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

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

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

```
145:                    }
146:                    else if(source == left){
147:                            try{
148:                                    this.set_kitty(x-1, y);
149:                            }
150:                            catch(Exception error){}
151:                    }
152:                    else if(source == right){
153:                            try{
154:                                    this.set_kitty(x+1, y);
155:                            }
156:                            catch(Exception error){}
157:                    }
158:            }
159:            public static void main(String[] args){
160:                    // Auto generated with caffine and lightdm.service
161:                    Checker c = new Checker();
162:                    c.setVisible(true);
163:            }
164: }
```