- July 2012
- June 2012
- May 2012
- April 2012
- March 2012
- February 2012
- January 2012
- December 2011
- November 2011
- October 2011
- September 2011
- August 2011
- July 2011
- June 2011
- May 2011
- April 2011
- March 2011
- February 2011
- January 2011
- December 2010
- November 2010
- October 2010
- September 2010
- August 2010
- July 2010
- June 2010
- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009