Programozás alapjai 3

Házifeladat

2024.11.23

Székely Bence
B1WYBR

Task description

The game of checkers is a widely known and popular board game played all over the world. It is a two-player game where one player controls the black pieces and the other controls the red ones. Both players start the game with 12 pieces each. Unlike chess, the traditional game begins with the dark pieces, and players take turns alternately.
The objective of the game is to capture all of your opponent's pieces or render them immovable. Any piece that reaches the opposite row of the board is promoted to a king. Kings can move more freely than regular pieces, as they are allowed to move and capture both forward and backward.

Game rules

1. Objective:

- The goal of the game is to capture all of your opponent's pieces or block them so they cannot make a move.

2. Setup:

- The game is played on an 8x8 board with alternating dark and light squares.
- Each player starts with 12 pieces placed on the dark squares of the first three rows closest to them.

3. Movement:

- Pieces can only move diagonally forward to an adjacent empty square.
- If a piece reaches the farthest row from its starting position, it is promoted to a "king."

4. Kings:

- Kings can move diagonally both forward and backward.
- To indicate a piece is a king, place another piece on top of it or flip it over.

5. Capturing:

- If an opponent's piece is adjacent to a player's piece and there is an empty square directly on the other side,
the player must jump over the opponent's piece and capture it.

6. Mandatory Captures:

- If a capture move is available, the player must take it.

7. Winning the Game:

- A player wins by capturing all the opponent's pieces or blocking them so they cannot make a move.

User Manual for the Checkers Game

Introduction
This user manual provides instructions on how to use the Checkers game application. The game allows two players to play a game of checkers on a graphical interface.

Installation
1. **Prerequisites**: Ensure you have Java installed on your system.
2. **Download**: Download the game application from the provided source.
3. **Run**: Execute the `GameFrame` class to start the game.

Main Window
When you start the game, the main window will appear. It consists of the following components:
- **CheckerBoardPanel**: Displays the checkerboard where the game is played.
- **Current Player Label**: Shows the current player's turn.
- **Red Captured Label**: Displays the number of red pieces captured.
- **Black Captured Label**: Displays the number of black pieces captured.
- **Menu Bar**: Contains options to load, save, and view game rules.

Menu Options
- **File Menu**:
  - **Load**: Load a previously saved game.
  - **Save**: Save the current game state.
- **Help Menu**:
  - **Game Rules**: View the rules of the game.

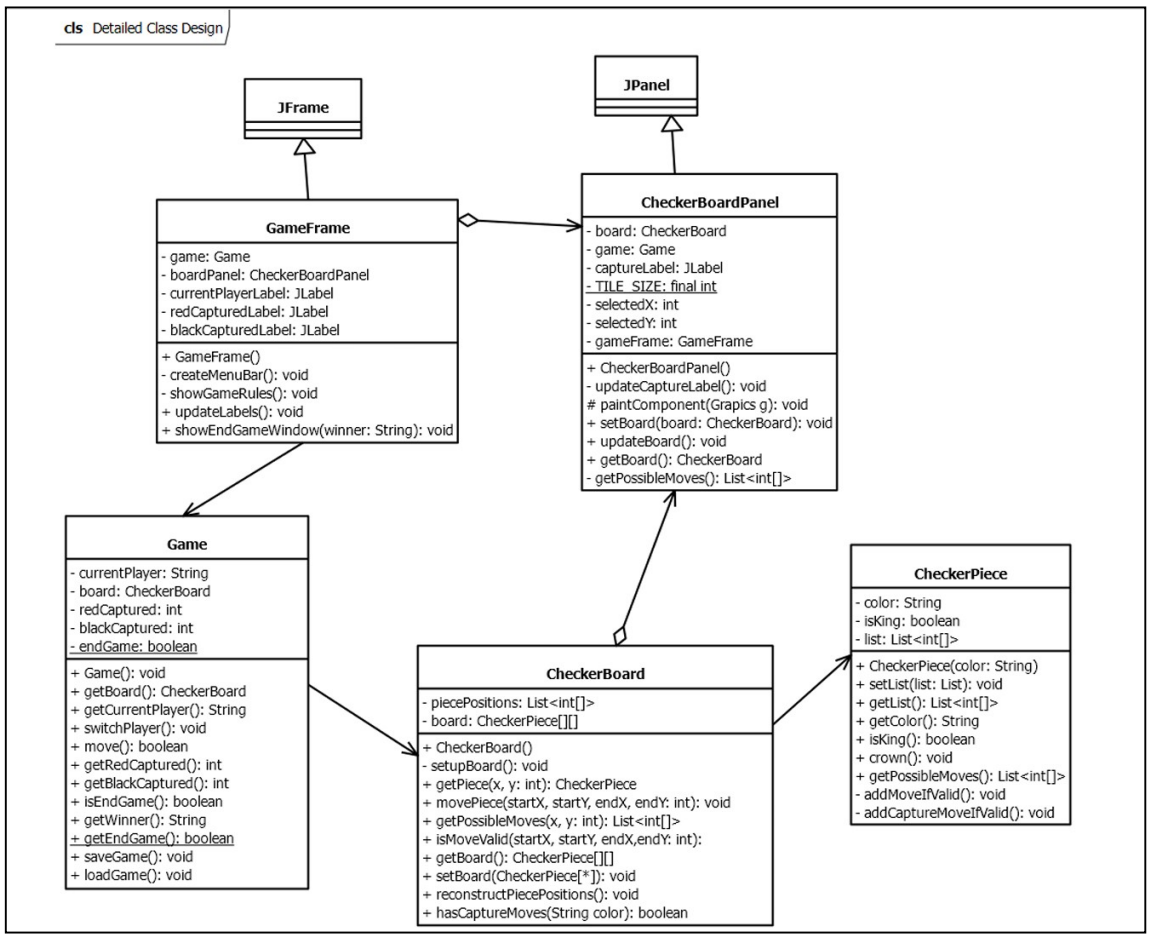
How to Play
1. **Starting the Game**: The game starts with the red player.
2. **Making a Move**: Click on a piece to select it, then click on a valid destination to move the piece.
3. **Capturing Pieces**: Jump over an opponent's piece to capture it.
4. **King Pieces**: Reach the opposite end of the board to crown a piece as a king.
5. **Winning the Game**: Capture all opponent's pieces or block them from making a move.

