Game Documentation for **Space Game**

***Black Hole* is an engaging two-player strategy game played on a square grid (e.g., 5x5, 7x7, or 9x9). At the center of the board lies a black hole, which plays a crucial role in the game. Each player starts with n-1 spaceships, which are placed on opposite sides of the board. The spaceships are positioned on the lower diagonal for one player and the upper diagonal for the other, ensuring they are on separate sides.**

The players take turns moving their spaceships across the grid. However, the black hole interferes with the spaceship's movement by making it "slide" rather than move just one step. A spaceship will continue moving in a straight line until it either reaches the edge of the board or diagonally by one position, lands in the black hole, or hits another spaceship. Spaceships cannot jump over one another, so they must avoid or collide with each other when navigating.

Game Rules and Objectives

Objective

The goal is to move half of your spaceships into the black hole before your opponent does.

Setup and Board Configuration

- **Grid Size**: Players select the board size, with options of 5x5, 7x7, or 9x9.
- **Spaceship Placement**:
    - Each player starts with n−1n-1n−1 spaceships, arranged diagonally on opposite sides of the board.
    - Player 1 (Green) starts from the lower diagonal, and Player 2 (Blue) from the upper diagonal.

Black Hole Mechanics

- The black hole sits in the center of the board and alters spaceship movement by causing them to "slide" across the board.
- Once a spaceship begins moving toward the black hole, it continues in a straight line until:

    - It hits the edge of the board,
    - Collides with another spaceship, or
    - Moves diagonally by one position, falling into the black hole.

Gameplay

- Players take turns to move one spaceship per turn.
- Spaceships move in a straight line until the edge of the board, a collision, or the black hole.
- A spaceship cannot jump over another spaceship, so each must be navigated carefully to avoid or strategically collide with an opponent's ship.

Winning Condition

The game ends and declares a winner when one player successfully moves half of their spaceships into the black hole. The board then resets automatically for a new round.

Class Descriptions and UML Diagram

Class Overview

Board

- The Board class manages the game logic, the grid of cells, the player interactions, spaceship movements, and the game flow.
- It initializes the game board, sets up the players, manages the player's moves, and determines if a player wins or loses.

Cell

- The Cell class represents each cell in the board. It holds a spaceship object, and information on whether it's selected or contains a black hole.
- Each cell can detect mouse clicks and is responsible for highlighting the selected spaceship.

Spaceship

- The Spaceship class represents a player's spaceship. It holds information on the ship's position and color and identifies its owning player.

