Muhammad Eman Aftab IJE4R1 ije4r1@inf.elte.hu Group 8	**1st assignment/7. Task**	12th October 2024

Task

Break-through is a two-player game, played on a board consists of n x n fields. Each player has 2n dolls in two rows, placed on at the player's side initially (similarly to the chess game, but now every dolls of a player look like the same). A player can move his doll one step forward or one step diagonally forward (can't step backward). A player can beat a doll of his opponent by stepping diagonally forward onto it. A player wins when his doll reaches the opposite edge of the board.

Implement this game, and let the board size be selectable (6x6, 8x8, 10x10). The game should recognize if it is ended, and it has to show in a message box which player won. After this, a new game should be started.

Observation:

The task is to implement a two-player board game called *Break-through*. The game is about:

1. **Game Setup**:
   - The game is played on a square board, which can be 6x6, 8x8, or 10x10.
   - Each player has 2 rows of dolls (like chess pawns). Player 1 starts on one side of the board, and Player 2 on the opposite side.

2. **Game Rules**:
   - Dolls can move one step forward or diagonally forward.
   - Dolls can capture the opponent's doll by stepping diagonally onto it.
   - A doll cannot move backward.

3. **Winning Condition**:
   - A player wins if any of their dolls reach the opposite edge of the board.

4. **Additional Requirements**:
   - The board size should be selectable before the game starts (6x6, 8x8, or 10x10).
   - When a player wins, the game should show a message indicating the winner and user can restart.

Analysis

To complete the task, I need to break it down into smaller steps. First, I need to create a game board that can be different sizes: 6x6, 8x8, or 10x10. The board should be set up with two rows of dolls for each player on their starting sides. Players need to be able to pick a doll and move it according to the rules, either forward or diagonally forward. The moves must be checked to make sure they follow the rules, and players should also be able to capture their opponent's dolls by moving diagonally onto them. The game should take turns between the two players.

I also need to make sure the game can detect when a player wins, which happens when one of their dolls reaches the other side of the board. When this happens, the game should show a message announcing the winner and restart. The game needs a simple, clear interface where players can see the board. There should be buttons to pick the board size at the start and restart the game after it ends. The interface should also show whose turn it is and let players know if a move is invalid or if someone has won.

Solution Plan:

Description about Classes and Objects :

1. Board Class:
    - Represents the game board.
    - Manages the placement of dolls and handles movement validations and executions .
    - Includes methods to initialize the board, validate moves, execute moves, and check for win conditions.
2. Doll Class:
    - Represents a small game piece .
    - Stores properties like position and whether it's captured.
    - Defines basic methods for movement and capturing that can be overridden.
3. Pawn Class:
    - A specific type of Doll.
    - Implements rules for movement (forward or diagonal) and capturing (diagonal only).
4. Position Class:
    - Stores the row and column of a doll on the board.
    - Provides methods to check equality and retrieve position details.
5. BreakthroughGameGUI Class:
    - Handles the graphical user interface.
    - Displays the board, tracks game status, and manages interactions like moves and turns.
    - Shows game rules, player names, and allows board size selection.
6. CellClickListener Class:
    - Handles player clicks on the board cells.
    - Manages doll selection, move validation, and turn toggling.
    - Updates the GUI after every action.
7. GameMenuBar Class:
    - Manages the game menu for selecting board size and restarting the game.
    - Provides options to change difficulty or restart the current game.

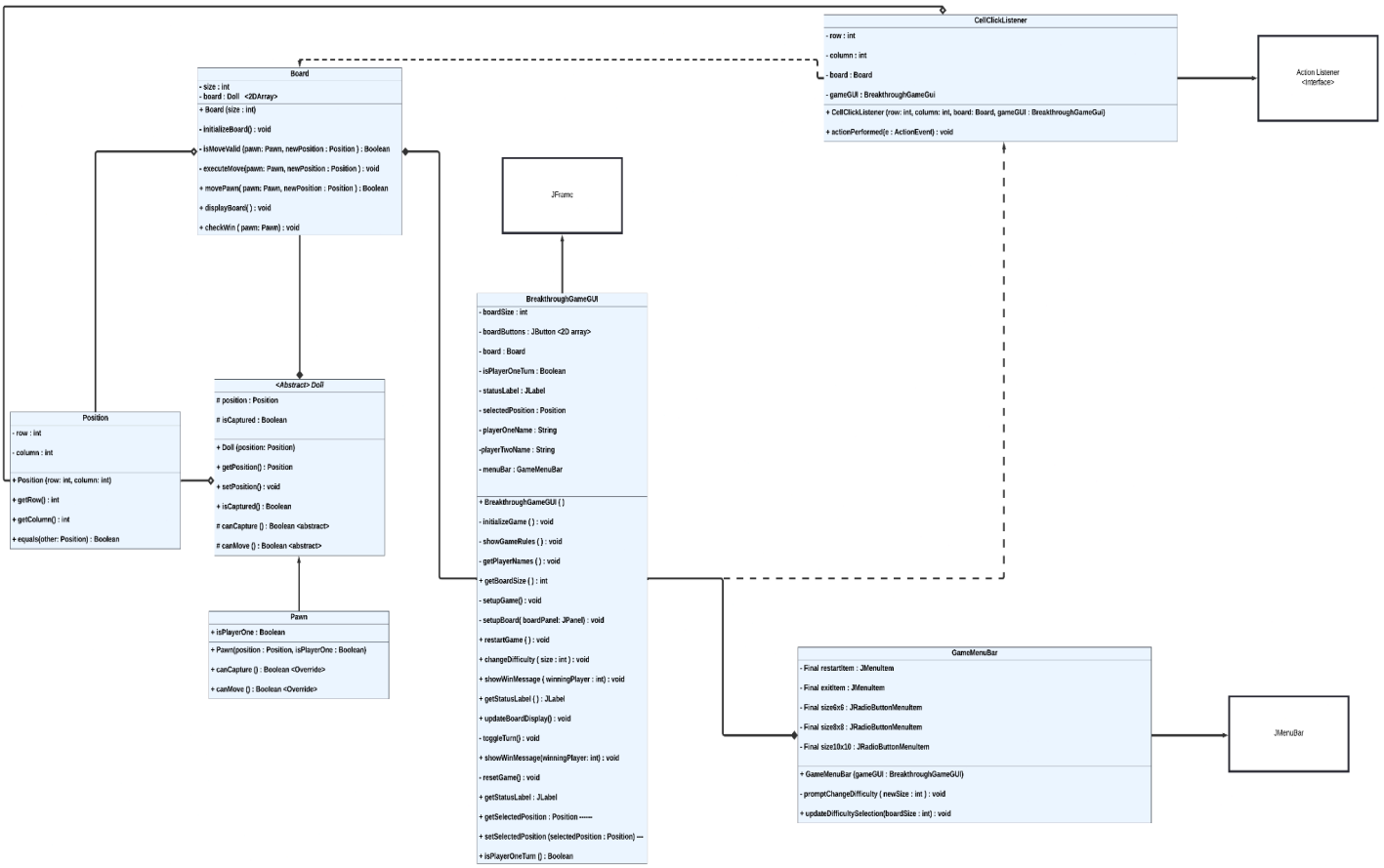
Game Flow :

