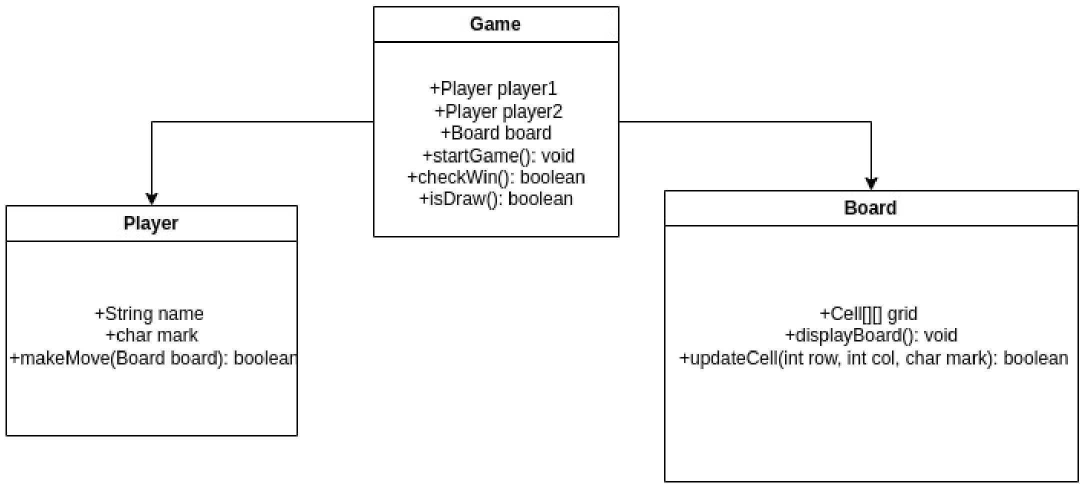
Tic-Tac-Toe in Java :



1. **Analyze (Minimum Requirements and Noun Gathering):**

Minimum Requirements:

- The app should allow two players to play Tic-tac-toe.
- Players take turns to place their mark (X or O) on a 3x3 grid.
- The game should check for win conditions after each turn
    - A player wins if they get three marks in a row, column, or diagonal.
- If the grid is full and no player has won, the game results in a draw.
- Players should be able to restart the game after it ends.

Nouns (Possible Classes):

- **Game**: Represents the overall management of the game
- **Player**: Represents each player in the game
- **Board**: Represents the Tic-tac-toe grid and its state.

Pros of OOAD:

- **Scalability**: Breaks down complex systems into manageable components (e.g., classes).
- **Reusability**: Classes and objects can be reused in other applications.
- **Maintainability**: Easier to update or fix as changes in one class minimally impact others.
- **Encapsulation**: Keeps implementation details hidden, reducing unintended interactions.
- **Design clarity**: UML diagrams and object models improve understanding of the system.

Cons of OOAD:

- **Time-consuming**: Requires significant initial effort for analysis and design.
- **Overhead**: Not suitable for simple projects where procedural programming might suffice.
- **Complexity**: May lead to over-design with unnecessary classes or relationships.
- **Feedback delays**: Customer feedback comes later compared to the MVP approach.