src\ConwayPanel.java

```java
import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.Graphics;
import javax.swing.JPanel;

public class ConwayPanel extends JPanel implements KeyListener, MouseListener,
MouseMotionListener {
    int fps = 1;
    final int START_WIDTH = 500;
    final int START_HEIGHT = 500;
    int gridSize = 10;
    int[][] cells;
    boolean go = true;
    boolean pause = true;

    public ConwayPanel() {
        setPreferredSize(new Dimension(START_WIDTH, START_HEIGHT));
        setBackground(Color.BLACK);
        addKeyListener(this);
        addMouseListener(this);
        addMouseMotionListener(this);
        final int NUM_CELLS = 50;
        cells = new int[NUM_CELLS][NUM_CELLS];
        cells[24][25] = 1;
        cells[25][25] = 1;
        cells[26][25] = 1;
        cells[24][26] = 1;
        cells[25][24] = 1;

    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        int width = this.getWidth();
        int height = this.getHeight();

        g.setColor(Color.LIGHT_GRAY);

        for (int y = 0; y <= height; y += gridSize) {
            g.drawLine(0, y, width, y);

        }
        for (int x = 0; x < width; x += gridSize) {
            g.drawLine(x, 0, x, height);
        }
        drawCells(g);
    }

```

```
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          int evendivisor = 2;
75          int odddivisor = 3;
76
77          for (int r = 0; r < cells.length; r++) {
78              for (int c = 0; c < cells[0].length; c++) {
79                  if (r % evendivisor == 0) {
80                      g.setColor(Color.WHITE);
81                  }
82                  if (c % evendivisor == 0) {
83                      g.setColor(Color.RED);
84                  }
85                  if (c % evendivisor != 0 && r % evendivisor != 0) {
86                      g.setColor(Color.BLUE);
87                  }
88                  if (cells[r][c] == 1) {
89                      g.fillOval(c * gridSize, r * gridSize, gridSize, gridSize);
90                  }
91              }
92          }
93      }
94      public int[][] updateCells(int[][] cells) {
95          int stayalive = 2;
96          int reproduce = 3;
97
98          int[][] updated = new int[cells.length][cells.length];
99          int[][] cellCheck = new int[][] { { 1, 0 }, { -1, 0 }, { 0, -1 },
100                 { 0, 1 }, { -1, -1 }, { -1, 1 }, { 1, -1 }, { 1, 1 } };
101
102         for (int r = 1; r < cells.length - 1; r++) {
103             for (int c = 1; c < cells[0].length - 1; c++) {
104                 int neighbor = 0;
105                 for (int[] checkCol : cellCheck) {
106                     int x = checkCol[0] + r;
107                     int y = checkCol[1] + c;
108                     if (cells[x][y] == 1) {
```

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

```
164             fps--;
165         }
166     }
167 
168     @Override
169     public void keyPressed(KeyEvent e) {
170         if(e.getKeyCode() == KeyEvent.VK_SPACE){
171             pause = !pause;
172         }
173     }
174 
175     @Override
176     public void keyReleased(KeyEvent e) {
177         // unused
178 
179     }
180 
181     @Override
182     public void mouseDragged(MouseEvent e) {
183 
184     }
185 
186     @Override
187     public void mouseMoved(MouseEvent e) {
188 
189     }
190 }
191 
```