Homework #3 Report

**Context:**

There is a movie theatre with a 10 x 10 seating arrangement. The reservations for these seats are sold by the ticket sellers. There are three types of sellers: 1 High Priority Seller (H0), who sells seats starting from the first row, 3 Medium Priority Sellers (M1, M2, M3), who sell seats starting from the middle row, and 6 Low Priority Sellers (L1, L2, L3, L4), who sell seats starting from the last row.

**Problem**:

Ticket sellers are each assigned a number of customers in a queue to handle. They all start at the same time to sell tickets to the customers. The goal of HW#3 is to handle how the seating arrangement and ticket purchases through the use of threads, synchronization, and locks.

**Implementation**:

Our group used 4 Java classes to solve the assignment: Customer, Seller, SeatMatrix, and TestMain. The Customer class comes with a name and an arrival time between 0 and 59 (representing minutes in the hour). The Seller class implements Runnable and takes 4 arguments: a name, the seller type, the number of customers, and a SeatMatrix. Each Seller will also have a randomly generated queue of Customers to handle. The sell() method in the Seller class is how each seller will sell their tickets, using necessary process synchronization and locks to handle the race conditions and ensure that the same seat does not go to two different people – also where the critical region of the code lies. The SeatMatrix class, will contain the shared data for each seller (the clock and the matrix), with methods to determine which seats have already been taken up. The sellers will not be able to modify this data concurrently, simulating thread synchronization. The TestMain class will be the driver class for the seating simulation, creating individual threads for each seller and waking them all up and starting them.