CT255 Assignment
Procedurally Generated ‘Caves’ using Cellular Automata
Sample Solution

---

```
import java.awt.Color;
import java.awt.Graphics;
import java.util.*;
import javax.swing.JFrame;

public class ProcGenCaves extends JFrame implements Runnable {

    private int iterations = 4;

    // wall=true, floor=false
    private boolean[][] tiles = new boolean[200][200];
    private int[][] numWallNeighbours = new int[200][200];

    // constructor
    public ProcGen() {
        this.setTitle("Procedurally Generated Caves");
        this.setBounds(10, 10, 800, 800);
        this.setVisible(true);

        // initial setup: each tile has a 60% chance of being wall
        for (int x=0;x<200;x++) {
            for (int y=0;y<200;y++) {
                tiles[x][y] = (Math.random()<=0.6);
            }
        }

        Thread t = new Thread(this);
        t.start();
    }

    // thread entry point/loop
    public void run() {
        // the cave-generating algorithm involves iterating a number of times
        for (int i=0; i<iterations; i++) {
            // 1: sleep for 1 sec
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) { }

            // 2: count the number of neighbours of each tile that are walls
            for (int x=0;x<200;x++) {
                for (int y=0;y<200;y++) {
                    numWallNeighbours[x][y] = 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=199;
                                else if (xxx>199)
                                    xxx=0;
                                int yyy=y+yy;
                                if (yyy<0)
                                    yyy=199;
                                else if (yyy>199)
                                    yyy=0;

                                if (tiles[xxx][yyy])
                                    numWallNeighbours[x][y]++;
                            }
                        }
                    }
                }
            }

            // 3: any tile with 5 or more wall neighbours is a wall, else it's a
floor
            for (int x=0;x<200;x++) {
                for (int y=0;y<200;y++) {
                    tiles[x][y] = (numWallNeighbours[x][y]>=5);
                }
            }

            // 4: repaint
            repaint();
        }
    }

    public void paint(Graphics g) {
        // 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<200;x++) {
             for (int y=0;y<200;y++) {
                if (tiles[x][y]) {
                    g.fillRect(x*4, y*4, 4, 4);
                }
             }
        }
    }

    // application entry point
    public static void main(String[] args) {
```

```
        ProcGen p = new ProcGen();
    }
}
```