```
 1: /*
 2:
 3: Jared Dyreson
 4: CWID: 889546529
 5: Checker.java ->  Move that kitty all over the board
 6: csrc_compile: TRUE
 7: */
 8:
 9: import java.awt.*;
10: import javax.swing.*;
11: import java.awt.Color;
12: import javax.swing.JFrame;
13: import java.awt.event.*;
14: import java.text.MessageFormat;
15:
16: // NOTE: PDF rendered does not contain screenshots of code
17: // these are PDFs created automatically via enscript
18: // script is included
19:
20: public class Checker extends JFrame implements ActionListener{
21:
22:         private final int ROWS = 8, COLS = 8, FRAME_HEIGHT = 500,  FRAME_WIDTH = 500;
23:         private JPanel pane = new JPanel(new GridLayout(ROWS, COLS,2, 2));
24:         private JPanel[][] tpanel = new JPanel[8][8];
25:
26:         // colors to make the checker board
27:         private Color c1 = Color.WHITE;
28:         private Color c2 = Color.CYAN;
29:         private Color tmp;
30:
31:         // movement buttons
32:         private JButton up = new JButton("UP");
33:         private JButton down = new JButton("DOWN");
34:         private JButton left = new JButton("LEFT");
35:         private JButton right = new JButton("RIGHT");
36:
37:         // cat object
38:         private Kitty cat = new Kitty();
39:         // how we represent the cat in the checker board
40:         private JLabel face_label = new JLabel(cat.get_face());
41:
42:         public Checker(){
43:                 super("Run Kitty Run");
44:                 this.setLayout(new BorderLayout());
45:                 this.setSize(FRAME_HEIGHT, FRAME_WIDTH);
46:                 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
47:
48:                 up.addActionListener(this);
```

```
49:                 down.addActionListener(this);
50:                 left.addActionListener(this);
51:                 right.addActionListener(this);
52:
53:                 this.add(up, BorderLayout.NORTH);
54:                 this.add(down, BorderLayout.SOUTH);
55:                 this.add(left, BorderLayout.WEST);
56:                 this.add(right, BorderLayout.EAST);
57:                 this.add(pane, BorderLayout.CENTER);
58:
59:                 // making the checker board using a nest for loop
60:
61:                 for(int x = 0; x < COLS; ++x){
62:                         // x -> COLUMNS
63:                         for(int y = 0; y < ROWS; ++y){
64:                                 // y -> ROWS
65:                                 tpanel[x][y] = new JPanel();
66:                                 pane.add(tpanel[x][y]);
67:                                 // this makes the alternating cyan, white appearence
68:                                 if(x % ROWS == 0){
69:                                         // swap the colors
70:                                         tmp = c1;
71:                                         c1 = c2;
72:                                         c2 = tmp;
73:                                 }
74:                                 else{
75:                                         tmp = c2;
76:                                         c2 = c1;
77:                                         c1 = tmp;
78:                                 }
79:                                 // if even index, set it to the first color, else the second
80:                                 if(x % 2 == 0){ tpanel[x][y].setBackground(c1); }
81:                                 else{ tpanel[x][y].setBackground(c2); }
82:                         }
83:                 }
84:                 // place the cat in the middle of the checker board
85:                 this.set_kitty(4, 4);
86:                 //tpanel[0][0].add(face_label);
87:         }
88:
89:         public void set_kitty(int x, int y) throws IndexOutOfBoundsException{
90:                 // since our arrays start at 0, we appear to be one off but in reality we are not
91:
92:                 // original coordinates
93:                 int x_naught = cat.get_x();
94:                 int y_naught = cat.get_y();
95:
96:
```

```
97:                 // place the cat
98:                 tpanel[y][x].add(face_label);
99:
100:                cat.set_position(x, y);
101:
102:                // getting the string representation of the coordinates for the cat
103:                String x_s = String.valueOf(cat.get_x());
104:                String y_s = String.valueOf(cat.get_y());
105:                String coordinate_message = MessageFormat.format("{0}, {1}", x_s, y_s);
106:
107:                // get the original place where the cat was and clean up
108:                tpanel[x_naught][y_naught].removeAll();
109:                // these methods reload the JPanel object
110:                pane.revalidate();
111:                pane.repaint();
112:
113:                // update the positions of the cat
114:                // show where the cat is
115:                System.out.println(coordinate_message);
116:
117:        }
118:        @Override
119:        public void actionPerformed(ActionEvent event){
120:
121:                Object source = event.getSource();
122:                int x = this.cat.get_x();
123:                int y = this.cat.get_y();
124:
125:                // -/+ are switched because of our frame of reference
126:
127:                if(source == up){
128:                        try{
129:                                this.set_kitty(x, y-1);
130:                                //this.set_kitty(this.cat.get_x()-1, this.cat.get_y()+1);
131:                                // ^ makes the cat go diagonal on the same color
132:                                // like a bishop on a chess board
133:                                //this.set_kitty(this.cat.get_y(), this.cat.get_x()-1);
134:                                // apparently this code ^ makes the cat move diagonally
135:                        }
136:                        // these try catch blocks are a cheeky work around for hitting the edge of the board
137:                        catch(Exception error){}
138:                }
139:                else if(source == down){
140:                        try{
141:                                this.set_kitty(x, y+1);
142:                        }
143:                        catch(Exception error){}
144:                }
```

```
145:                 else if(source == left){
146:                         try{
147:                                 this.set_kitty(x-1, y);
148:                         }
149:                         catch(Exception error){}
150:                 }
151:                 else if(source == right){
152:                         try{
153:                                 this.set_kitty(x+1, y);
154:                         }
155:                         catch(Exception error){}
156:                 }
157:         }
158:         public static void main(String[] args){
159:                 // Auto generated with caffine and lightdm.service
160:                 Checker c = new Checker();
161:                 c.setVisible(true);
162:         }
163: }
```