COMP 210 - Lab 9

Spring 2014

In this lab you'll work as a group to explore the design space for *Populations* and *Neighborhoods* of *Cells* in our Game of Life (GoL) project.

Data Structure Requirements

We've spent some class time looking at algorithms for managing the GoL population evolution and arrived at two collections of cells: Neighborhoods and Populations. These collections have the following requirements:

1. Neighborhood : a collection of 2D points
   - Easy removal of duplicate entries
   - Easy insertion and removal to construct
   - Easy Traversal for sums
2. Population : integers accessible by 2D points
   - Efficient access of a cell by its 2D point/location

The efficiency requirements for our Neighborhood aren't as tight as those for our Population. The Neighborhood has a max size of eight, where the population contains $r \times c$ cells and each cell will be accessed four to nine times[1]. Arrays are obvious choices, but aren't the only choice. You need to explore the other options that Java has and compare and contrast them.

[1] you should prove to yourself this is true

The Lab

You need to explore the Java collections framework, determine how to use each collection type provided to implement a Neighborhood and a Population of cells, and then compare the trade-offs made with each choice. You should then be prepared to present your choice and the reasoning behind it. Such a presentation should include code samples to demonstrate how key requirements can be met. **No electronic submission is made with this lab. Your lab grade will be based on the presentation made the day after lab. All members of the group should play a role in the presentation.**

The following resources are critical for your research:

- Point class: `http://docs.oracle.com/javase/7/docs/api/java/awt/Point.html`

- Collections Trail: `http://docs.oracle.com/javase/tutorial/collections/`

When considering collections you need to first look at the basic interfaces and then look at the different implementations offered for each interface.