```
/**
 *Elliot Duncan
 *Horton 7th
 *5/11/24
 *
 *@(#)WorldPanel.java
 *
 * Creates a JPanel to display the World and its animation.
 */

import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.Timer;
class WorldPanel extends JPanel {
  private int windowSize;
  private boolean boundingBox;
  private int PIXEL_SIZE;
  private final int ORIGINAL_PIXEL_SIZE;
  private int offsetX = 25;
  private int offsetY = 25;
  private int delay = 16;
  private boolean drawEdge = false;
  private boolean grid = false;

  private final ActionListener al = new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      if (timer.isRunning()) {
        World.nextGeneration();
        repaint();
        Toolkit.getDefaultToolkit().sync();
      }
    }
  };
  private final Timer timer = new Timer(delay, al);
  private final KeyListener kl = new KeyListener() {
    @Override
    public void keyReleased(KeyEvent e) {}
    @Override
    public void keyPressed(KeyEvent e) {
      System.out.println("code=" + e.getKeyCode() + ", char=" + e.getKeyChar());
      if (e.getKeyChar() == '?')
```

```
  JOptionPane.showMessageDialog(
      new JPanel(),
      "Esc - quit\n"
          + "r - restart from loaded seed\n"
          + "e - shows the edge of the world\n"
          + "b - draws the bounding box of the current cells\n"
          + "-, =, Arrow keys, move around and zoom\n"
          + "i - flip some the different seeds - press S to sort while " +
            "in the menu\n"
          + " b - toggle bounding box\n"
          + "SPC - pause\n"
          + "d - change delay\n"
          + "l - load new seed\n"
          + "While paused...\n"
          + "    . - Step generation\n"
          + "    s - save to file\n");

if (e.getKeyCode() == KeyEvent.VK_ESCAPE)
  System.exit(0);
if (e.getKeyCode() == KeyEvent.VK_SPACE) // Pause Execution
  toggleTimer();
if (e.getKeyCode() == KeyEvent.VK_B) { // toggle bounding box
  boundingBox = !boundingBox;
}
if (e.getKeyCode() == KeyEvent.VK_D) { // Change Delay
  String d = "" + JOptionPane.showInputDialog(
                      "Input new delay (milliseconds)", "" + delay);
  try {
    delay = (d.equals("")) ? delay : Integer.parseInt(d);
    timer.stop();
    timer.setDelay(delay);
    timer.start();
  } catch (NumberFormatException a) {
    System.err.println(a);
  }
}
if (e.getKeyCode() == KeyEvent.VK_EQUALS) {
  PIXEL_SIZE++;
}
if (e.getKeyCode() == KeyEvent.VK_MINUS) {
  PIXEL_SIZE--;
}
if (e.getKeyCode() == KeyEvent.VK_LEFT) {
  offsetX += 50;
}
if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
  offsetX -= 50;
}
if (e.getKeyCode() == KeyEvent.VK_UP) {
  offsetY += 50;
```

```
    }
    if (e.getKeyCode() == KeyEvent.VK_DOWN) {
      offsetY -= 50;
    }
    if (e.getKeyCode() == KeyEvent.VK_R) {
      World.setWorld(new Seed(World.getSeedName()));
    }
    if (e.getKeyCode() == KeyEvent.VK_L) { // load a new Seed
      Main.askForFile();
    }
    if (e.getKeyCode() == KeyEvent.VK_I) {
      timer.stop();
      Main.displayAllSeeds(Main.getPatterns(100));
    }
    if (e.getKeyCode() == KeyEvent.VK_E) {
      drawEdge = !drawEdge;
    }
    if (e.getKeyCode() == KeyEvent.VK_G) {
      grid = !grid;
    }
    // Save Current World to file (MUST BE PAUSED)
    if (e.getKeyCode() == KeyEvent.VK_S) {
      if (!timer.isRunning()) {
        Main.saveWorld();
      }
    }
    if (e.getKeyCode() == KeyEvent.VK_PERIOD) {
      World.nextGeneration();
      Toolkit.getDefaultToolkit().sync();
    }
    if (e.getKeyCode() == KeyEvent.VK_0) {
      offsetY = offsetX = 25;
      PIXEL_SIZE = ORIGINAL_PIXEL_SIZE;
    }
    repaint();
  }

  @Override
  public void keyTyped(KeyEvent e) {}
};

/**
 * Generates a new WorldPanel, with a given size in pixels
 * @param size the size of the window in pixels
 */
public WorldPanel(int size) {
  setBackground(new Color(20, 20, 20));
  PIXEL_SIZE = Math.max(Math.min(100, size / World.getWORLD_SIZE()), 2);
  ORIGINAL_PIXEL_SIZE = PIXEL_SIZE;
  windowSize = (World.getWORLD_SIZE() * PIXEL_SIZE) + 100;
```

```
}

/**
 * Returns the KeyListener for the WorldPanel object, which defines the
 * keybinds of the window.
 * @return the key listener
 */
public KeyListener getKeyListener() { return kl; }

/**
 *Returns the dimension representing the preferred size of the window.
 *@return the preferred size
 */
public Dimension getPreferredSize() {
  return new Dimension(windowSize, windowSize);
}

/**
 *paints the seed to the window
 *@param g the current graphics context
 */
@Override
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  this.setDoubleBuffered(false);
  g.setColor(new Color(200, 200, 200));
  drawSeed(g);
  if (drawEdge)
    g.drawRect(offsetX, offsetY, PIXEL_SIZE * World.getWORLD_SIZE(),
               PIXEL_SIZE * World.getWORLD_SIZE());
  Toolkit.getDefaultToolkit().sync();
}

private void toggleTimer() {
  if (timer.isRunning()) {
    timer.stop();
  } else {
    timer.start();
  }
}

private void drawSeed(Graphics g) {
  if (boundingBox)
    g.drawRect(offsetX + PIXEL_SIZE * (int)World.getBoundingBox().getX(),
               offsetY + PIXEL_SIZE * (int)World.getBoundingBox().getY(),
               PIXEL_SIZE * ((int)World.getBoundingBox().getWidth()),
               PIXEL_SIZE * ((int)World.getBoundingBox().getHeight()));
  g.drawString(
      String.format("Press ? for help    FrameDelay: %d  Generation: %d",
                    timer.getDelay(), World.getGeneration()),
```

```
        10, 10);

    int gw = (grid) ? 1 : 0; // draw grid?
    for (int i = 0; i < World.getWORLD_SIZE(); i++)
      for (int j = 0; j < World.getWORLD_SIZE(); j++)
        if (World.getWorld()[i][j] > 0)
          g.fillRect(offsetX + PIXEL_SIZE * j, offsetY + PIXEL_SIZE * i,
                     PIXEL_SIZE - gw, PIXEL_SIZE - gw);
  }
}
```