Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

Cover Page

I, _*Xiaoxi Zheng*___ affirm that the work submitted is my own and that the Honor Code was neither bent nor broken.

The easiest part of this HW is the setup of this project, since the structure of code was quite straight forward implemented on top of variety of provided set of codes. The more difficult parts of the HW is buried within implementation of the Game of life algorithm and how the components of JFrame, and JPanel interact with each other. This HW is relatively straight forward comparing to the past couple, and the past HWs are extremely helpful for animation and etc. I spend some time debugging for minor Array Out of Bound at certain places(when testing

I believe the objective of this assignment was for us to reiterate the understanding of the Timer object and implement Animation frame by frame using JPanel and JFrames. How these classes and objects call on each other, and making sure we understand the structural relationship between them. It's also pretty neat to visually the classic algorithm for Conway's game of life.

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

Code

```
import javax.swing.SwingUtilities;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.*;
import javax.swing.JOptionPane;
import javax.swing.JFileChooser;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JLabel;
import javax.swing.Timer;
import javax.imageio.ImageIO;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.image.BufferedImage;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseMotionAdapter;
import java.awt.event.KeyEvent;
import javax.swing.KeyStroke;
import java.io.File;
import java.io.IOException;
import java.util.Random;


public class hw11{
	private static final int WIDTH = 800;
	private static final int HEIGHT = 800;

	public static void main( String[] args){
		SwingUtilities.invokeLater( new Runnable() {
			public void run()
			{
				createAndShowGUI();
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
                        }
                } );

        }
        public static void createAndShowGUI(){
                JFrame frame = new ImageFrame(WIDTH,HEIGHT);
                frame.setVisible( true );
        }
}
//#########################################################################
class ImageFrame extends JFrame {

        //static string that represents state of cell with color and int
        static private final Color BLACK = Color.BLACK;//indicates dead cell
        static private final Color RED = Color.RED;//died this generation
        static private final Color GREEN = Color.GREEN;//born this generation
        static private final Color BLUE = Color.BLUE;//live cell for more than 1 generation

        static private final int DEAD = 0;
        static private final int JUST_DIED = 1;
        static private final int JUST_ALIVE = 2;
        static private final int ALIVE = 3;

        static final Color[] colorBaseOnState = {BLACK, RED,GREEN,BLUE};


        //a 2D array to hold the states of the each individual cell in the grid
        private int [][] cell;

        private int MILLESECONDS_BETWEEN_FRAMES;

        private DisplayPanel panel;
        private JButton button;
        private JLabel label;

        //changes a cell being alive or dead [0, 1]
        private double aliveChance;
        private boolean randomWorld;
        private int width = 800;
        private int height = 800;
        private BufferedImage image;


        //=========================
        public ImageFrame(int width, int height){
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
this.setTitle("CAP 3027 2015 - HW11 -XiaoxiZheng");
this.setSize( width, height );
this.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE);

addMenu();////add a menu to the frame

image = simulatedImage(width,height);

panel = new DisplayPanel( image);

cell = new int[100][100];
//initialize every cell to dead
for(int i=0; i<100;i++){
	for(int j = 0; j<100;j++){
		cell[i][j] = DEAD;
	}
}

button = new JButton("Start");
button.addActionListener( new ActionListener(){
	public void actionPerformed( ActionEvent event ){
		if(!(panel.timerisRunning())) {
			button.setText("Pause");
			panel.startTimer();
			System.out.println("Pause");
		}

		else{
			button.setText("Start");
			panel.stopTimer();
			System.out.println("Start");
		}
	}
} );

JSlider slider = new JSlider(JSlider.VERTICAL, 100, 800, 500);
 MILLESECONDS_BETWEEN_FRAMES = slider.getValue();
//label = new JLabel("Click Start/Pause to pause and resume your animation");
this.getContentPane().add( panel, BorderLayout.CENTER );
this.getContentPane().add(button, BorderLayout.SOUTH);
this.getContentPane().add(slider,BorderLayout.EAST);
//this.getContentPane().add( label, BorderLayout.NORTH );
this.pack();
this.setVisible( true );    }
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
	private void addMenu(){
		JMenu fileMenu = new JMenu("File Menu");
		//load IFS description
		JMenuItem randPop = new JMenuItem("Randomly Populate World");
		randPop.addActionListener( new ActionListener(){
			public void actionPerformed( ActionEvent event){
				aliveChance = promptForAliveChances();
				randomWorld();
			}
		}  );
		fileMenu.add(randPop);

		JMenuItem emptyWorld = new JMenuItem("Empty World");