Saving and Loading Games
- **Saving**: Use the `Save` option in the `File` menu to save the current game state.
- **Loading**: Use the `Load` option in the `File` menu to load a previously saved game.

Viewing Game Rules
- **Game Rules**: Select `Game Rules` from the `Help` menu to view the rules of checkers.

Troubleshooting
- **Error Loading Game**: Ensure the saved game file is in the correct format and location.
- **Game Not Starting**: Verify that Java is installed and properly configured on your system.

Class Documentation



game.CheckerBoard Class Reference

Public Member Functions

- **CheckerBoard** ()
- **CheckerPiece getPiece** (int x, int y)
- void **movePiece** (int startX, int startY, int endX, int endY)
- List< int[]> **getPossibleMoves** (int x, int y)
- boolean **isMoveValid** (int startX, int startY, int endX, int endY)
- void **printList** ()
- **CheckerPiece**[][] **getBoard** ()
- void **setBoard** (**CheckerPiece**[][] board)
- void **reconstructPiecePositions** ()
- boolean **hasCaptureMoves** (String color)

Detailed Description

The CheckerBoard class represents the board of a checkers game. The board is a 8x8 grid of CheckerPiece objects. The class has methods to get a piece at a given position, move a piece, get possible moves for a piece, check if a move is valid, and check if there are any capture moves available for a player.

Constructor & Destructor Documentation

game.CheckerBoard.CheckerBoard ()

Constructs a new CheckerBoard object and initializes the board with pieces in their starting positions.

Member Function Documentation

CheckerPiece[][] game.CheckerBoard.getBoard ()

Returns the list of piece positions on the board.

Returns

the list of piece positions on the board

CheckerPiece game.CheckerBoard.getPiece (int *x*, int *y*)

Returns the piece at the given position on the board.

Parameters

x	the x-coordinate of the position
y	the y-coordinate of the position

Returns

the piece at the given position, or null if there is no piece

List< int[]> game.CheckerBoard.getPossibleMoves (int *x*, int *y*)

Returns a list of possible moves for the piece at the given position on the board.

Parameters

x	the x-coordinate of the position
y	the y-coordinate of the position

Returns

a list of possible moves for the piece at the given position

boolean game.CheckerBoard.hasCaptureMoves (String *color*)

Checks if there are any capture moves available for a player of the given color.

Parameters

color	the color of the player

Returns

true if there are capture moves available, false otherwise

boolean game.CheckerBoard.isMoveValid (int *startX*, int *startY*, int *endX*, int *endY*)

Checks if a move from the start position to the end position is valid.

