ChineseChekkers

Overview

ChineseChekkers is a multiplayer board game implemented in Java. This project includes both server and client components, allowing multiple players to connect and play the game.

Prerequisites

Before you begin, ensure you have the following installed on your system:

- Java Development Kit (JDK) 8 or higher
- Apache Maven

How to Build and Run

Building the Project

1. **Clone the repository:**

```
git clone https://github.com/your-repo/chinesechekkers.git
cd chinesechekkers
```

2. **Build the project using Maven:**

```
mvn clean install
```

Running the Project

Running the Server

1. **Navigate to the target directory:**

```
cd target
```

2. **Run the server:**

```
java -cp classes server.GameServer
```

Running the Client

1. **Navigate to the target directory:**

```
cd target
```

2. **Run the client:**

```
java -cp classes client.GameClient
```

Project Structure

The project has the following structure:

```
ChineseChekkers/
├── .gitignore
├── docs/
│   ├── html/
│   │   ├── annotated.html
│   │   ├── classclient_1_1ClientConnection-members.html
│   │   ├── classclient_1_1ClientConnection.html
│   │   ├── classclient_1_1ClientInputHandler-members.html
│   │   ├── classclient_1_1ClientInputHandler.html
│   │   ├── classclient_1_1ClientOutputHandler__coll__graph.md5
│   │   ├── classclient_1_1ClientOutputHandler__inherit__graph.md5
│   │   └── ...
│   ├── latex/
│   │   ├── annotated.tex
│   │   └── ...
├── Doxyfile
├── pom.xml
├── Readme.md
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── client/
│   │   │   │   ├── ClientConnection.java
│   │   │   │   ├── ClientInputHandler.java
│   │   │   │   ├── ClientOutputHandler.java
│   │   │   │   └── GameClient.java
│   │   │   ├── GUI/
│   │   │   │   ├── ClientStartController.java
│   │   │   │   ├── GameOverController.java
│   │   │   │   ├── InGameClientController.java
```

```
│   │   │   │   ├── InGameClient.fxml
│   │   │   │   └── GameOver.fxml
│   │   │   ├── server/
│   │   │   │   ├── GameServer.java
│   │   │   │   └── PlayerHandler.java
│   │   │   ├── game/
│   │   │   │   ├── Game.java
│   │   │   │   ├── board/
│   │   │   │   │   ├── Board.java
│   │   │   │   │   ├── StandardBoard.java
│   │   │   │   ├── move/
│   │   │   │   │   ├── Move.java
│   │   │   │   ├── state/
│   │   │   │   │   ├── GameState.java
│   │   │   │   │   ├── WaitingForPlayersState.java
│   │   │   │   │   ├── GameInProgressState.java
│   │   │   │   │   ├── GameOverState.java
│   │   │   ├── rules/
│   │   │   │   ├── GameRuleSet.java
│   │   │   │   ├── MultipleJumpsRuleSet.java
│   │   │   │   ├── BananaJump.java
│   │   │   ├── utils/
│   │   │   │   ├── SerializationUtils.java
│   │   │   │   ├── message/
│   │   │   │   │   ├── Message.java
│   │   │   │   │   ├── MessageType.java
│   │   │   │   │   ├── MessageUtils.java
│   ├── test/
│       ├── java/
│       │   ├── client/
│       │   ├── server/
│       │   ├── common/
│       └── resources/
├── UML/
│   ├── Idea.md
```

UML Diagrams

The UML diagrams for the project can be found in the UML directory. These diagrams provide a visual representation of the project's architecture and design patterns used.

Design Patterns

The project utilizes several design patterns, including:

1. **Singleton**: Ensures that there is only one instance of the server managing all connections and games.
2. **Abstract Factory**: Allows for the creation of different game variants (standard and custom boards and rules) without changing the main game logic.

3. **Factory Method**: Responsible for creating appropriate instances of boards and game rules.
4. **Observer**: Notifies players about moves, allowing them to see the effects of moves made by other players.
5. **Strategy**: Allows for dynamic assignment of different sets of game rules.
6. **Facade**: Simplifies client operations by providing a unified interface to manage the connection, input, and output.
7. **State**: Manages the different states of the game (waiting for players, player's turn, game over) and changes the behavior of the game based on its state.

License