Assignment 9

Refer to Canvas for assignment due dates for your section.

Objectives:

- Refactor earlier work to improve modularity.
- Design a solution with MV* architecture.
- Use design patterns to solve common problems.

General Requirements

Create a new Gradle project for this assignment in your ***group*** GitHub repo.

For this assignment, please continue to use packages as in the past. There is only one problem so you may have only one package, but it can be helpful to create additional packages to keep your code organized. The requirements for repository contents are the same as in previous assignments.

New: Please include a ***brief*** write-up of which design pattern(s) you used, where, and why. This is just to help your grader understand your approach.

GitHub and Branches

Each individual group member should create their own branch while working on this assignment. Only merge to the master branch when you're confident that your code is working well. You may submit pull requests and merge to master as often as you like. Guidelines on working with branches are available from the assignment page in Canvas.

To submit your work, push it to GitHub and create a release on your master branch. Only one person needs to create the release.

The problem: Theater reservation system

For this assignment, you'll implement a reservation system for movie theaters that automatically places users in the most desirable seats available for their party. This problem is split into two parts to make it easier to digest but you should only submit a single, complete solution (don't write separate solutions for part 1 and part 2).

Part 1

The user will interact with the system on the command line. The system will prompt the user with the phrase, "What would you like to do?", and the user can enter:

- "reserve <number>" to reserve that number of seats;

- “show” to display the current available seating in the theater;
- “done” to shut down the system.

When a user requests to reserve some number of seats, the system automatically finds the best seats in the theater that share a *single* row. The “best seats” are assumed to be in the rows nearest the middle of the theater (not too close, not too far). If no rows contain enough unreserved seats, the system will apologize and decline to make a reservation. You will take wheelchair-accessibility into account in part 2.

A few points to consider:

- Best seats are filled first-come / first-serve, provided there are enough empty seats in a row to accommodate a party.
- Best seats are determined by proximity to the center row. A seat N rows ahead of the center row is regarded as approximately equivalent to a seat N rows behind the center row. The front row and the back row (in either order) should be the last two rows assigned.
- Left / right / center priority need not be considered. Rows can fill up from left to right or vice versa, or from the center, as long as parties are not separated.
- No party is separated by the system across multiple rows. The number of seats in a single reservation must all be together on the same row, or the reservation will not be made.