Parameters

startX	the x-coordinate of the start position
startY	the y-coordinate of the start position
endX	the x-coordinate of the end position
endY	the y-coordinate of the end position

Returns

true if the move is valid, false otherwise

void game.CheckerBoard.movePiece (int *startX*, int *startY*, int *endX*, int *endY*)

Moves a piece from the start position to the end position on the board. Handles capturing of pieces and promotion to king.

Parameters

startX	the x-coordinate of the start position
startY	the y-coordinate of the start position
endX	the x-coordinate of the end position
endY	the y-coordinate of the end position

void game.CheckerBoard.printList ()

Prints the positions of the pieces on the board.

void game.CheckerBoard.reconstructPiecePositions ()

Rebuilds the list of piece positions on the board. This method should be called after moving a piece to update the list of piece positions.

void game.CheckerBoard.setBoard (CheckerPiece *board*[][])

Sets the board to the given board.

Parameters

board	the board to set

The documentation for this class was generated from the following file:

- src/main/java/game/**CheckerBoard.java**

game.CheckerBoardPanel Class Reference

Inheritance diagram for game.CheckerBoardPanel:

Public Member Functions

- **CheckerBoardPanel** (**CheckerBoard** board, **Game** game, **GameFrame** gameFrame)
- void **setBoard** (**CheckerBoard** board)
- void **updateBoard** ()
- **CheckerBoard getBoard** ()

Protected Member Functions

- void **paintComponent** (Graphics g)

Detailed Description

The CheckerBoardPanel class represents the panel that displays the checkerboard. It extends the JPanel class and contains a CheckerBoard object to represent the board, a JLabel to display a message if a capture move is required, and fields to keep track of the selected position.

Constructor & Destructor Documentation

game.CheckerBoardPanel.CheckerBoardPanel (CheckerBoard *board*, Game *game*, GameFrame *gameFrame*)

Constructs a new CheckerBoardPanel object with the given CheckerBoard, Game, and GameFrame objects.

Parameters

board	the CheckerBoard object representing the board
game	the Game object representing the game
gameFrame	the GameFrame object representing the game frame

Member Function Documentation

CheckerBoard game.CheckerBoardPanel.getBoard ()

Returns the CheckerBoard object representing the board.

Returns

the CheckerBoard object representing the board

void game.CheckerBoardPanel.paintComponent (Graphics *g*) `[protected]`

Paints the checkerboard and pieces on the panel.

Parameters

g	the Graphics object used to paint the components

void game.CheckerBoardPanel.setBoard (CheckerBoard *board*)

Sets the CheckerBoard object for the panel.

Parameters

board	the CheckerBoard object representing the board

void game.CheckerBoardPanel.updateBoard ()

Updates the board by repainting the panel.

The documentation for this class was generated from the following file:

- src/main/java/game/**CheckerBoardPanel.java**

game.CheckerPiece Class Reference

Public Member Functions

- **CheckerPiece** (String color)
- void **setList** (List< int[]> list)
- List< int[]> **getList** ()
- String **getColor** ()
- boolean **isKing** ()
- void **crown** ()
- List< int[]> **getPossibleMoves** (int x, int y, **CheckerBoard** board)

Detailed Description

The CheckerPiece class represents a piece on the checkerboard. Each piece has a color ("RED" or "BLACK") and a boolean flag to indicate if it is a king. The class has methods to get and set the color, check if the piece is a king, crown the piece as a king, get the possible moves for the piece, and get the list of possible moves.

Constructor & Destructor Documentation

game.CheckerPiece.CheckerPiece (String *color*)

Constructs a new CheckerPiece object with the given color and sets the king flag to false.

Parameters

color	the color of the piece ("RED" or "BLACK")

Member Function Documentation

void game.CheckerPiece.crown ()

Crowns the piece as a king.

String game.CheckerPiece.getColor ()

Returns the color of the piece.

Returns

the color of the piece

List< int[]> game.CheckerPiece.getList ()

Returns the list of possible moves for the piece.

Returns

the list of possible moves

List< int[]> game.CheckerPiece.getPossibleMoves (int *x*, int *y*, CheckerBoard *board*)

Returns a list of possible moves for the piece at the given position on the board.

Parameters

x	the x-coordinate of the position

y	the y-coordinate of the position
board	the CheckerBoard object representing the board

Returns

a list of possible moves for the piece at the given position

boolean game.CheckerPiece.isKing ()

Returns whether the piece is a king.