1. Initialization:
    - Show game rules and gather player names.
    - Allow players to enter their names.
    - Allow players to select the board size (6x6, 8x8, or 10x10).
    - Place two rows of dolls for each player at their respective sides of the board.
2. Player Turns:
    - Players take turns selecting a doll and its destination.
    - Validate moves:
        - Forward or diagonal forward for movement.
        - Diagonal forward for capturing opponent dolls.
    - Execute valid moves and update the board display.
3. Game Progression:
    - After every move, check for win conditions (a doll reaching the opposite edge).
    - If a player wins, show a message box with the winner's name.
4. Win and Restart:
    - When a player wins:
        - Display the winner in a message box.
        - Ask if players want to play again.
    - If yes, restart the game with the same board size.

GUI Implementation :

1. Board Display:
    - Use Java Swing to display a grid representing the board.
    - Each cell is a button, color-coded to represent player 1 (red) or player 2 (blue).
    - Empty cells are gray.
2. Interactions:
    - Players interact by clicking on the cells:
        - First click selects a doll.
        - Second click selects a destination.
    - Show messages for invalid moves and provide feedback on player actions.
3. Game Status:
    - Display a status label showing whose turn it is.
    - Update the status after every valid move.

UML Diagram :


CellClickListener
- row : int
- column : int
- board : Board
- gameGUI : BreakthroughGameGui
+ CellClickListener (row: int, column: int, board: Board, gameGUI : BreakthroughGameGui)
+ actionPerformed(e : ActionEvent) : void
Action Listener
<Interface>
Board
- size : int
- board : Doll <2DArray>
+ Board (size : int)
- initializeBoard() : void
- isMoveValid (pawn: Pawn, newPosition : Position) : Boolean
- executeMove(pawn: Pawn, newPosition : Position) : void
+ movePawn(pawn: Pawn, newPosition : Position) : Boolean
+ displayBoard( ) : void
+ checkWin (pawn: Pawn) : void
JFrame
BreakthroughGameGUI
- boardSize : int
- boardButtons : JButton <2D array>
- board : Board
- isPlayerOneTurn : Boolean
- statusLabel : JLabel
- selectedPosition : Position
- playerOneName : String
-playerTwoName : String
- menuBar : GameMenuBar
+ BreakthroughGameGUI ( )
- initializeGame ( ) : void
- showGameRules ( ) : void
- getPlayerNames ( ) : void
+ getBoardSize ( ) : int
- setupGame() : void
- setupBoard(boardPanel: JPanel) : void
+ restartGame ( ) : void
+ changeDifficulty (size : int) : void
+ showWinMessage (winningPlayer : int) : void
+ getStatusLabel ( ) : JLabel
+ updateBoardDisplay() : void
- toggleTurn() : void
+ showWinMessage(winningPlayer: int) : void
- resetGame() : void
+ getStatusLabel : JLabel
+ getSelectedPosition : Position ------
+ setSelectedPosition (selectedPosition : Position) ---
+ isPlayerOneTurn () : Boolean
<Abstract> Doll
position : Position
isCaptured : Boolean
+ Doll (position: Position)
+ getPosition() : Position
+ setPosition() : void
+ isCaptured() : Boolean
# canCapture () : Boolean <abstract>
# canMove () : Boolean <abstract>
Position
- row : int
- column : int
+ Position (row: int, column: int)
+ getRow() : int
+ getColumn() : int
+ equals(other: Position) : Boolean
Pawn
+ isPlayerOne : Boolean
+ Pawn(position : Position, isPlayerOne : Boolean)
+ canCapture () : Boolean <Override>
+ canMove () : Boolean <Override>
GameMenuBar
- Final restartItem : JMenuItem
- Final exitItem : JMenuItem
- Final size6x6 : JRadioButtonMenuItem
- Final size8x8 : JRadioButtonMenuItem
- Final size10x10 : JRadioButtonMenuItem
+ GameMenuBar (gameGUI : BreakthroughGameGUI)
- promptChangeDifficulty (newSize : int) : void
+ updateDifficultySelection(boardSize : int) : void
JMenuBar

Testing

White-Box:

- Invalid Move on Empty Cell
    - AS A developer
    - I WANT TO prevent actions on empty cells
    - GIVEN an empty cell is selected
    - WHEN a player attempts to move to or from an empty cell
    - THEN the move is rejected, and the game state remains unchanged.

- Valid Diagonal Move Without Capture
    - AS A developer
    - I WANT TO allow diagonal moves to empty cells
    - GIVEN a pawn is selected
    - WHEN it moves diagonally to an empty cell
    - THEN the move is accepted, and the pawn updates its position.

- Valid Forward Move
    - AS A developer
    - I WANT TO verify that forward moves are allowed
    - GIVEN a pawn is selected
    - WHEN it moves one step forward to an empty cell
    - THEN the move is executed successfully, and the board updates.

- Backward Move Restriction
    - AS A developer
    - I WANT TO prevent backward moves
    - GIVEN a pawn is selected
    - WHEN an attempt is made to move backward
    - THEN the move is rejected, and the pawn remains in place.

- Valid Diagonal Capture
    - AS A developer
    - I WANT TO allow pawns to capture opponent pieces diagonally
    - GIVEN a pawn and an opponent's pawn are in diagonal positions
    - WHEN the pawn moves onto the opponent's position
    - THEN the move is accepted, and the opponent's piece is removed.

- Out-of-Bounds Move Check
    - AS A developer
    - I WANT TO restrict moves outside the board boundaries
    - GIVEN a pawn is selected
    - WHEN an attempt is made to move off the board
    - THEN the move is rejected, and the game state remains unchanged.

- Win Condition Recognition
    - AS A developer
    - I WANT TO detect when a player wins
    - GIVEN a pawn is one step away from the opposite edge
    - WHEN it moves to the opponent's edge
    - THEN the game recognizes the win and announces the winner.