A session with the reservations system might look as follows:

```
What would you like to do?
reserve 5
What’s your name?
Emily
I’ve reserved 5 seats for you at the Roxy in row 8, Emily.

What would you like to do?
show
1  _ _ _ _ _ _ _ _ _ _
2  _ _ _ _ _ _ _ _ _ _
3  _ _ _ _ _ _ _ _ _ _
4  _ _ _ _ _ _ _ _ _ _
5  _ _ _ _ _ _ _ _ _ _
6  _ _ _ _ _ _ _ _ _ _
7  _ _ _ _ _ _ _ _ _ _
8  _ X X X X X _ _ _ _
9  _ _ _ _ _ _ _ _ _ _
10 _ _ _ _ _ _ _ _ _ _
11 _ _ _ _ _ _ _ _ _ _
12 _ _ _ _ _ _ _ _ _ _
```

```
13 _ _ _ _ _ _ _ _ _ _
14 _ _ _ _ _ _ _ _ _ _
15 _ _ _ _ _ _ _ _ _ _

What would you like to do?
reserve 11
Sorry, we don’t have that many seats together for you.

What would you like to do?
reserve 9
What’s your name?
Daniel
I’ve reserved 9 seats for you at the Roxy in row 9, Daniel.

What would you like to do?
show
1  _ _ _ _ _ _ _ _ _ _
2  _ _ _ _ _ _ _ _ _ _
3  _ _ _ _ _ _ _ _ _ _
4  _ _ _ _ _ _ _ _ _ _
5  _ _ _ _ _ _ _ _ _ _
6  _ _ _ _ _ _ _ _ _ _
7  _ _ _ _ _ _ _ _ _ _
8  X X X X X X X _ _ _
9  X X X X X X X X X _
10 _ _ _ _ _ _ _ _ _ _
11 _ _ _ _ _ _ _ _ _ _
12 _ _ _ _ _ _ _ _ _ _
13 _ _ _ _ _ _ _ _ _ _
14 _ _ _ _ _ _ _ _ _ _
15 _ _ _ _ _ _ _ _ _ _

What would you like to do?
reserve 5
What’s your name?
Tamara
I’ve reserved 5 seats for you at the Roxy in row 7, Tamara.

What would you like to do?
show
1  _ _ _ _ _ _ _ _ _ _
2  _ _ _ _ _ _ _ _ _ _
3  _ _ _ _ _ _ _ _ _ _
4  _ _ _ _ _ _ _ _ _ _
5  _ _ _ _ _ _ _ _ _ _
```

```
6   _ _ _ _ _ _ _ _ _ _
7   X X X X X _ _ _ _ _
8   X X X X X X X _ _ _
9   X X X X X X X X X _
10  _ _ _ _ _ _ _ _ _ _
11  _ _ _ _ _ _ _ _ _ _
12  _ _ _ _ _ _ _ _ _ _
13  _ _ _ _ _ _ _ _ _ _
14  _ _ _ _ _ _ _ _ _ _
15  _ _ _ _ _ _ _ _ _ _

What would you like to do?
reserve 3
What's your name?
Jean Claude
I've reserved 3 seats for you at the Roxy in row, 8, Jean Claude.
1   _ _ _ _ _ _ _ _ _ _
2   _ _ _ _ _ _ _ _ _ _
3   _ _ _ _ _ _ _ _ _ _
4   _ _ _ _ _ _ _ _ _ _
5   _ _ _ _ _ _ _ _ _ _
6   _ _ _ _ _ _ _ _ _ _
7   X X X X X _ _ _ _ _
8   X X X X X X X X X X
9   X X X X X X X X X _
10  _ _ _ _ _ _ _ _ _ _
11  _ _ _ _ _ _ _ _ _ _
12  _ _ _ _ _ _ _ _ _ _
13  _ _ _ _ _ _ _ _ _ _
14  _ _ _ _ _ _ _ _ _ _
15  _ _ _ _ _ _ _ _ _ _

What would you like to do?
done
Have a nice day!
```

Your system should define the following classes:

- `Seat`. A seat object must have:
    - A name, which is a string value representing a capital letter from A to Z.
    - A "reserved for" value representing the name of the person for whom it has been reserved, or null if the seat has not been reserved.
- `Row`. A row is a list of seats. The Row class **must extend ArrayList of Seat** objects.
    - A Row object should have a row number (1 is closest to the screen, etc).
    - A field indicating whether or not it is wheelchair accessible.

    - The Row constructor should take a number of seats and accessibility as arguments.
- `Theater`. A theater has:
    - A name.
    - A collection of rows. You may implement this however you choose.
    - A non-empty list or array of integers indicating which of the rows are wheelchair accessible.
- `ReservationsService`. This class will implement the actual service as a public method that takes a theater as its argument.
- `ReservationSystem`. This class will only contain a main method. It will create a new instance of Theater and call the service implemented in ReservationsService with the theater object as its argument.

Part 2

Modify your code to take into consideration wheelchair accessibility. We need to make sure that at least one row in every movie theater is wheelchair accessible, and we need to factor this into the reservations process.

The system will function as before, but now, when the user requests to reserve seats it will respond with the prompt, “Do you need wheelchair accessible seats?” If the user answers “yes”, the system will search for the best seats among the rows that are wheelchair accessible. If the user answers “no”, the system will search for the best seats from among the rows that are not wheelchair accessible.

If and only if all other rows are occupied, then the system will reserve seats in accessible rows to users who do not need accessible seats.

The “show” command should indicate which rows are wheelchair-accessible. Whereas seats in rows that are not accessible are represented by “_”, seats in rows that are wheelchair-accessible should be represented by “=”.

General tips/expectations

- Decide where the appropriate functionality should reside. For example, consider how the “show” command should interact with the various classes’ toString() methods.
- Design the algorithm that finds the optimal row (accessible or not) based on proximity to the center row. Use the simplest approach you can to determine the optimal row.
- Ensure that it is not possible to create a Theater without at least one accessible row.
- If the user enters something unexpected or invalid, give them the opportunity to try again—the program should not crash or terminate if the user enters something unexpected.