Nathan Gillette

The basis of this project is what color is a certain coordinate in a 2-d array. Green corresponds with a seat, and white is an empty space. These are encapsulated in a 2-d array, which are treated as a seating chart. The dimensions of this array are modeled after 2 rooms in Shineman. These rooms are encapsulated in an array of charts, where the size of this array depends on the number of cores to use. Each seating chart is initialized with a single, but legal seat. Then while looping through the seating chart, it is filled with either blank space or a seat, with bounds checking constantly being done so that there are no illegal seating charts. Randomly, one of the seating charts may swap with a better seating chart. I use the exchanger in the util.concurrency library. This works well for me. Whatever legal layout has the highest number of chairs is the best layout. The GUI updates every second, and after a predetermined number of refreshes(15 is the default), the GUI displays the best layout and ceases updating.

Originally, I was going to use an arraylist of people and go a more OOP route. I was having trouble with that, and I figured just working directly with colors would suffice and be much easier. I was experimenting with other colors, specifically for bounds checking. I moved away from that because one of my functions would overwrite my out-of-bounds, so the room is fully utilized much like 425 or 444 are set up now. No teaching, just a socially distant place for students to get work done. Another big challenger for me was getting the swap to work. It was suggested that I take a look at the exchange function in java.util.concurrent, and that made figuring out how to swap a lot easier.

Link to Screenshot and Screen Recording