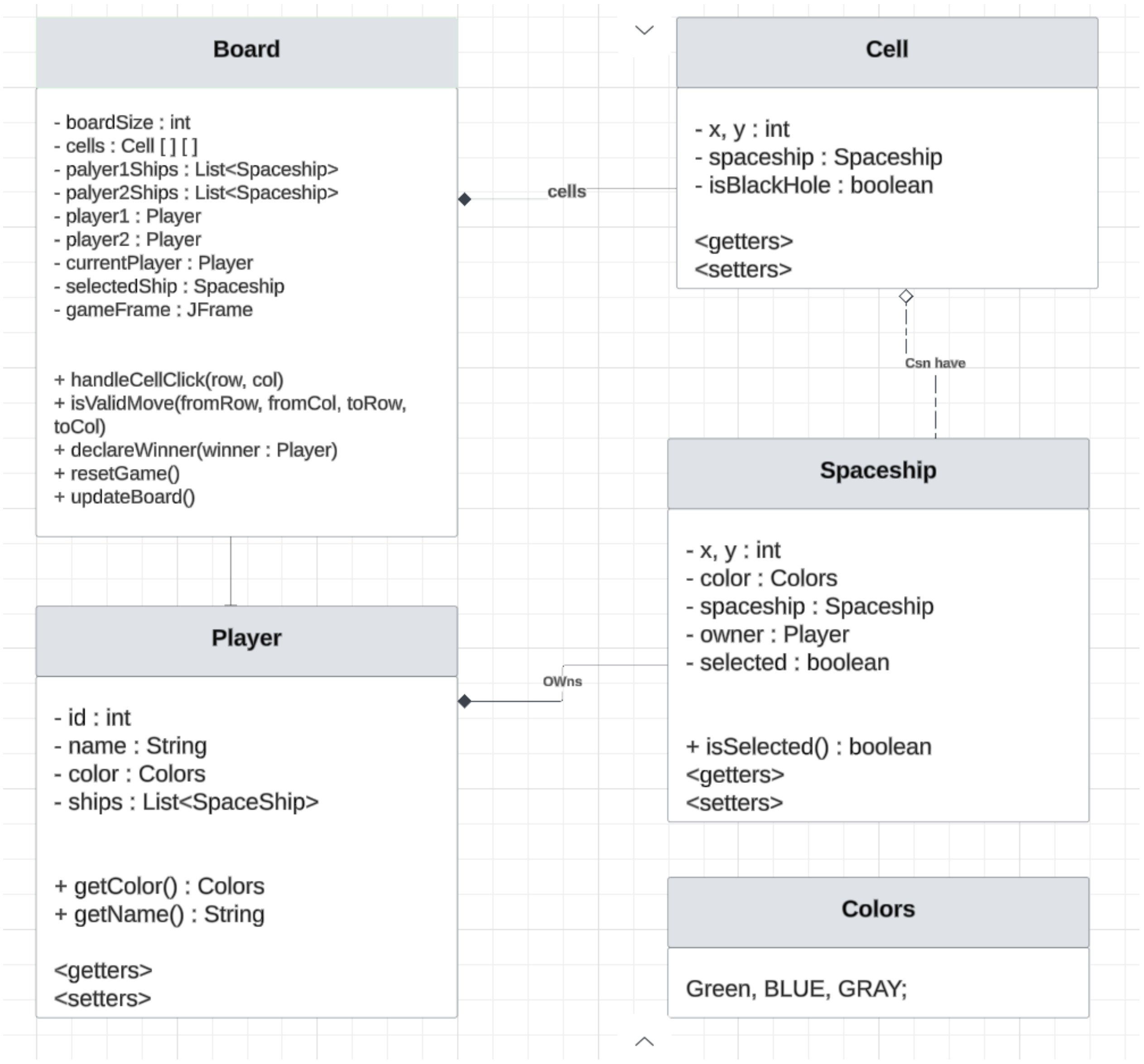
Player

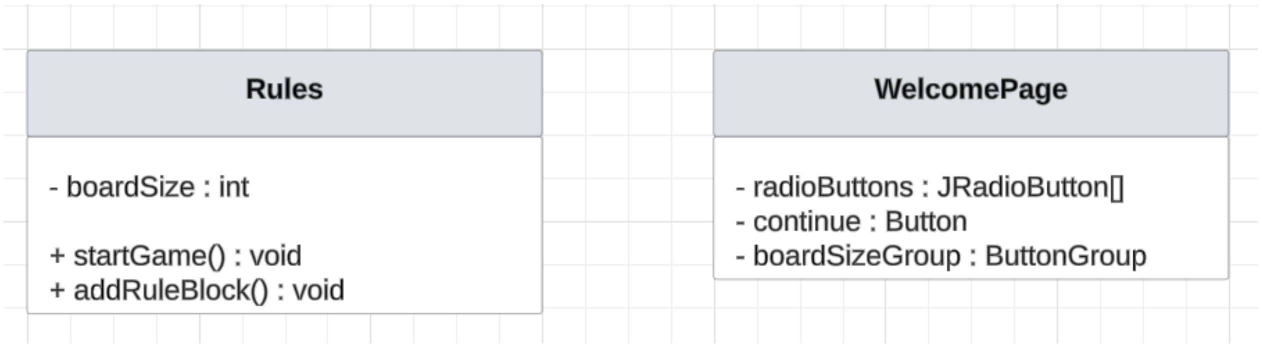
- The Player class holds the player's information, such as their ID, color, and list of spaceships.

Colors

- The Colors class defines constant colors for both players (e.g., GREEN and BLUE) to represent them on the game board.

UML Diagram





`handleCellClick` Method

The `handleCellClick` method in the `Board` class is a core part of the game's event-handling mechanism. It processes mouse click events on the game board's cells and updates the game state accordingly. Here's an overview of its functionality:

```
public void handleCellClick(int row, int col)
```

Parameters:

- `row` - The row index of the clicked cell.
- `col` - The column index of the clicked cell.

