Use Case Description: BoardGame Go

1. Title

Launch and Play a Game of Go (JavaFX Implementation)

2. Primary Actors

- **Player/User:** Interacts with the game by clicking on the board.
- **System/Application:** The JavaFX program managing game initialization, rendering, and move processing.

3. Preconditions

- **Application Launch:** The JavaFX application is successfully started.
- **Board Initialization:** A 19×19 board is created (represented by a two-dimensional array) and displayed on a canvas.
- **Initial Settings:** The board is rendered with grid lines, star points, and no stones. The turn is set to Black (indicating Black moves first).

4. Main Success Scenario (Basic Flow)

1. **Game Initialization:**
    a. The application window opens with the title "围棋游戏".
    b. A Canvas is created with dimensions based on the board size and tile size.
    c. The board array is initialized with all positions set to 0 (empty).
    d. The `drawBoard()` method is called to render the grid, star points, and any existing stones.
    e. Mouse click event handling is enabled on the Canvas.
2. **Player Move Input:**
    a. **User Action:** The player clicks on a desired intersection on the board.
    b. **System Response:** The application calculates the board coordinates by adjusting the

mouse click position relative to the tile size.

3. **Move Validation and Stone Placement:**

    a. **Validation:** The system checks if the clicked coordinates are within bounds and that the corresponding cell in the board array is empty.

    b. Valid Move:

        i. The board array is updated with a value (1 for Black, 2 for White) based on whose turn it is.

        ii. The `drawBoard()` method is called again to refresh the display with the new stone.

        iii. The turn toggles to the opposing player.

    c. Invalid Move:

        i. If the selected position is out-of-bounds or already occupied, an Alert is triggered informing the player with the message "Please select a correct location."

4. **Game Continuation:**

    a. The process repeats as players take turns placing stones until the application is closed or additional game logic (such as game termination or scoring) is introduced.

5. Alternative Flows

- **Invalid Move Attempt:**
    - *Scenario:* The player clicks on a cell that is either occupied or outside the valid board area.
    - *Outcome:* The system displays a warning Alert, and the move is rejected. The turn does not change.
- **No Further Move Handling:**
    - *Scenario:* The game currently lacks features such as capturing stones or scoring.
    - *Outcome:* The game continues solely with stone placement and turn alternation until the application is terminated.

6. Postconditions

- **Board State Update:** The board (both the visual display and the internal array) reflects the stones placed during the session.
- **Turn Management:** The application maintains which player's turn is next.
- **User Feedback:** Players are informed of invalid moves via alerts, ensuring proper move validation.

7. Extensions and Future Considerations

- **Game Rule Enhancements:** Future versions may include capture detection, scoring logic, and end-game conditions.
- **User Interface Improvements:** Additional UI elements such as a status bar, move history, or reset options could enhance the user experience.
- **Error Handling:** More robust error handling could be introduced to manage unexpected input or runtime exceptions.