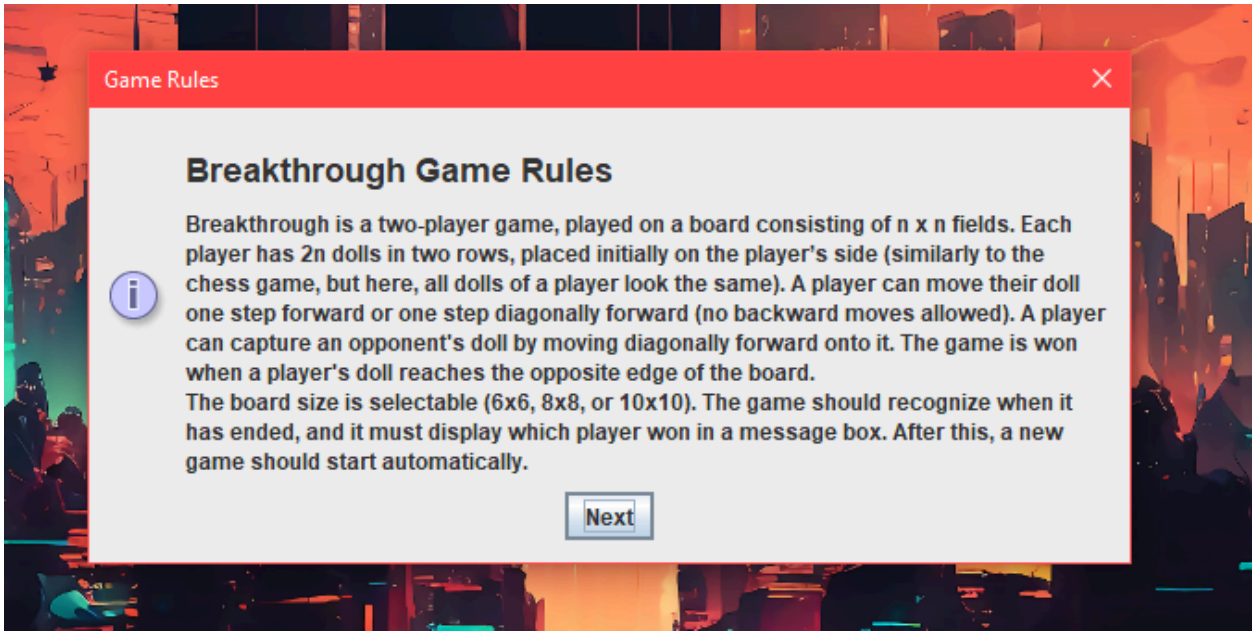
- Turn Alternation Logic
    - AS A developer
    - I WANT TO ensure turns alternate between players
    - GIVEN Player 1 completes their turn
    - WHEN the game logic updates
    - THEN it switches to Player 2's turn and vice versa.

- New Game Restart
    - AS A developer
    - I WANT TO restart the game after it ends
    - GIVEN the game has ended with a winner
    - WHEN the restart logic is triggered or the restart button is pressed.
    - THEN the board resets, and a new game begins by starting from game rules.

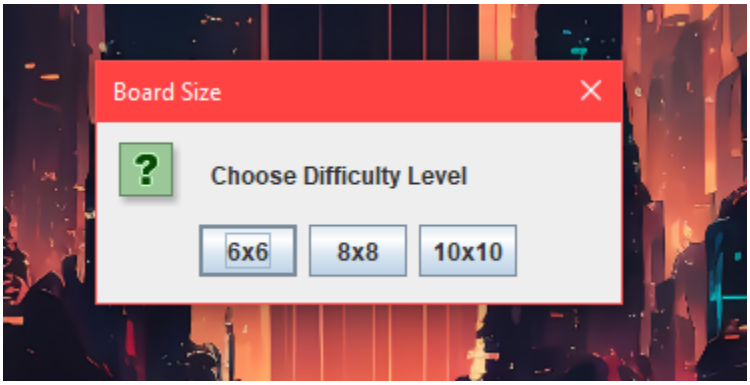
- Board Size Selection
    - AS A developer
    - I WANT TO allow board size selection before starting a game
    - GIVEN the player selects a board size (6x6, 8x8, or 10x10)
    - WHEN the game initializes
    - THEN the board is created with the chosen size, and the game starts.

Black-Box:

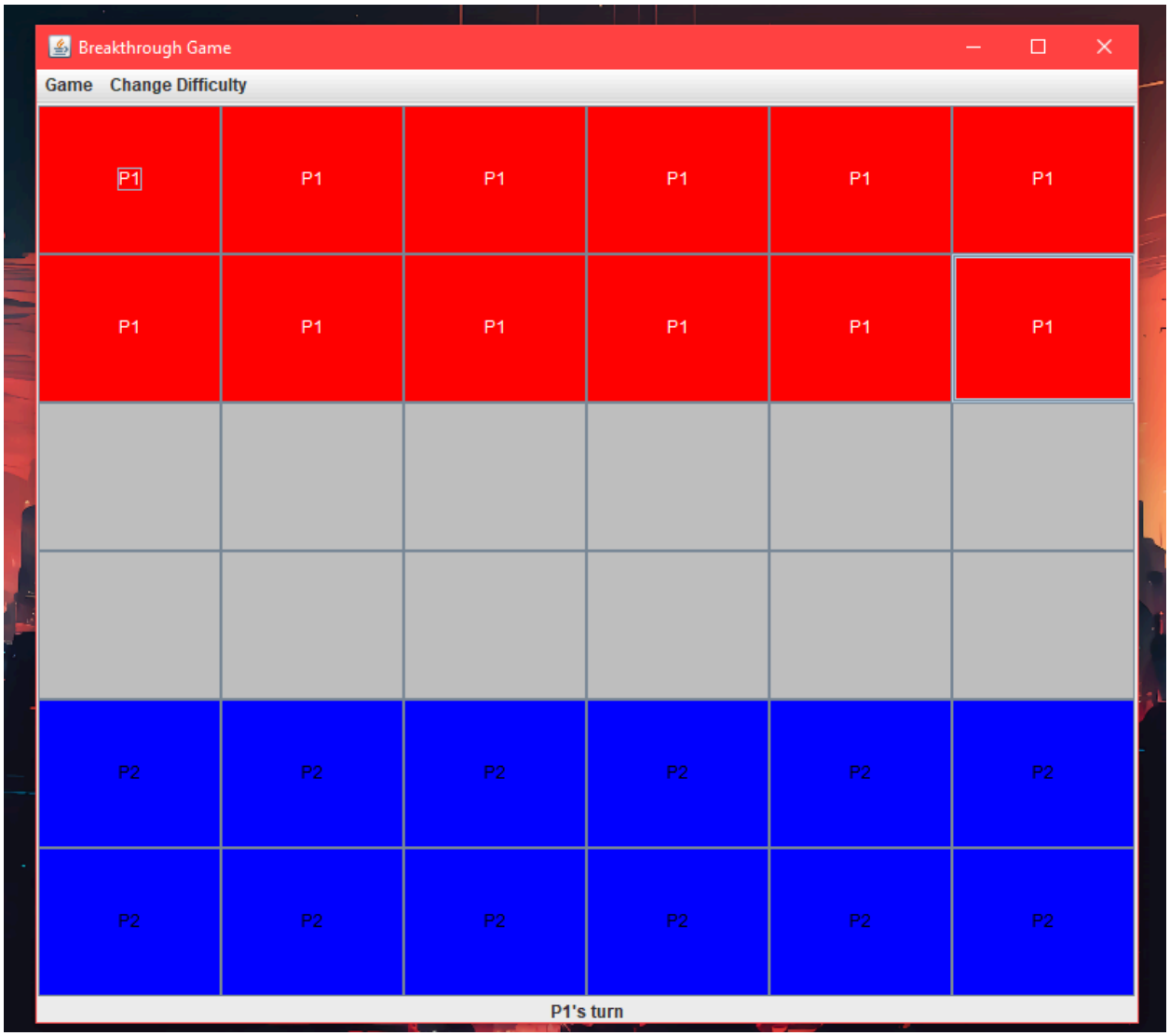
- Rules and Initial Setup
    1. Game Rules Display
        - AS A new user
        - I WANT TO understand the rules of the game
        - GIVEN the game has just started
        - WHEN the rules dialog is shown
        - THEN I can read the rules and proceed to the game.



