BattleShip Game

Problem Statement

Design and implement a battleship game to be played between two players until one comes out as the winner.

Requirements:

- The game will be played in a square area of the sea with **NxN grids** which will be called a battlefield. **“N”** should be taken as input in your code.
- The battlefield will be **divided in half** between both the players. So in a NxN battlefield, **NxN/2 grids will belong to PlayerA** and the other **NxN/2 grids will belong to playerB**
- The **size** and **location** of each ship will be taken as input. Each ship will be assumed to be of **Square shape**. **Both the players should be assigned equal fleet.**
- The location of each ship in the NxN grids has to be taken as input (X, Y). X and Y should be integers. For eg. if a ship “SH1” is at (2, 2) and has the size of 4, its corners will be at (0, 0), (0, 4), (4, 0) and (4,4)
- Ships will remain stationary. No two ships should overlap with each other. However they can touch boundaries with each other.
- Each player will fire one missile towards the enemy field during his turn using the **“random coordinate fire”** strategy, which means the missile will hit at a random coordinate of the opponent’s field. It might hit or miss the opponent ship. In either case the turn is then transferred to the other player.
    - In case of a hit, the opponent’s ship is destroyed.
    - In case of a miss, nothing happens.
- **No two missiles should ever be fired at the same coordinates** throughout the course of the game.
- When all the ships of a particular player has been destroyed, he loses the game.

The following APIs have to be implemented:

Mandatory:

- initGame(N)

  This will initialize the game with a battlefield of size NxN. Where the left half of N/2xN will be assigned to PlayerA and the right half will be assigned to PlayerB
- addShip(id, size, x position PlayerA, y position PlayerA, x position PlayerB, y position PlayerB)

This will add a ship of given size at the given coordinates in both the player's fleet.

- startGame()

  This will begin the game, where PlayerA will always take the first turn. The output of each step should be printed clearly in the console.
  For eg.
  PlayerA's turn: Missile fired at (2, 4). "Hit". PlayerB's ship with id "SH1" destroyed.
  PlayerB's turn: Missile fired at (6, 1). "Miss"

Optional

- viewBattleField()

  This will display the battlefield as a NxN grid and all the ships along with the grids occupied by each ship. PlayerA's ship with id SH1 will be marked as A-SH1, with id SH2 as A-SH2 and so on. Whereas PlayerB's ships will be marked as B-SH1, B-SH2 and so on.
  Note: It should mark all the grids occupied by a ship and not just the center coordinate.

Guidelines:

- You should store the data in-memory using a language-specific data-structure.
- You can print the output in the console.
- Implement clear separation between your data layers and service layers.

Expectations:

- Your code should cover all the mandatory functionalities explained above.
- Your code should be executable and clean.
- Your code should be properly refactored, and exceptions should be gracefully handled.
- Appropriate errors should be displayed on console when user input violates the rules of the game.

How will you be evaluated?

- **Code Should be working**
- Code readability and testability
- Separation Of Concerns
- Abstraction
- Object-Oriented concepts.
- Language proficiency.
- Exception Handling
- Scalability

- Test Coverage (Bonus Points)

Sample Execution:

```
>> initGame(6)
>> addShip("SH1", size = 2, 1, 5, 4, 4)
>> viewBattleField()
```

(6, 6)

A-SH 1	A-S H1				
A-SH 1	A-S H1		B-S H1	B-SH 1	
			B-S H1	B-SH 1	

(0, 0)

```
>> startGame()
     PlayerA's turn: Missile fired at (3, 0) : "Miss" : Ships
Remaining - PlayerA:1, PlayerB:1
     PlayerB's turn: Missile fired at (1, 1) : "Miss" : Ships
Remaining - PlayerA:1, PlayerB:1
     PlayerA's turn: Missile fired at (5, 3) : "Hit" B-SH2 destroyed
: Ships Remaining - PlayerA:1,   PlayerB:0
    GameOver. PlayerA wins.
```