**Movie Theater Seating Challenge**

**Language used:** Java

**Program Description:**

This program takes an input file from command line argument, reads line by line and processes the user requests for reserving seats in the movie theater.

**The algorithm follows following Assumptions:**

1. Customers that come first will be allocated seats in the top rows.
2. Each group will be allocated seats in a single row if there is availability. If the group can’t be seated in the same row, split the group and allocate the people in the group from top.
3. After assigning seats for a request 3 seats are marked as COVID-19 to remind customers of social distancing and these seats can’t be booked.
4. While prioritizing customers comfort and safety this algorithm also gives priority to fill as many seats as possible to maximize profit for the Theater Management so 3 seat gap is maintained between different requests but a buffer row is not maintained.
5. If the numbers of requested seats are not available in the theater then, inform the customer about insufficient seats.
6. If the theater is full Inform the customers that the theater is full.
7. All the requests must be sent in the following format R#### (R followed by the request Id) followed by space and then followed by the number of seats the customer wants. Ex: - R001 9
8. Top rows are given more priority than buffer rows. So the algorithm dose not provide a BufferRow.
9. All the requests that cant be handled due to number of requested seats greater than available seats, theater full, invalid number of seats requested(Ex: -1 ) are informed to the customer via command prompt and in the outputfile a empty seat list is sent beside the request id.

**Balancing customer satisfaction, safety and Theater Management Interests(maximizing profits)**

1. When a customer is reserving seats in group, they would prefer sitting next to each other. So, we will allocate seats for the group in a single row.
2. Since the top seats give a better viewing experience in the movie theater, customers who come first will be allocated seats in the top rows.
3. Social Distancing is over top priority so between any two groups of people there are 3 empty seats.
4. To make sure that we are able to accommodate as many groups as possible and also keep them satisfied by allocating consecutive seats, we start allocating seats from the first column. This will allow us to allocate seats for maximum number of groups in a single row.
5. While prioritizing customers comfort and safety this algorithm also gives priority to fill as many seats as possible to maximize profit for the Theater Management so 3 seat gap is maintained between different requests but a buffer row is not maintained.

**Steps for running**

1. Download and unzip the folder .
2. Compile all the java files in src by going into src folder and open command prompt and use this command **javac MovieTheater/\*.java** (MovieTheater is the package).
3. Run the Runner.java File to get the desired output. The command is

**java -cp . MovieTheater/Runner MovieTheater/inputRequests.txt**

(inputRequests.txt is the argument it Is the path to the input file which contains all the requests)

**Output:- Returns all the not possible allocations along with reasons, prints a Theater Seating Arrangement map and the last line prints the location of the output file.**

1. Create a Jar file so that we can use it to run tests .Use the following command

**jar -cvf MovieTheater.jar MovieTheater/\*.class**

1. Now change the working Directory to test and Run the following command to compile the MovieTheaterTest.java

**javac -cp C:\Users\chait\IdeaProjects\MovieTheater\src\MovieTheater.jar MovieTheaterTest\MovieTheaterTest.java**

(-cp is used to specify class path to the jar file the argument after that is the path of jar file)

1. Run the following command to Run the MovieTheaterTest.java

**java -cp C:\Users\chait\IdeaProjects\MovieTheater\src\MovieTheater.jar MovieTheaterTest\MovieTheaterTest.java**

**Output:- Returns all the testcases along with if they have passed or failed , name of the Test Case.**