Functional Flow

1. Initial Checks

- The method first identifies if a cell contains a **spaceship** or if it's an empty cell:
  - **If the clicked cell contains the same spaceship currently selected**:
    - The ship is **deselected**.
    - The method exits early.
  - **If the cell is empty**:
    - It validates if a move can be performed based on the selected ship's position and movement rules.

2. Handling Empty Cells

When an empty cell is clicked:

- If no ship is currently selected, the click is ignored.
- If a ship is selected:
  - **X-axis movement**:
    - Determines valid movement along the row.
    - Stops movement if encountering a **Black Hole** or another spaceship.
  - **Y-axis movement**:
    - Determines valid movement along the column.
    - Stops movement if encountering a **Black Hole** or another spaceship.
- The ship is moved to the calculated position:
  - The previous cell is cleared.
  - The ship's new position is updated in the new cell.
  - If the new cell contains a **Black Hole**, the ship is removed, and the current player gains a point.

3. Handling Cells with Spaceships

If a clicked cell contains a spaceship:

- The method checks if the ship belongs to the **current player**.
- If the ship belongs to the current player:
  - It is selected for movement.
  - The ship's cell is visually marked as selected.
- If another ship is already selected, no action is taken.

After a valid move:

- The turn switches to the other player.
- The player's UI square color updates to indicate whose turn it is.

How the Game Works

Player Setup:

- Upon starting the game, two players are initialized with their respective colors (Green for Player 1 and Blue for Player 2). The board is set up with spaceships placed for each player in specific rows and columns.

Handling Mouse Clicks:

- The game responds to mouse clicks to either select a spaceship or move it. Once a spaceship is selected, valid move options are presented, and the spaceship can be moved to an adjacent empty cell.

Moving Spaceships:

- Spaceships move in a straight line either to the edge of the board, to a colliding ship, or diagonally to the black hole.
- If a spaceship moves into the black hole, it is removed from the board.

Score and Victory Conditions:

- After each turn, the game checks if the player has captured enough spaceships or forced the opponent into black holes. If this condition is met, the game announces the winner.

Game Rules

Objective:

- The goal of the game is to either capture a specific number of the opponent's spaceships or force them to fall into black holes. The game is won when one player achieves this objective.

Board Setup:

- The board consists of a grid where each player places their spaceships in the first and last rows. The grid contains black holes at random locations, and players' spaceships are controlled via clicks.

Gameplay:

- Players take turns to move their spaceships. They can only move to an adjacent square (up, down, left, right, or diagonally) that is empty. If the spaceship moves into a black hole, the spaceship is removed from the board.
- If a player's spaceship lands in a cell that has an enemy spaceship, the enemy spaceship is captured and removed from the board.

- Players cannot move their spaceship to a cell that already contains another spaceship.
- Each player has a score counter, and the player with the most captured spaceships or successful black hole moves wins.

Turn Structure:

- The game alternates turns between Player 1 (Green) and Player 2 (Blue).
- Each player selects a spaceship and moves it to an adjacent empty cell.
- The game checks if the move is valid and applies the consequences (capture or black hole).

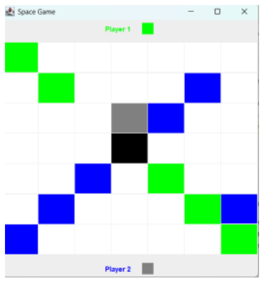
Winning the Game:

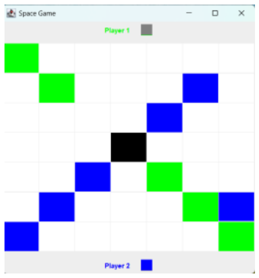
- A player wins if they either capture a specific number of the opponent's spaceships or force them into black holes.

