Java Board Game: A Strategic Two-Player Challenge

This Java-based board game pits a human player against a computer opponent on a customizable grid. Players take turns moving their pieces, aiming to trap their opponent or block their moves. Featuring a robust AI, intuitive interface, and precise error handling, the game is designed for modularity and extensibility, with plans for multiple games and a graphical UI. Built with JDK 17, it's perfect for strategy enthusiasts and developers alike.

Features

- **Customizable Board**: Set board size and place black square obstacles.
- **Two-Player Gameplay**: Human (Player B) vs. Computer (Player A) with strategic moves.
- **Smart AI**: Powered by a minimax algorithm for challenging computer moves.
- **Error Handling**: Detailed messages with coordinates (e.g., "Cannot move down 2: out of bounds at (4,3)").
- **Clear Display**: Grid with row/column indices for easy navigation.
- **Modular Design**: Separates state (`BoardState`), logic (`GameLogic`), and control (`Main`).

Getting Started

Prerequisites

- Java Development Kit (JDK) 17 or later.
- A terminal or IDE (e.g., IntelliJ, Eclipse).

Installation

1. Clone the repository:

```
git clone https://github.com/yourusername/java-board-game.git
cd java-board-game
```

2. Compile the Java files:

```
javac Main.java BoardState.java GameLogic.java MoveResult.java
   Direction.java
```

3. Run the game:

```
java Main
```

How to Play

1. **Setup**:
   - Enter board size (e.g., “4 4” for 4x4).
   - Optionally set black squares (“yes” to add, e.g., at (1,1)).
   - Specify starting positions for Player A (computer) and Player B (you).
2. **Gameplay**:
   - Players move pieces (A or B) up to 2 squares in directions (e.g., “up$_r ight'').Inputmovesas“direct$

- Computer uses minimax to plan moves.
- Game ends when a player cannot move (win/loss) or ties.

3. **Example Input/Output**:

```
Give number of rows and columns:
4 4
Do you want to set any black squares? Type "yes" for positive:
no
Give i and j of player A (Computer):
0 0
Give i and j of player B (You):
3 3
STARTING POSITION:
  0 1 2 3
0|A| | | |
1| | | | |
2| | | | |
3| | | |B|
```

Project Structure

File	Description
`Main.java`	Orchestrates setup, game loop, AI, and user input.
`BoardState.java`	Manages board state, player positions, and display.
`GameLogic.java`	Handles move validation, AI logic, and win conditions.
`MoveResult.java`	Enum for move outcomes (e.g., SUCCESS, OUT$_O F_B OUNDS$).
`Direction.java`	Enum for move directions (e.g., up, down$_r ight$).

Future Plans

- Introduce `GameManager` to manage game flow and multiple games.
- Add `Player` class for scalable player data.
- Implement player position validation (no black squares or overlap).
- Develop a graphical UI (e.g., JavaFX) for enhanced interaction.
- Support multiplayer modes (e.g., human vs. human).

Contributing

Contributions are welcome! Please:

- Fork the repository and create a pull request.
- Follow Java coding standards.
- Test changes thoroughly (e.g., 4x4 board, A at (1,0), B at (0,0)).

License