		emptyWorld.addActionListener( new ActionListener(){
			public void actionPerformed(ActionEvent event){
				aliveChance = 0;
				emptyWorld();
			}
		}  );
		fileMenu.add(emptyWorld);

		//Save image
		JMenuItem saveImage = new JMenuItem("Save Image");
		saveImage.addActionListener( new ActionListener(){
			public void actionPerformed( ActionEvent event){
				saveImage();
			}
		}  );
		fileMenu.add(saveImage);
		//Exit
		JMenuItem exitItem = new JMenuItem("Exit");
		exitItem.addActionListener( new ActionListener(){
			public void actionPerformed(ActionEvent event){
				System.exit( 0 );
			}
		}	);
		fileMenu.add( exitItem);

		//attach menu to a menu bar
		JMenuBar menuBar = new JMenuBar();
		menuBar.add( fileMenu);
		this.setJMenuBar( menuBar);
	}

	private void randomWorld(){
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
                randomWorld = true;
                configureCellState(aliveChance);
                //panel.repaint();
                //panel.setImage();
        }
        private void emptyWorld(){
                randomWorld = false;
                configureCellState(aliveChance);
                //panel.repaint();
                //panel.setImage();
        }
        private void configureCellState(double alive_){
                if (alive_ == 0.0) {
                        //looping thru every grid on the x, y plane
                        for (int i = 0; i < 100; i++)
                                for (int j = 0; j < 100; j++)
                                        panel.setState(i, j, DEAD);
                } else if (alive_ == 1.0) {
                        for (int i = 0; i < 100; i++)
                                for (int j = 0; j < 100; j++)
                                        panel.setState(i, j, ALIVE);
                } else {
                        Random rand = new Random();
                        for (int i = 0; i < 100; i++) {
                                for (int j = 0; j < 100; j++) {
                                        if (rand.nextDouble() < alive_)
                                                panel.setState(i, j, ALIVE);
                                        else
                                                panel.setState(i, j, DEAD);
                                }
                        }
                }
                repaint();
        }
        private void saveImage(){
                try
                        {
                                File outputfile = new File("life.png");
                                javax.imageio.ImageIO.write(image, "png", outputfile );
                        }
                        catch ( IOException e )
                        {
                          JOptionPane.showMessageDialog( ImageFrame.this,
                                                        "Error saving file",
                                                            "oops!",
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
                                                                    JOptionPane.ERROR_MESSAGE );
                        }
        }
        private void stop(){
                panel.stopTimer();
                button.setText("Start");
        }
        private double promptForAliveChances(){
                String input1 = JOptionPane.showInputDialog("Please enter the probability of a random
cell being alive [0.0-1.0]");
                if(valideInput(input1)){
                        double temp = Double.parseDouble(input1);
                        return temp;
                }
                else if (input1 == null){ //User clicked "Cancel"
              System.exit(0);
              return -1.99999;
             }
            else{
                return promptForAliveChances();
            }
        }
        private boolean valideInput(String input_){
                try{
                        double num = Double.parseDouble(input_);
                        if(num<0 || num > 1){
                                JOptionPane.showMessageDialog(null, "Invalid Input", "alert",
JOptionPane.ERROR_MESSAGE);
                                return false;
                        }
                        return true;
                }
                catch(NumberFormatException e){
                        JOptionPane.showMessageDialog(null, "Invalid Input", "alert",
JOptionPane.ERROR_MESSAGE);
                        return false;
                }
        }
        protected BufferedImage simulatedImage(int width_,int height_){
                while (true) {
                                if (width_ < 0 || height_ < 0)
                                        return null;
                                try {
                                        BufferedImage img = new
BufferedImage(width_,height_,BufferedImage.TYPE_INT_RGB);
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
                                        return img;
                                    } catch (OutOfMemoryError err) {
                                        JOptionPane.showMessageDialog(this, "Ran out of memory! Try
using a smaller image size.");
                                    }
                                }
            }
//nested FractalDisplayPanel class
private class DisplayPanel extends JPanel{