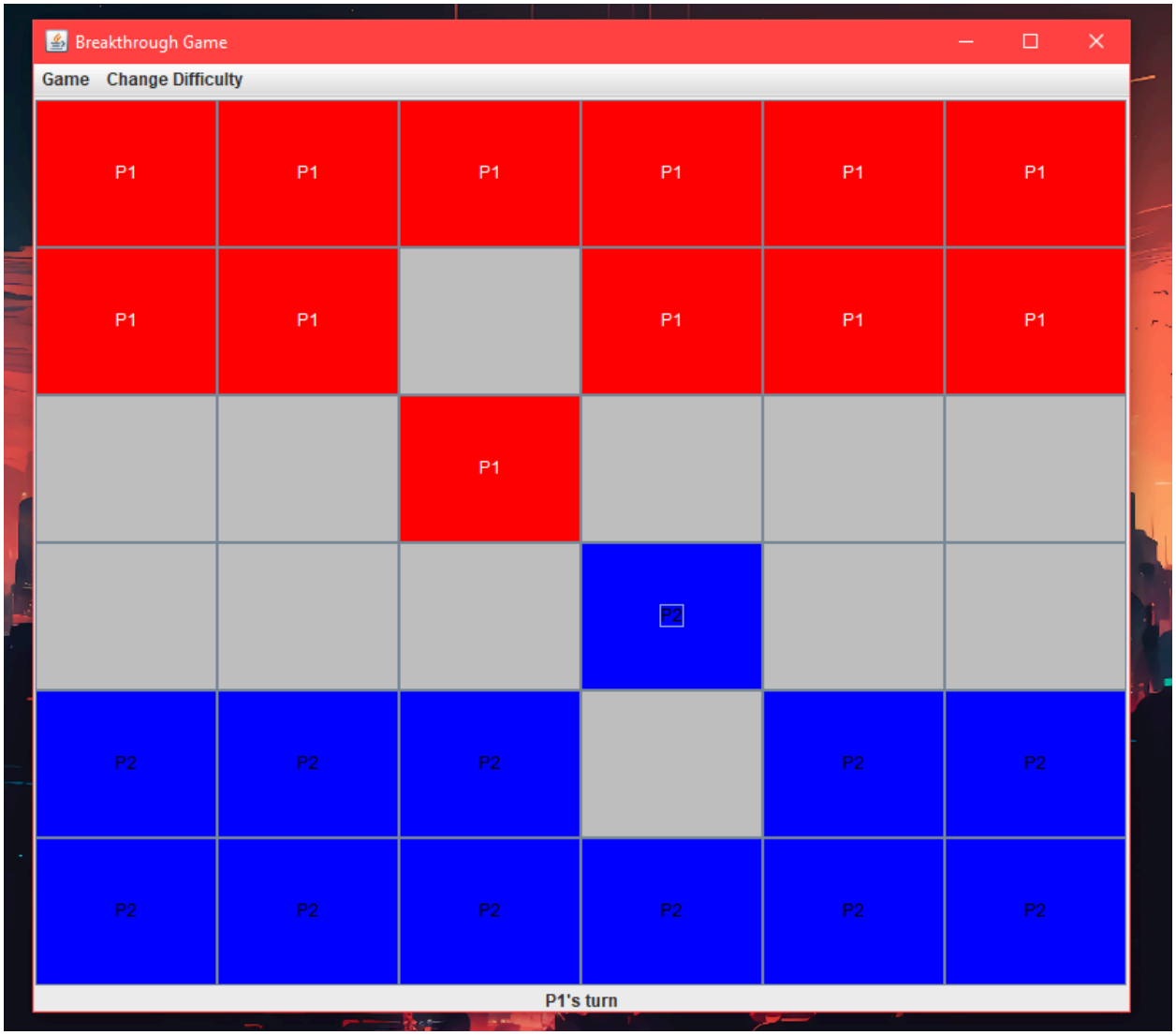
2. Board Size Selection
    - AS A user
    - I WANT TO select the board size
    - GIVEN a menu with options for 6x6, 8x8, and 10x10
    - WHEN I select a size
    - THEN the board is displayed with the correct dimensions.



