CT255 -- Week#7 Sample Solution -- Conway's Game of Life

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.image.*;

public class ConwaysLife extends JFrame implements Runnable, MouseListener
{

   // member data
   private BufferStrategy strategy;
   private Graphics offscreenBuffer;
   private boolean gameState[][][] = new boolean[40][40][2];
   private int gameStateFrontBuffer = 0;
   private boolean isGameRunning = false;
   private boolean initialised = false;

   // constructor
   public ConwaysLife () {
        //Display the window, centred on the screen
        Dimension screensize =
           java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        int x = screensize.width/2 - 400;
        int y = screensize.height/2 - 400;
        setBounds(x, y, 800, 800);
        setVisible(true);
         this.setTitle("Conway's game of life");

        // initialise double-buffering
        createBufferStrategy(2);
        strategy = getBufferStrategy();
        offscreenBuffer = strategy.getDrawGraphics();

        // register the Jframe itself to receive mouse events
        addMouseListener(this);

        // initialise the game state
        for (x=0;x<40;x++) {
           for (y=0;y<40;y++) {
              gameState[x][y][0]=gameState[x][y][1]=false;
           }
        }

        // create and start our animation thread
        Thread t = new Thread(this);
        t.start();

        initialised = true;
   }

   // thread's entry point
   public void run() {
```

```
      while ( 1==1 ) {
         // 1: sleep for 1/10 sec
         try {
            Thread.sleep(100);
         } catch (InterruptedException e) { }

         // 2: animate game objects
         if (isGameRunning)
            doOneEpochOfGame();

         // 3: force an application repaint
         this.repaint();

      }
   }

   private void doOneEpochOfGame() {
      // apply game rules to game state 'front buffer', copying results
into 'back buffer'
      int front = gameStateFrontBuffer;
      int back = (front+1)%2;
      for (int x=0;x<40;x++) {
         for (int y=0;y<40;y++) {
            // count the neighbours of cell x,y
            int liveneighbours=0;
            for (int xx=-1;xx<=1;xx++) {
               for (int yy=-1;yy<=1;yy++) {
                  if (xx!=0 || yy!=0) {
                     int xxx=x+xx;
                     if (xxx<0)
                        xxx=39;
                     else if (xxx>39)
                        xxx=0;
                     int yyy=y+yy;
                     if (yyy<0)
                        yyy=39;
                     else if (yyy>39)
                        yyy=0;
                     if (gameState[xxx][yyy][front])
                        liveneighbours++;
                  }
               }
            }

            // apply rules for cell x,y
            if (gameState[x][y][front]) {
               // cell x,y was alive
            // #1. Any live cell with fewer than two live neighbours dies
               if (liveneighbours<2)
                  gameState[x][y][back] = false;
            // #2. Any live cell with two or three live neighbours lives
               else if (liveneighbours<4)
                  gameState[x][y][back] = true;
            // #3. Any live cell with more than three live neighbours dies
```

```
            else
               gameState[x][y][back] = false;
         }
         else {
            // cell x,y was dead
            // #4. Dead cells with three live neighbours become live
            if (liveneighbours==3)
               gameState[x][y][back] = true;
            else
               gameState[x][y][back] = false;
         }
      }
      }

   // now flip the game state buffers
   gameStateFrontBuffer = back;
}

private void randomiseGameState() {
     for (int x=0;x<40;x++) {
        for (int y=0;y<40;y++) {
           gameState[x][y][gameStateFrontBuffer]=(Math.random()<0.25);
        }
     }
}

// mouse events which must be implemented for MouseListener
 public void mousePressed(MouseEvent e) {
     if (!isGameRunning) {
        // was the click on the 'start button'?
        int x = e.getX();
        int y = e.getY();
        if (x>=15 && x<=85 && y>=40 && y<=70) {
           isGameRunning=true;
           return;
        }
        // or on the 'random' button?
        if (x>=115 && x<=215 && y>=40 && y<=70) {
           randomiseGameState();
           return;
        }
     }

      // determine which cell of the gameState array was clicked on
      int x = e.getX()/20;
      int y = e.getY()/20;
      // toggle the state of the cell
      gameState[x][y][gameStateFrontBuffer] =
                                 !gameState[x][y][gameStateFrontBuffer];
      // throw an extra repaint, to get immediate visual feedback
      this.repaint();
  }

  public void mouseReleased(MouseEvent e) { }
```

```
    public void mouseEntered(MouseEvent e) { }

    public void mouseExited(MouseEvent e) { }

    public void mouseClicked(MouseEvent e) { }

  // application's paint method
  public void paint(Graphics g) {
    if (!initialised)
      return;

    g = offscreenBuffer; // draw to offscreen graphics buffer

    // clear the canvas with a big black rectangle
    g.setColor(Color.BLACK);
    g.fillRect(0, 0, 800, 800);

    // redraw all game objects
    g.setColor(Color.WHITE);
      for (int x=0;x<40;x++) {
       for (int y=0;y<40;y++) {
         if (gameState[x][y][gameStateFrontBuffer]) {
           g.fillRect(x*20, y*20, 20, 20);
         }
       }
      }

      if (!isGameRunning) {
       // game is not running..
       // draw a 'start button' as a rectangle with text on top
       // also draw a 'randomise' button
       g.setColor(Color.GREEN);
       g.fillRect(15, 40, 70, 30);
       g.fillRect(115, 40, 100, 30);
       g.setFont(new Font("Times", Font.PLAIN, 24));
       g.setColor(Color.BLACK);
       g.drawString("Start", 22, 62);
       g.drawString("Random", 122, 62);
      }

    // flip the graphics buffers
    strategy.show();
  }

  // application entry point
  public static void main(String[] args) {
    ConwaysLife w = new ConwaysLife();
  }

}
```