White-Box Testing

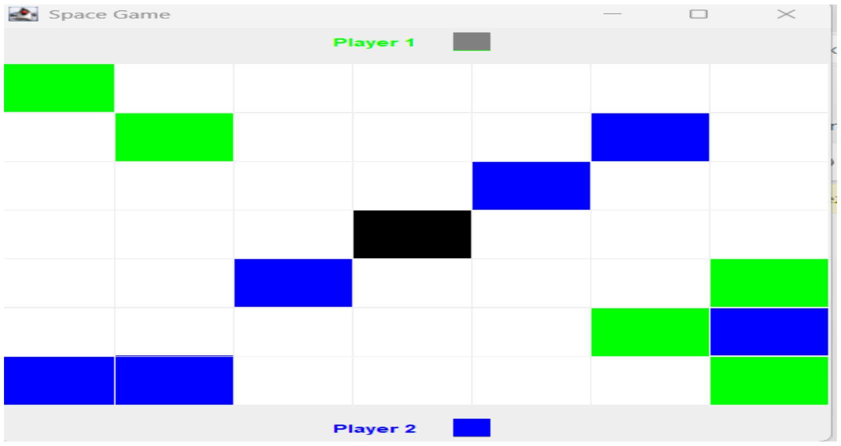
White-box testing is the process of testing internal structures or workings of a game, where the tester has access to the internal code and logic. Below are some possible white-box test cases for the **Board** class:

1. **Test Case 1**: Test spaceship removal in black hole.
   - **Input**: Move a spaceship into the black hole.
   - **Expected Output**: The spaceship is removed, and the cell is empty.

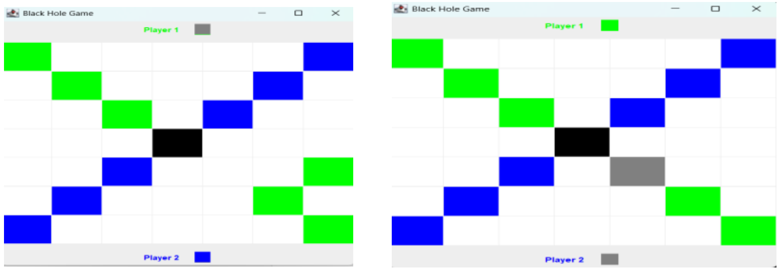




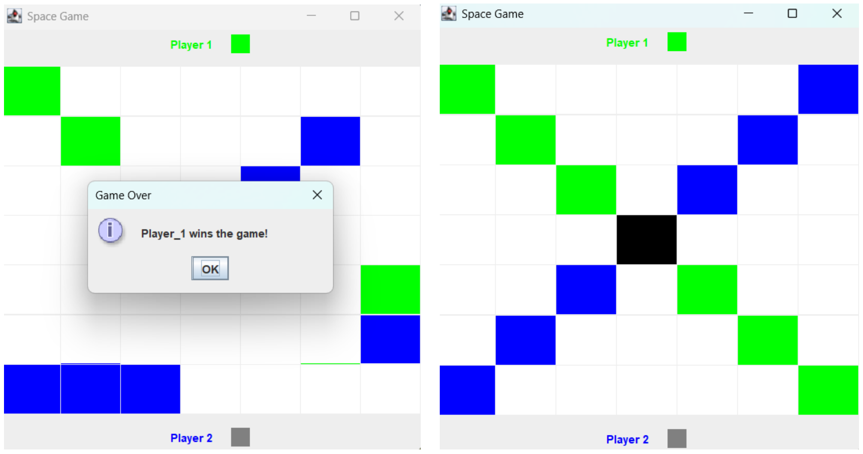
2. **Test Case 2**: Test capturing an opponent's spaceship.
   - **Input**: Move Player 1's spaceship into a cell occupied by Player 2's spaceship.
3. **Expected Output**: Player 2's spaceship is blocked,



4. **Test Case 3**: Test turn alternation.
   - **Input**: Player 1 makes a move.
   - **Expected Output**: The turn switches to Player 2.