            // panel size
            private final int WIDTH, MAX_X;
            private final int HEIGHT, MAX_Y;
            // image displayed on panel
            private BufferedImage image;
            private Graphics2D g2d;
            private Point currentMouse;
            private Timer timer;
            //use by mouse events
            private int mAliveState;

            private boolean isARGBColor( Point p, int argb ){
                        return (image.getRGB( p.x, p.y ) == argb );
            }
 //---------------------------------------------------------------------------
 // constructor
 public DisplayPanel( BufferedImage image ){
            this.image = image;
            g2d = image.createGraphics();

            // define panel characteristics
            WIDTH = image.getWidth();
            HEIGHT = image.getHeight();
            MILLESECONDS_BETWEEN_FRAMES = 500;
            Dimension size = new Dimension( WIDTH, HEIGHT );
            setMinimumSize( size );
            setMaximumSize( size );
            setPreferredSize( size );
            MAX_X = WIDTH - 1;
            MAX_Y = HEIGHT - 1;

            currentMouse = null;
           //initialize Timer

           timer = new Timer(MILLESECONDS_BETWEEN_FRAMES, new ImageLooper());
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
	this.addMouseListener( new MouseAdapter(){
		public void mousePressed( MouseEvent event ){
			if(!timer.isRunning()){
				currentMouse = event.getPoint();
				int i = ((currentMouse.x >>3) << 3) / 8;
				int j = ((currentMouse.y >>3) << 3) / 8;

				if (isAlive(cell[i][j])){
					mAliveState = DEAD;
				}
				else{
					mAliveState = ALIVE;
				}
			setState(i, j, mAliveState);
			}

		}
	});

	this.addMouseMotionListener( new MouseMotionAdapter(){
		public void mouseDragged(MouseEvent event){
			if(!timer.isRunning()){
				currentMouse = event.getPoint();
				int i = ((currentMouse.x >>3) << 3) / 8;
				int j = ((currentMouse.y >>3) << 3) / 8;

				setState(i, j, mAliveState);

			}
		}
	});
}
 //---------------------------------------------------------------------
	private void setState(int x_, int y_, int state_){
		//wraps around and preten to be a tortorial plane
		if (x_ < 0)
			x_ = 100- 1;
		else if (x_ >= 100)
			x_ = 0;
		if (y_ < 0)
			y_ = 100 - 1;
		else if (y_ >= 100)
			y_ = 0;
		//where (x,y) is the coordinate and state is represented by an int.
		cell[x_][y_] = state_;
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
                //draw the state reflecting changes
                //if(isAlive(state_)){
                        //drawSquare(x_,y_,true,state_);
                //}
                //else{
                        drawSquare(x_,y_,state_);
                //}
        }
 private boolean isAlive(int state_){

                if(state_ == ALIVE || state_ == JUST_ALIVE){
                        return true;
                }
                else{
                        return false;
                }
}
 private void drawSquare(int x_,int y_,int state_){
        //if(alive_){
                g2d.setColor( colorBaseOnState[state_] );
                g2d.fillRect( x_*8, y_*8, 8, 8 );


        repaint();
        //setImage;
}
/*
 public void setImage( BufferedImage src ){
        g2d.drawImage( src,
                0, 0, MAX_X, MAX_Y,
                0, 0, (src.getWidth() - 1), (src.getHeight() - 1),
                null
         );
        repaint();
}
 */
 //---------------------------------------------------------------------------
 // behaviors

        public void paintComponent( Graphics g ){
                super.paintComponent( g );
                g.drawImage( image, 0, 0, null );
        }
        public boolean timerisRunning(){
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
                return timer.isRunning();
        }
        public void stopTimer(){
                timer.stop();
        }
        public void startTimer(){
                timer.start();
        }
        private class ImageLooper implements ActionListener{
                private int [] xMoore = {-1,0,1};
                private  int [] yMoore = {-1,0,1};
                public void actionPerformed(ActionEvent evt){
                        //
                int [][] newCell;
                newCell = new int[100][100];
                //initialize every new cell to dead
                for(int i=0; i<100;i++){
                        for(int j = 0; j<100;j++){
                                newCell[i][j] = DEAD;
                        }
                }

                        for (int j = 0; j < 100; j++) {
                                for (int i = 0; i< 100; i++) {
                                        int aliveNeighbors = 0;
                                        //for (int x=-1; x<2;x++ ){ // cycle through the Moore
neighborhood of the current cell, checking whether each cell is alive
                                                //for (int y=-1; y<2; y++) {
                                                for(int x : xMoore){
                                                        for(int y : yMoore){

                                                                int tempX = i+x;
                                                                int tempY = j+y;
