

src\ConwayPanel.java

```
1  import java.awt.Color;
2  import java.awt.Dimension;
3  import java.awt.event.KeyEvent;
4  import java.awt.event.KeyListener;
5  import java.awt.event.MouseEvent;
6  import java.awt.event.MouseListener;
7  import java.awt.event.MouseMotionListener;
8  import java.awt.Graphics;
9  import javax.swing.JPanel;
10
11  public class ConwayPanel extends JPanel implements KeyListener, MouseListener,
    MouseMotionListener {
12      int fps = 1;
13      final int START_WIDTH = 500;
14      final int START_HEIGHT = 500;
15      int gridSize = 10;
16      int[][] cells;
17      boolean go = true;
18      boolean pause = true;
19
20      public ConwayPanel() {
21          setPreferredSize(new Dimension(START_WIDTH, START_HEIGHT));
22          setBackground(Color.BLACK);
23          addKeyListener(this);
24          addMouseListener(this);
25          addMouseMotionListener(this);
26          final int NUM_CELLS = 50;
27          cells = new int[NUM_CELLS][NUM_CELLS];
28          cells[24][25] = 1;
29          cells[25][25] = 1;
30          cells[26][25] = 1;
31          cells[24][26] = 1;
32          cells[25][24] = 1;
33
34      }
35
36      public void paintComponent(Graphics g) {
37          super.paintComponent(g);
38          int width = this.getWidth();
39          int height = this.getHeight();
40
41          g.setColor(Color.LIGHT_GRAY);
42
43          for (int y = 0; y <= height; y += 10) {
44              g.drawLine(0, y, width, y);
45          }
46          for (int x = 0; x < width; x += 10) {
47              g.drawLine(x, 0, x, height);
48          }
49          drawCells(g);
50      }
51  }
52
```

```
53     public void run() {
54
55         while (go) {
56             if(!pause){
57                 repaint();
58                 cells = updateCells(cells);
59             }
60             delay(1000 / fps);
61         }
62         System.exit(0);
63     }
64
65     public void delay(int n) {
66         try {
67             Thread.sleep(n);
68         } catch (InterruptedException ex) {
69             Thread.currentThread().interrupt();
70         }
71     }
72
73     public void drawCells(Graphics g) {
74
75         for (int r = 0; r < cells.length; r++) {
76             for (int c = 0; c < cells[0].length; c++) {
77                 if (r % 2 == 0) {
78                     g.setColor(Color.WHITE);
79                 }
80                 if (c % 2 == 0) {
81                     g.setColor(Color.RED);
82                 }
83                 if (c % 2 != 0 && r % 2 != 0) {
84                     g.setColor(Color.BLUE);
85                 }
86                 if (cells[r][c] == 1) {
87                     g.fillOval(c * gridSize, r * gridSize, gridSize, gridSize);
88                 }
89             }
90         }
91     }
92     public int[][] updateCells(int[][] cells) {
93
94         int[][] updated = new int[cells.length][cells.length];
95         int[][] cellCheck = new int[][] { { 1, 0 }, { -1, 0 }, { 0, -1 },
96             { 0, 1 }, { -1, -1 }, { -1, 1 }, { 1, -1 }, { 1, 1 } };
97
98         for (int r = 1; r < cells.length - 1; r++) {
99             for (int c = 1; c < cells[0].length - 1; c++) {
100                 int neighbor = 0;
101                 for (int[] checkCol : cellCheck) {
102                     int x = checkCol[0] + r;
103                     int y = checkCol[1] + c;
104                     if (cells[x][y] == 1) {
105                         neighbor++;
106                     }
107                 }
108                 if (cells[r][c] == 1 && neighbor == 3) {
```

```
109         updated[r][c] = 1;
110     } else if (cells[r][c] == 1 && (neighbor == 2) || neighbor == 3) {
111         updated[r][c] = 1;
112     }
113 }
114 }
115     return updated;
116 }
117
118 @Override
119 public void mouseClicked(MouseEvent e) {
120     pause = true;
121     int x = (int)e.getX()/ gridSize;
122     int y = (int)e.getY()/ gridSize;
123
124     cells[y][x]++;
125     repaint();
126 }
127
128
129 @Override
130 public void mousePressed(MouseEvent e) {
131
132 }
133
134 @Override
135 public void mouseReleased(MouseEvent e) {
136
137 }
138
139 @Override
140 public void mouseEntered(MouseEvent e) {
141
142
143 }
144
145 @Override
146 public void mouseExited(MouseEvent e) {
147
148
149 }
150
151 @Override
152 public void keyTyped(KeyEvent e) {
153     if (e.getKeyChar() == 'q') {
154         go = false;
155     }
156     if (e.getKeyChar() == '+') {
157         fps++;
158     }
159     if (e.getKeyChar() == '-' && fps > 0) {
160         fps--;
161     }
162 }
163
164 @Override
```

```
165     public void keyPressed(KeyEvent e) {
166         if(e.getKeyCode() == KeyEvent.VK_SPACE){
167             pause = !pause;
168         }
169     }
170
171     @Override
172     public void keyReleased(KeyEvent e) {
173         // unused
174     }
175
176
177     @Override
178     public void mouseDragged(MouseEvent e) {
179     }
180
181
182     @Override
183     public void mouseMoved(MouseEvent e) {
184     }
185 }
186
187
```