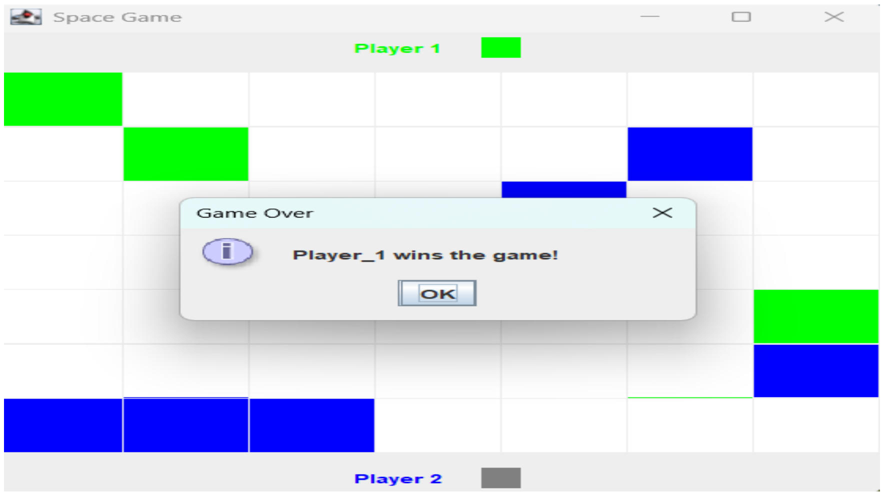
5. **Test Case 4**: Test game reset after a win.
   - **Input**: Player 1 achieves the win condition.
   - **Expected Output**: The board resets, scores clear, and Player 1 starts the new game.



Player_1 wins the game!

6. **Test Case 5**: Test winning by entering the black hole.

- **Input**: Player 1 moves half of their spaceships into the black hole.
- **Expected Output**: Game ends with a "Player 1 wins" message.



Game Over

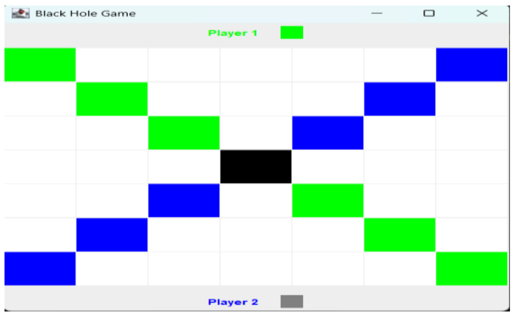
Player_1 wins the game!

7. **Test Case 6**: Test winning by capturing spaceships.
   - **Input**: Player 1 captures the required number of Player 2’s spaceships.
   - **Expected Output**: Game ends with a "Player 1 wins" message

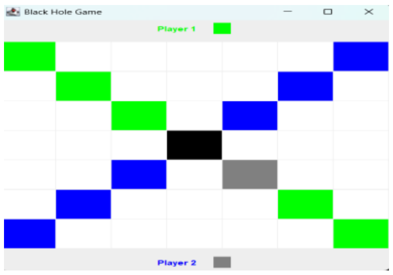
Black-Box Testing

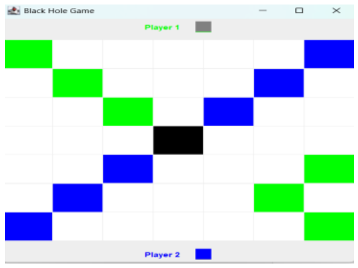
Black-box testing is the process of testing the game based solely on inputs and expected outputs without knowledge of the internal code. Here are some possible black-box test cases:

1. **Test Case 1**: Test board initialization.
   - **Input**: Start a new game.
   - **Expected Output**: The grid displays spaceships for both players on opposite sides, with a black hole in the center.

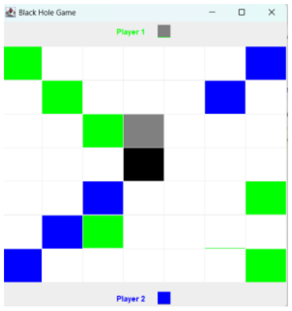


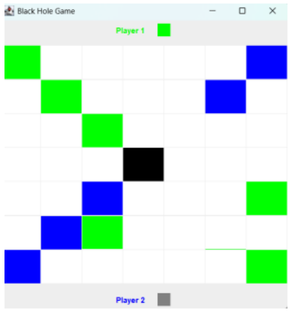
2. **Test Case 2**: Test moving to an adjacent empty cell.
    - **Input**: Move a spaceship from (0,0) to (0, n-2).
    - **Expected Output**: The spaceship moves to (0,n-2) if the cell is empty.



