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 += 10) {
            g.drawLine(0, y, width, y);

        }
        for (int x = 0; x < width; x += 10) {
            g.drawLine(x, 0, x, height);
        }
        drawCells(g);
    }

    public void run() {

        while (go) {
            if(!pause){
                repaint();
                cells = updateCells(cells);
            }
            delay(1000 / fps);
        }
        System.exit(0);
    }

    public void delay(int n) {
        try {
            Thread.sleep(n);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
    }

    public void drawCells(Graphics g) {

        for (int r = 0; r < cells.length; r++) {
            for (int c = 0; c < cells[0].length; c++) {
                if (r % 2 == 0) {
                    g.setColor(Color.WHITE);
                }
                if (c % 2 == 0) {
                    g.setColor(Color.RED);
                }
                if (c % 2 != 0 && r % 2 != 0) {
                    g.setColor(Color.BLUE);
                }
                if (cells[r][c] == 1) {
                    g.fillOval(c * gridSize, r * gridSize, gridSize, gridSize);
                }
            }
        }
    }
    public int[][] updateCells(int[][] cells) {

        int[][] updated = new int[cells.length][cells.length];
        int[][] cellCheck = new int[][] { { 1, 0 }, { -1, 0 }, { 0, -1 },
                { 0, 1 }, { -1, -1 }, { -1, 1 }, { 1, -1 }, { 1, 1 } };

        for (int r = 1; r < cells.length - 1; r++) {
            for (int c = 1; c < cells[0].length - 1; c++) {
                int neighbor = 0;
                for (int[] checkCol : cellCheck) {
                    int x = checkCol[0] + r;
                    int y = checkCol[1] + c;
                    if (cells[x][y] == 1) {
                        neighbor++;
                    }
                }
                if (cells[r][c] == 1 && neighbor == 3) {
                    updated[r][c] = 1;
                } else if (cells[r][c] == 1 && (neighbor == 2) || neighbor == 3) {
                    updated[r][c] = 1;
            }
        }
    }
        return updated;
    }

    @Override
    public void mouseClicked(MouseEvent e) {
        pause = true;
        int x = (int)e.getX()/ gridSize;
        int y = (int)e.getY()/ gridSize;

        cells[y][x]++;
        repaint();

    }

    @Override
    public void mousePressed(MouseEvent e) {

    }

    @Override
    public void mouseReleased(MouseEvent e) {

    }

    @Override
    public void mouseEntered(MouseEvent e) {


    }

    @Override
    public void mouseExited(MouseEvent e) {


    }

    @Override
    public void keyTyped(KeyEvent e) {
        if (e.getKeyChar() == 'q') {
            go = false;
        }
        if (e.getKeyChar() == '+') {
            fps++;
        }
        if (e.getKeyChar() == '-' && fps > 0) {
            fps--;
        }
    }

    @Override
```

```java
    public void keyPressed(KeyEvent e) {
        if(e.getKeyCode() == KeyEvent.VK_SPACE){
            pause = !pause;
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {
        // unused

    }

    @Override
    public void mouseDragged(MouseEvent e) {

    }

    @Override
    public void mouseMoved(MouseEvent e) {

    }
}

```