CS 251 - Project 1

Description

A small concert hall has hired you to create a robust and user-friendly seating management system in Java. The seating arrangement for the event is represented by a 2D character array read from a file, with each row in the chart labeled from 'A' onwards and each column numbered beginning with 1.

Input File Information

The input file will contain information about the number of rows and columns and a grid of characters representing the seating arrangement:

Available Seat: Unicode Character '□' (U+25FB)

Booked Seat: Unicode Character '■' (U+25FC)

Not a Seat: Minus Sign '-'

```
6 20
---□□□□□□--■■□□□□---
--□■■■■□■--□□□□■■■--
-□□□□□■■■--■■■□□□□□-
□□□□□□□□■--□□□□□□□□□
□□□□□□□□□--□□□□□□□□□
□□□□□□□□□--□□□□□□□□□
```

User Requirements

Book a seat: A user can provide the row and column indications to book a specific seat.

Cancel a reservation: If a user no longer requires a seat, they can cancel the reservation by specifying the row and column.

Find N adjacent available seats: If a user wants to find *n* number of seats adjacent to each other, they can do so by providing the number *n*.

Move a reservation: If a user needs to shift their reservation from one seat to another, they can achieve this by providing the row and column for both the original and new seat.

Display the seating chart: After every operation, the updated seating chart will be displayed.

Print Seating Statistics: Information about how many available and booked seats will be displayed to the user after every operation.

Save Seating Chart File: When the user is finished modifying the Seating arrangement, the updated array should be written to a file structured like the input file.

Tasks:

Task 1: File IO

Create two methods to handle File input and Output. The input method should read the input file into a two-dimensional character array. The output method should write the number of rows and columns followed by the character array.

Task 2: Display Seating Information

Create a method for outputting the seating chart to the console. The seating chart should include the Row identifiers as letters and the column identifiers as numbers. Since each column only takes up one column, you may use 2 lines to represent the identifier, with the first line displaying the 10s place and the next line displaying the 1s place. You should also calculate and display how many seats have been booked and how many are available

```
0         1         2
 12345678901234567890
A---□□□□□□--■■□□□□---
B--□■■■■□■--□□□□■■■--
C-□□□□□■■■--■■■□□□□□-
D□□□□□□□□■--□□□□□□□□□
E□□□□□□□□□--□□□□□□□□□
F□□□□□□□□□--□□□□□□□□□
Booked seats: 17/96
Available seats: 79/96
```

Task 3: Booking and Cancellation

Create two more methods to handle booking and cancelling seats. The first method books a seat in the given seating chart at the specified row and column. If the seat is available, it will be marked as booked ('■'). If the seat is already booked or out of bounds, a message will be displayed, and the seating chart will remain unchanged.

The next method cancels a reservation for a seat in the given seating chart at the specified row and column. If the seat is booked, it will be marked as available ('□') If the seat is not booked or out of bounds, a message will be displayed, and the seating chart will remain unchanged.

Task 4: Finding and Booking Adjacent Seats

It is often the case where families, couples, or groups of friends want to book seats together. Create two methods to handle finding and booking groups of available adjacent seats. The first method should take the number of seats to find as a parameter and should return the start position of the adjacent seats. Adjacent available seats are defined as available seats such that all seats are in the same row and no seats are separated by an occupied seat.

The next method should take as input the number of seats to fill and the starting position of the neighboring seats. It should then iterate through the adjacent seats and mark them as booked. You may use your existing booking method for each seat.

Task 5: Menu

The program should implement a menu to allow the user to select between the operations outlined above.

```
1. Book a seat
2. Cancel a reservation
3. Find N adjacent available seats
4. Move a reservation
5. Save and Exit
Enter your choice:
```

Rubric

Task 1: File IO (20 points)

- **File input method (10 points)**
  - Reads input file correctly into a 2D character array.
  - Handles different file formats and edge cases gracefully.
- **File output method (10 points)**
  - Writes the number of rows and columns to the output file.
  - Formats the output correctly, including row and column identifiers.

Task 2: Seating Information (15 points)

- **Seating Chart Output (5 points)**
  - Accurately represents the data in the 2D character array.
  - Uses appropriate row and column identifiers.
  - Handles edge cases, such as empty rows or columns.
- **Booking and availability information (5 points)**
  - Summarizes the number of booked and available seats.
  - Updates the information correctly as seats are booked or canceled.

Task 3: Booking and Cancelation (25 points)

- **Booking method (10 points)**
  - Marks a seat as booked.
  - Handles invalid input gracefully.
  - Updates booking information correctly.
- **Canceling method (15 points)**
  - Marks a seat as available.
  - Handles invalid input gracefully.
  - Updates booking information correctly.

Task 4: Adjacent Seats (25 points)

- **Finding Adjacent Seats (15 points)**
  - Locates groups of available seats.
  - Handles edge cases and seat availability correctly.
- **Booking Adjacent Seats (10 points)**
  - Marks the specified seats as booked.
  - Updates booking information correctly.

Task 5: User interface (15 points)

- **Menu (5 points)**
  - Provides clear and concise options.
  - Guides the user through the program.
- **Input and output (5 points)**
  - Prompts the user for input.
  - Formats output in a readable and informative way.
- **Error handling (5 points)**
  - Provides meaningful error messages.
  - Guides the user to recover from errors.