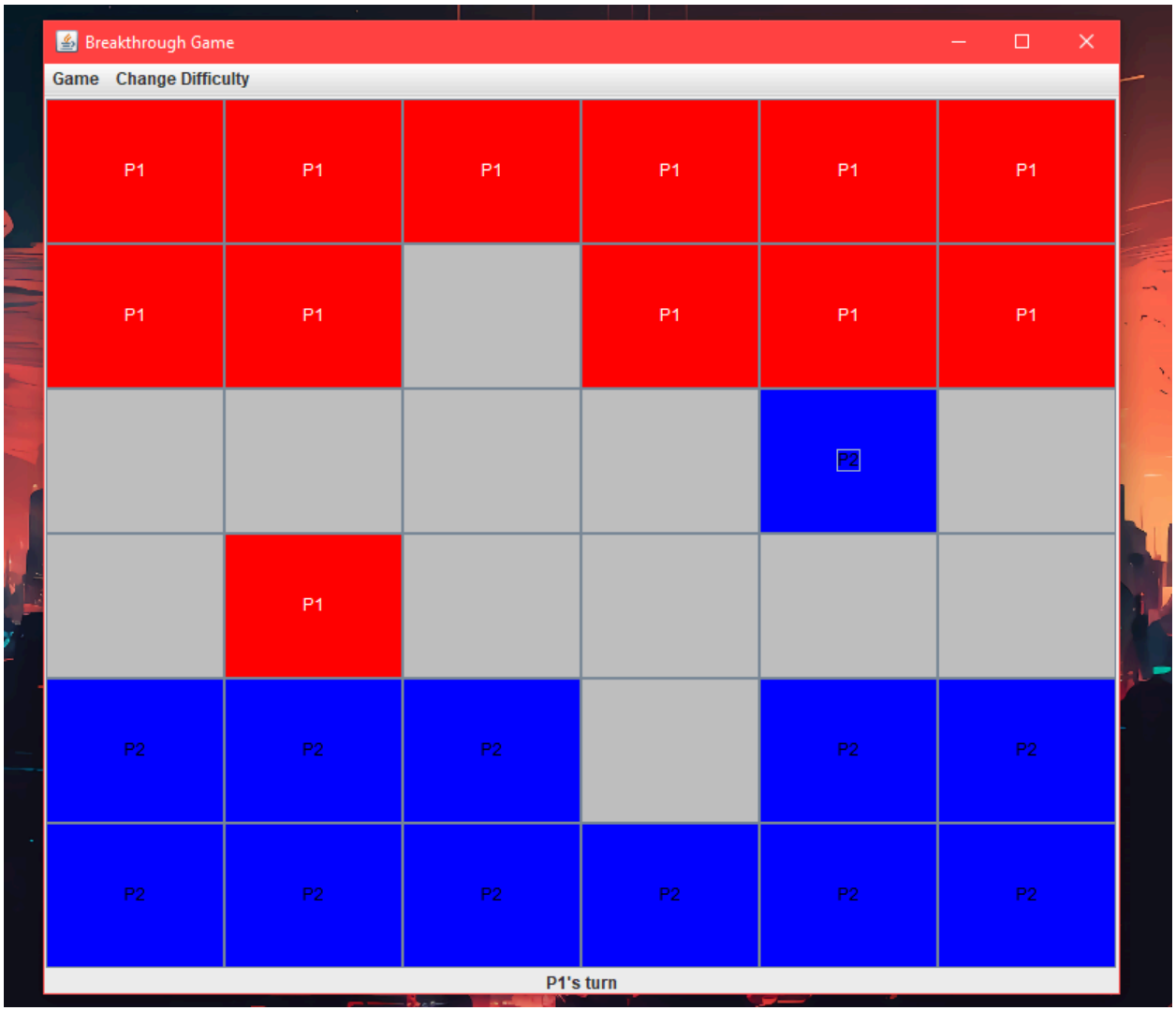
3. Initial Board Setup
    - AS A player
    - I WANT TO see the initial setup of the board
    - GIVEN the game starts after the board size is selected
    - WHEN the board is displayed
    - THEN each player's dolls are placed correctly on their respective sides.



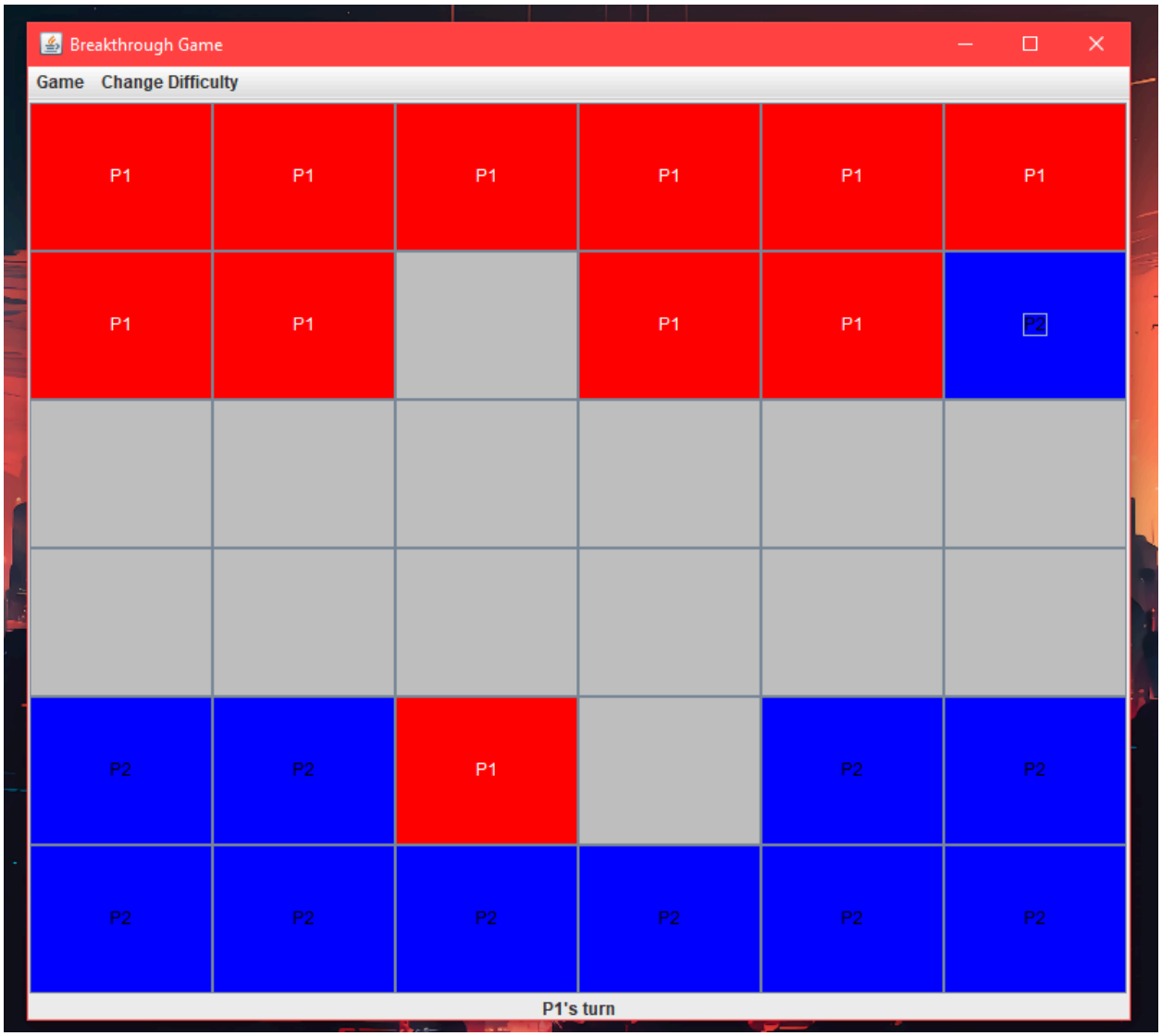
- Player Actions
    4. Valid Forward Move
        - AS A player
        - I WANT TO move my pawn one step forward
        - GIVEN it is my turn, and I select my pawn
        - WHEN I click a valid forward cell
        - THEN the pawn moves to the selected cell, and the board updates.



