Flyweight

```
public class StringTest {
	public static void main(String[] args) {
		String fly = "fly", weight = "weight";
		String fly2 = "fly", weight2 = "weight";
		System.out.println(fly == fly2);       // fly and fly2 refer to the same String instance
		System.out.println(weight == weight2); // weight and weight2 also refer to
		                                       // the same String instance
		String distinctString = fly + weight;
		System.out.println(distinctString == "flyweight"); // flyweight and "flyweight" do not
		                                    // refer to the same instance
		String flyweight = (fly + weight).intern();
		System.out.println(flyweight == "flyweight"); // The intern() method returns a flyweight
	}
}
```

Esimerkki1. Flyweight String-tyyppiä.

Luodaan neljä String-muuttujaa fly, fly2, weight, weight2.

fly == fly2 vertailu palauttaa true koska ovat viittaus samaan instanssimuuttujaan.

weight == weight2 vertailu palauttaa true koska ovat viittaus samaan instanssimuuttujaan.

Kun luodaan ohjelman aikana uusi String distinctString yhdistäen fly ja weight josta syntyy String flyweight

se ei ole sama kun verrataan distinctString = ”flyweight” nämä eivät viittaa samaan instanssimuuttujaan, koska = String + String luo aina uuden String instanssimuuttujan joka koostuu kahdesta Stringistä.

metodilla .intern() voidaan yhdistää Stringit viittaamaan samaan muuttujaan.

näin ollen uusi String == ”flyweight” on true.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
public class Test extends JFrame {
   public static void main(String[] args) {
      Test test = new Test();
      //test.setBounds(20,20,250,150);
      test.setBounds(50,150,550,550);
      test.show();
   }
   public Test() {
      super("Border flyweights");
      JPanel panel   = new JPanel(), panel2 = new JPanel();
      Border border  = BorderFactory.createRaisedBevelBorder();
      Border border2 = BorderFactory.createRaisedBevelBorder();
      panel.setBorder(border);
      panel.setPreferredSize(new Dimension(100,100));
      panel2.setBorder(border2);
      panel2.setPreferredSize(new Dimension(200,200));
      Container contentPane = getContentPane();
      contentPane.setLayout(new FlowLayout());
      contentPane.add(panel);
      contentPane.add(panel2);
      setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
      addWindowListener(new WindowAdapter() {
         public void windowClosed(WindowEvent e) {
            System.exit(0);
         }
      });
      if(border == border2)
         System.out.println("bevel borders are shared");
      else
         System.out.println("bevel borders are NOT shared");
   }
}
```

Esimerkki2. BorderFactory luo ruudun, border ja border2, jotka ovat samaa objektia.

Vaikka me muokkaamme setPreferredSize joka on ensin 100,100 ja sitten 200,200

on silti border == border2 totta koska me muokkaamme samaa Border objektia ja viittaus siihen on aina sama.