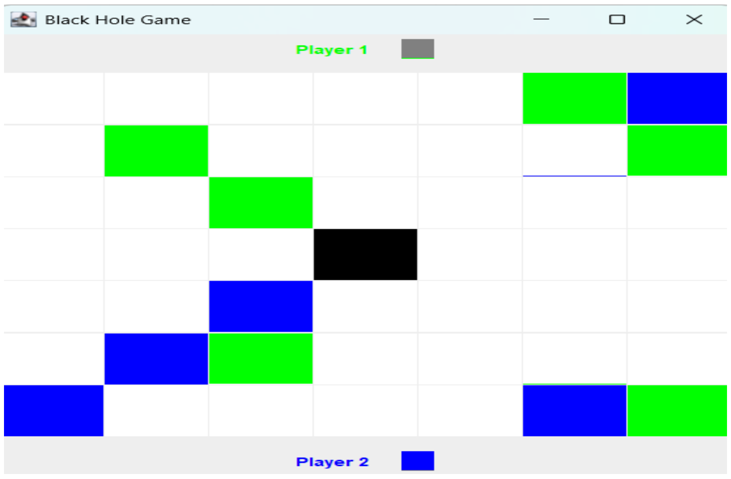


3. **Test Case 3**: Test black hole interaction.
    - **Input**: Move a spaceship into the black hole's cell.
    - **Expected Output**: The spaceship is removed from the board.

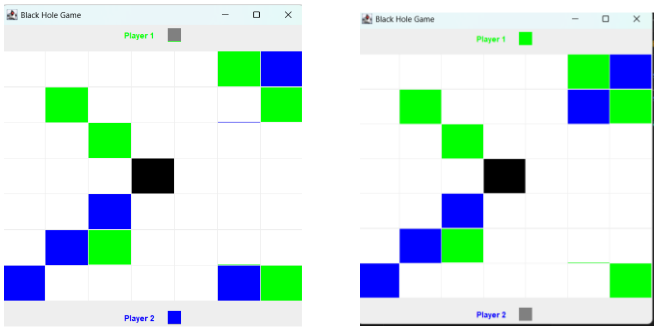




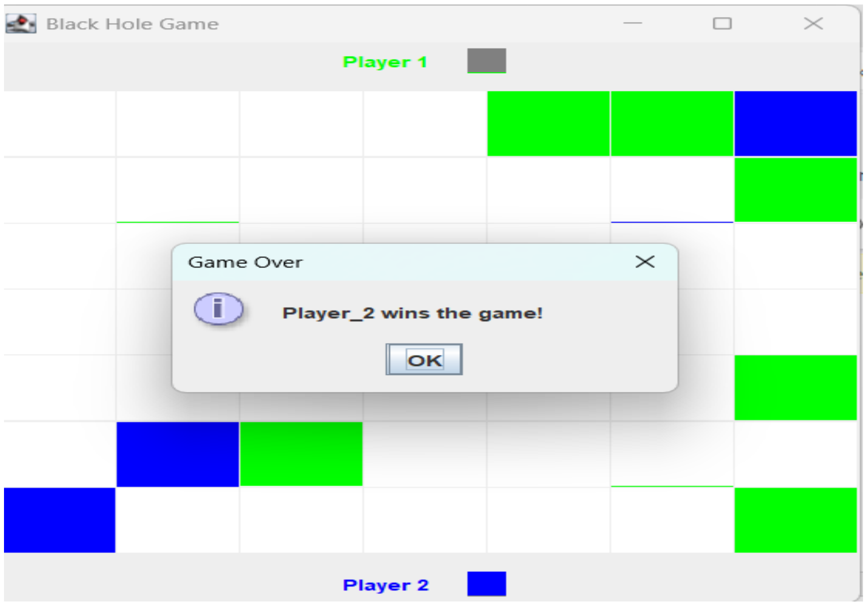
4. **Test Case 4**: Test capturing an opponent's spaceship.
    - **Input**: Player 1 moves a spaceship into a cell occupied by Player 2's spaceship.
    - **Expected Output**: Player 2's spaceship are blocking its path so player 1 can't move his ship in this situation.



5. **Test Case 5**: Test turn alternation.
    - **Input**: Player 1 makes a move.
    - **Expected Output**: Control switches to Player 2.



6. **Test Case 6**: Test winning condition announcement.
    - **Input**: Player 1 meets the win condition by capturing or moving spaceships into the black hole.
    - **Expected Output**: The game displays "Player 1 wins" and resets for a new game.

7. 

Player_2 wins the game!