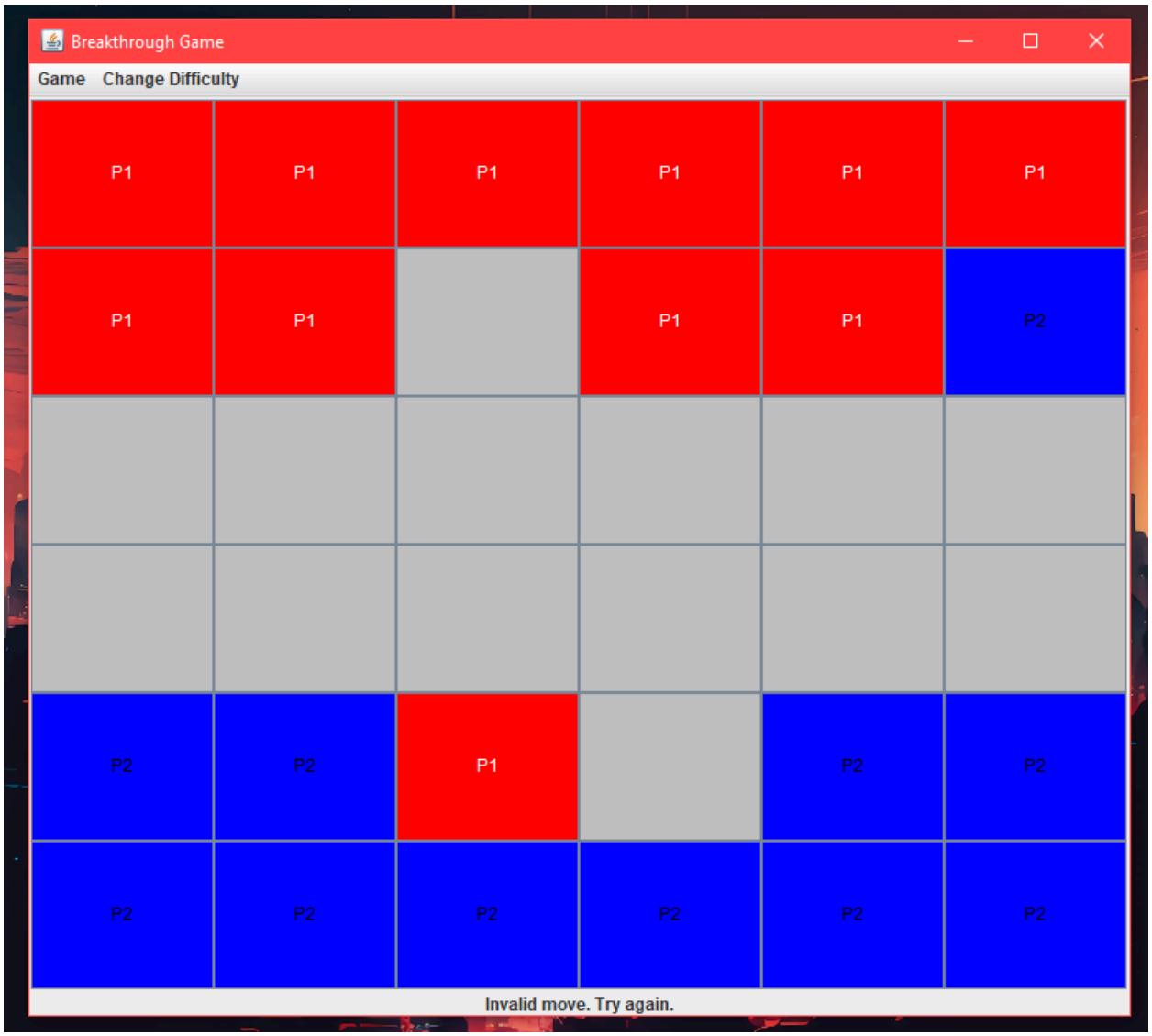
5. Valid Diagonal Move
    - AS A player
    - I WANT TO move my pawn diagonally forward
    - GIVEN it is my turn, and I select my pawn
    - WHEN I click a valid diagonal cell
    - THEN the pawn moves to the selected cell, and the board updates.



