```
            Rectangle r = new Rectangle(x,row,W,H);
            if (r.contains(e.getX(), e.getY())) {
                selectedName=(String)names.elementAt(i);
                repaint();
            }
            x = x + HSpace;       //change to the next position
            j++;
            if (j >= HCount) {    //reset for the next row
                j = 0;
                row += VSpace;
                x = Left;
            }
        }
    }
```

Flyweight Uses in Java

Flyweights are not often used at the application level in Java. They are more of a system resource management technique that is used at a lower level than Java. However, it is useful to recognize that this technique exists so that you can use it if you need it.

We have already seen the Flyweight pattern in the cell renderer code used for tables and list boxes. Usually the cell renderer is just a JLabel; however, there might be two or three types of labels or renderers for different colors or fonts. But there are far fewer renderers than there are cells in the table or list. Some objects within the Java language could be implemented under the covers as Flyweights. For example, if two instances of a String constant with identical characters exist, they could refer to the same storage location. Similarly, two Integer or Float objects that contain the same value might be implemented as Flyweights, although they probably are not. To prove the absence of Flyweights here, we run the following code:

```
Integer five = new Integer(5);
Integer myfive = new Integer(5);
System.out.println(five == myfive);

String fred=new String("fred");
String fred1 = new String("fred");
System.out.println(fred == fred1);
```

Both cases print out "false." However, note that you can easily determine that you are dealing with two identical instances of a Flyweight by using the == operator. This operator compares actual object references (memory addresses) rather than the "equals" operator, which will probably be slower, if it is implemented at all.