Returns

true if the piece is a king, false otherwise

void game.CheckerPiece.setList (List< int[]> *list*)

Sets the list of possible moves for the piece.

Parameters

list	the list of possible moves

The documentation for this class was generated from the following file:

- src/main/java/game/**CheckerPiece.java**

game.Game Class Reference

Public Member Functions

- **Game** ()
- **CheckerBoard getBoard** ()
- String **getCurrentPlayer** ()
- void **switchPlayer** ()
- boolean **move** (int startX, int startY, int endX, int endY)
- int **getRedCaptured** ()
- int **getBlackCaptured** ()
- boolean **isEndGame** ()
- String **getWinner** ()
- void **saveGame** ()

Static Public Member Functions

- static boolean **getEndGame** ()
- static **Game loadGame** ()

Detailed Description

The Game class represents a game of checkers. The game has a CheckerBoard object to represent the board, a currentPlayer field to keep track of the current player, and redCaptured and blackCaptured fields to keep track of the number of pieces captured by each player. It also has a static endGame field to keep track of whether the game has ended. The class has methods to get the board, get the current player, switch the current player, move a piece on the board, get the number of pieces captured by each player, check if the game has ended, get the winner, and save and load the game.

Constructor & Destructor Documentation

game.Game.Game ()

Constructs a new Game object with a new CheckerBoard, the current player set to "RED", and the number of pieces captured by each player set to 0. The endGame field is set to false.

Member Function Documentation

int game.Game.getBlackCaptured ()

Returns the number of black pieces captured.

Returns

the number of black pieces captured

CheckerBoard game.Game.getBoard ()

Returns the CheckerBoard object representing the board of the game.

Returns

the CheckerBoard object

String game.Game.getCurrentPlayer ()

Returns the current player.

Returns

the current player

static boolean game.Game.getEndGame () `[static]`

Returns the value of the endGame field.

Returns

the value of the endGame field

int game.Game.getRedCaptured ()

Returns the number of red pieces captured.

Returns

the number of red pieces captured

String game.Game.getWinner ()

Returns the winner of the game. The winner is the player with all pieces captured or the other player if the current player has no possible moves.

Returns

the winner of the game

boolean game.Game.isEndGame ()

Checks if the game has ended. The game has ended if all pieces of one color are taken or if the current player has no possible moves.

Returns

true if the game has ended, false otherwise

static Game game.Game.loadGame () `[static]`

Loads a game from a JSON file.

Returns

the loaded Game object

boolean game.Game.move (int *startX*, int *startY*, int *endX*, int *endY*)

Moves a piece on the board from the start position to the end position. If the move is valid, the piece is moved, and the current player is switched. If a piece is captured, the number of pieces captured by the corresponding player is incremented. If the game has ended, the endGame field is set to true.

Parameters

startX	the x-coordinate of the start position
startY	the y-coordinate of the start position
endX	the x-coordinate of the end position
endY	the y-coordinate of the end position

Returns

true if the move is valid and the piece is moved, false otherwise

void game.Game.saveGame ()

Saves the game to a JSON file.

void game.Game.switchPlayer ()

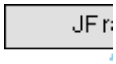
Switches the current player.

The documentation for this class was generated from the following file:

- src/main/java/game/**Game.java**

game.GameFrame Class Reference

Inheritance diagram for game.GameFrame:



Public Member Functions

- **GameFrame** ()
- void **updateLabels** ()

Static Public Member Functions

- static void **main** (String[] args)

Detailed Description

The GameFrame class represents the main frame of the game. It extends the JFrame class and contains a Game object to represent the game, a CheckerBoardPanel object to display the checkerboard, and JLabels to display information about the game.

Constructor & Destructor Documentation

game.GameFrame.GameFrame ()

Constructs a new GameFrame object with a new Game object and a new CheckerBoardPanel object. The frame is set up with a title, size, default close operation, layout, and menu bar.

Member Function Documentation

static void game.GameFrame.main (String[] *args*) `[static]`

Creates a new GameFrame object and sets it to be visible.

Parameters

args	the command-line arguments

void game.GameFrame.updateLabels ()

Updates the labels to display the current player and the number of captured pieces for each player. If the game is over, a message dialog is displayed with the winner.

The documentation for this class was generated from the following file:

- src/main/java/game/**GameFrame.java**