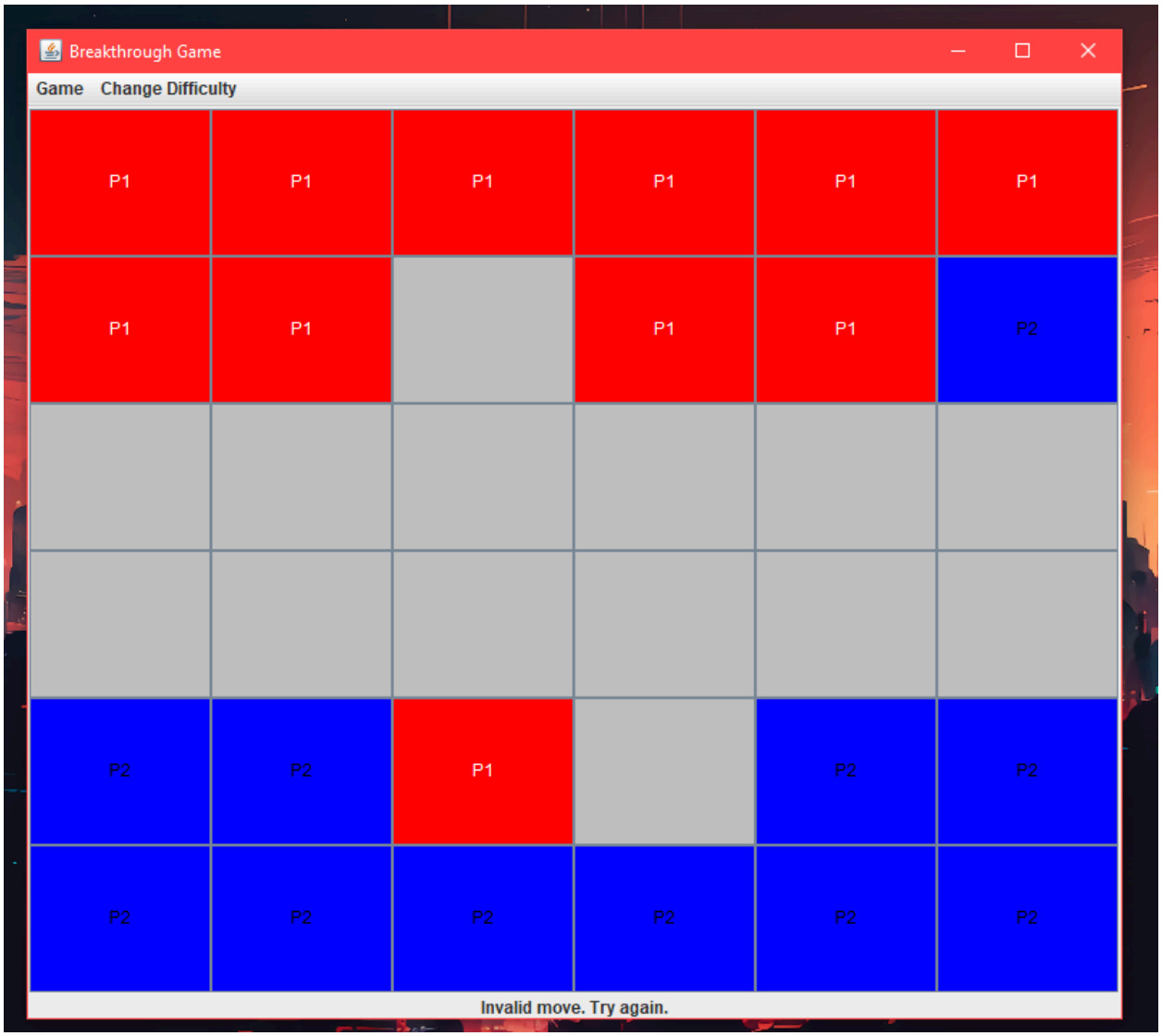
6. Valid Capture Move
    - AS A player
    - I WANT TO capture my opponent's doll
    - GIVEN my pawn is adjacent diagonally to an opponent's doll
    - WHEN I move my pawn onto the opponent's cell
    - THEN the opponent's doll is removed, and my pawn occupies the cell.



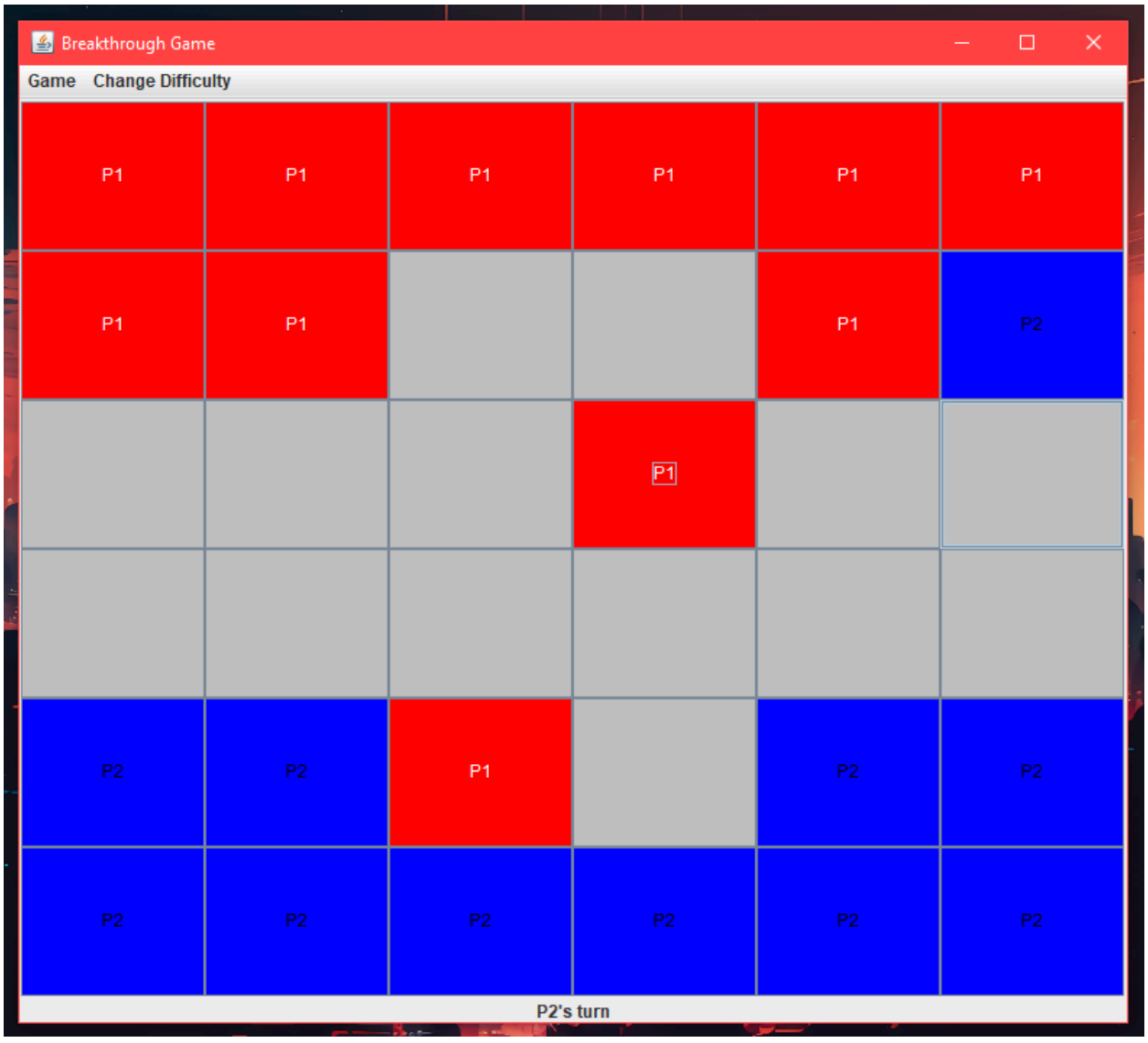
7. Invalid Move
    - AS A player
    - I WANT TO know if I attempt an invalid move
    - GIVEN it is my turn, and I select a pawn
    - WHEN I click an invalid cell (e.g., backward or non-adjacent)
    - THEN the game displays an error message like "Invalid move."