                                                                        //clamp values to wrap around
                                                                        if(tempX<0){
                                                                                tempX = 100-1;
                                                                        }
                                                                        else if(tempX>=100){
                                                                                tempX = 0;
                                                                        }
                                                                        if(tempY<0){
                                                                                tempY=100-1;
                                                                        }
                                                                        else if(tempY >=100){
                                                                                tempY =0;
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
                                        }
                                    if (x == 0 && y == 0)
                                // exclude the cell itself from the count
                                        continue;

                                    if (isAlive(cell[tempX][tempY])) {
                                        ++aliveNeighbors;
                                    }
                                }
                            }

                            if (isAlive(cell[i][j])) { // if the cell is alive
                                if (aliveNeighbors < 2 || aliveNeighbors > 3) {
                                // kill it bc too lonely or too crowded
                                    newCell[i][j]=JUST_DIED;
                                    drawSquare(i,j,JUST_DIED);
                                }
                                else { // keep it alive
                                    if (cell[i][j] == JUST_ALIVE){
                                        newCell[i][j]=ALIVE;
                                        drawSquare(i,j,ALIVE);
                                    }
                                    else{
                        // otherwise, make sure to transfer the cell state over to the new world
                                        //setState(i, j, cell[i][j]);
                                        newCell[i][j] = cell[i][j];
                                        drawSquare(i,j,newCell[i][j]);
                                    }
                                }
                            }
                            else { // if the cell is dead, either...
                                if (aliveNeighbors == 3){
                                // bring it to life, or
                                    //setState(i, j, JUST_ALIVE);
                                    newCell[i][j] = JUST_ALIVE;
                                    drawSquare(i,j,JUST_ALIVE);
                                }
                                else { // keep it dead
                                    if (cell[i][j] == JUST_DIED) {
                        // change state to a "permanent" dead state if it just died last frame
                                        //setState(i, j, DEAD);
                                        newCell[i][j] = DEAD;
                                        drawSquare(i,j,DEAD);
                                    }
                                    else {
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

```
                                        // otherwise, make sure to transfer the cell state over to the new world
                                                                //setState(i, j, cell[i][j]);
                                                                newCell[i][j]= cell[i][j];
                                                                drawSquare(i,j,newCell[i][j]);
                                                        }
                                                }
                                        }
                                }
                        }
                        cell = newCell;
                        repaint();
                }//action performed
        }//image looper

}//DisplayPanel
}//Jframe
```

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

Questions

1. Does the program compile without errors?
Yes.
2. 2. Does the program compile without warnings?
Yes
3. 3. Does the program run without crashing?
Yes
4. 4. Describe how you tested the program.
I ran several test cases with different random inputs for the chances of a cell being alive. And give simple test case on empty world
set is displaying and zooming correctly.
5. Describe the ways in which the program does not meet assignment's specifications.
In the GIFs I provided, the image appear to be tearing a bit, and slow to update for every frame. I based on the assumption that could be the nature of my Graphics Card and the amount of other programs I had running on my laptop at the time.
6. Describe all known and suspected bugs.
There are no known bugs.
7. Does the program run correctly?
Yes

Xiaoxi Zheng
CAP3027
Section 1925
11/2/15
HW11 + Bonus: Slider

Screenshots