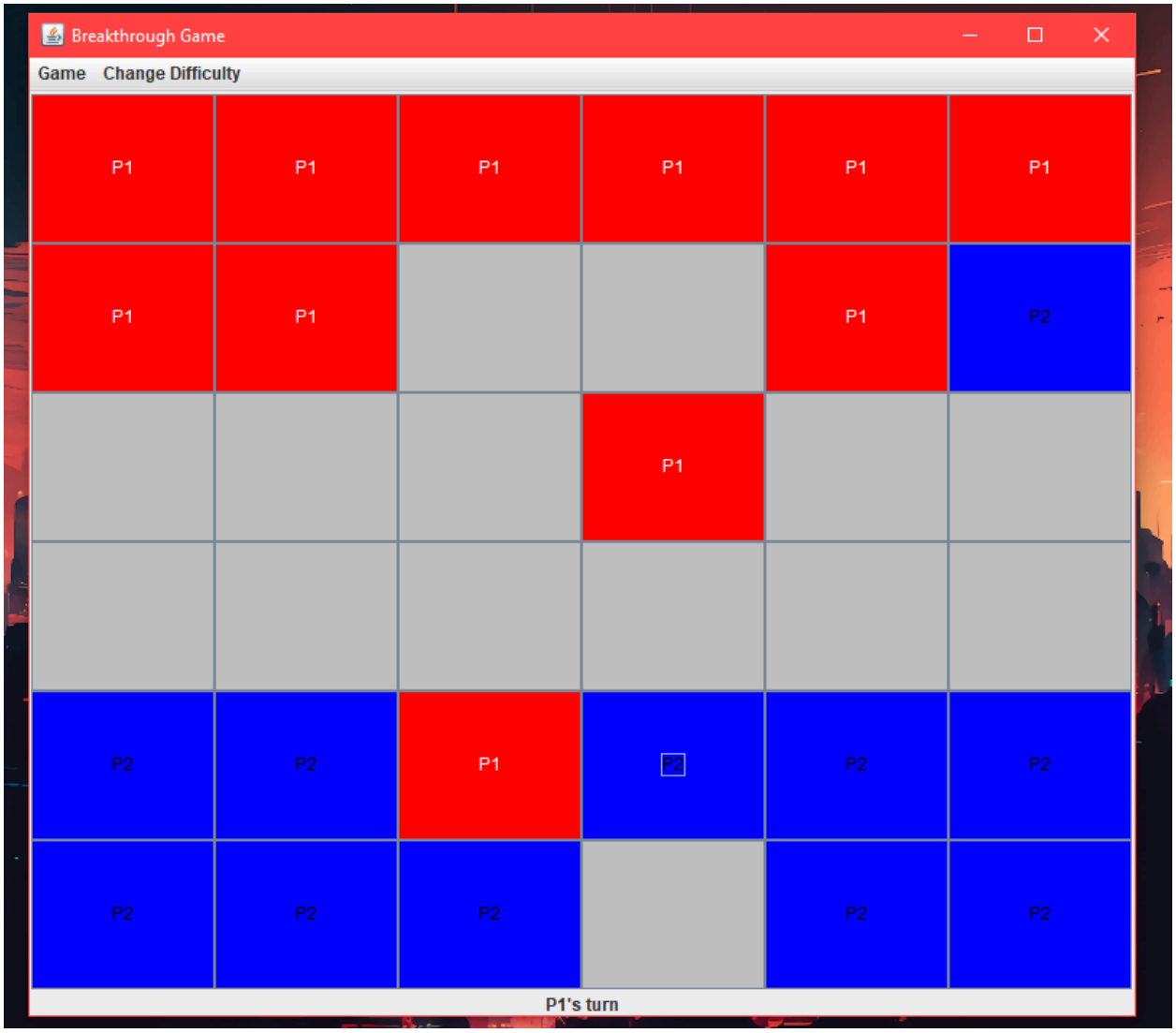


8. Empty Cell Click
    - AS A player
    - I WANT TO prevent actions on empty cells
    - GIVEN I click an empty cell
    - WHEN I try to perform an action
    - THEN nothing happens, and the game state remains unchanged.



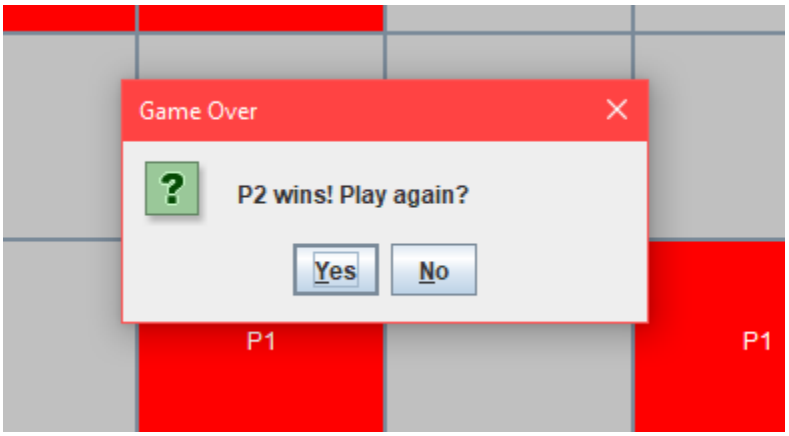
- Game Flow
    9. Turn Alternation
        - AS A player
        - I WANT TO see the turn switch to my opponent after my move
        - GIVEN I complete my turn
        - WHEN I make a valid move
        - THEN the game shows that it is now my opponent's turn.




Breakthrough Game
Game
Change Difficulty
P1
P1
P1
P1
P1
P1
P1
P1
P1
P2
P1
P2
P2
P1
P2
P2
P2
P2
P2
P2
P2
P2
P1's turn

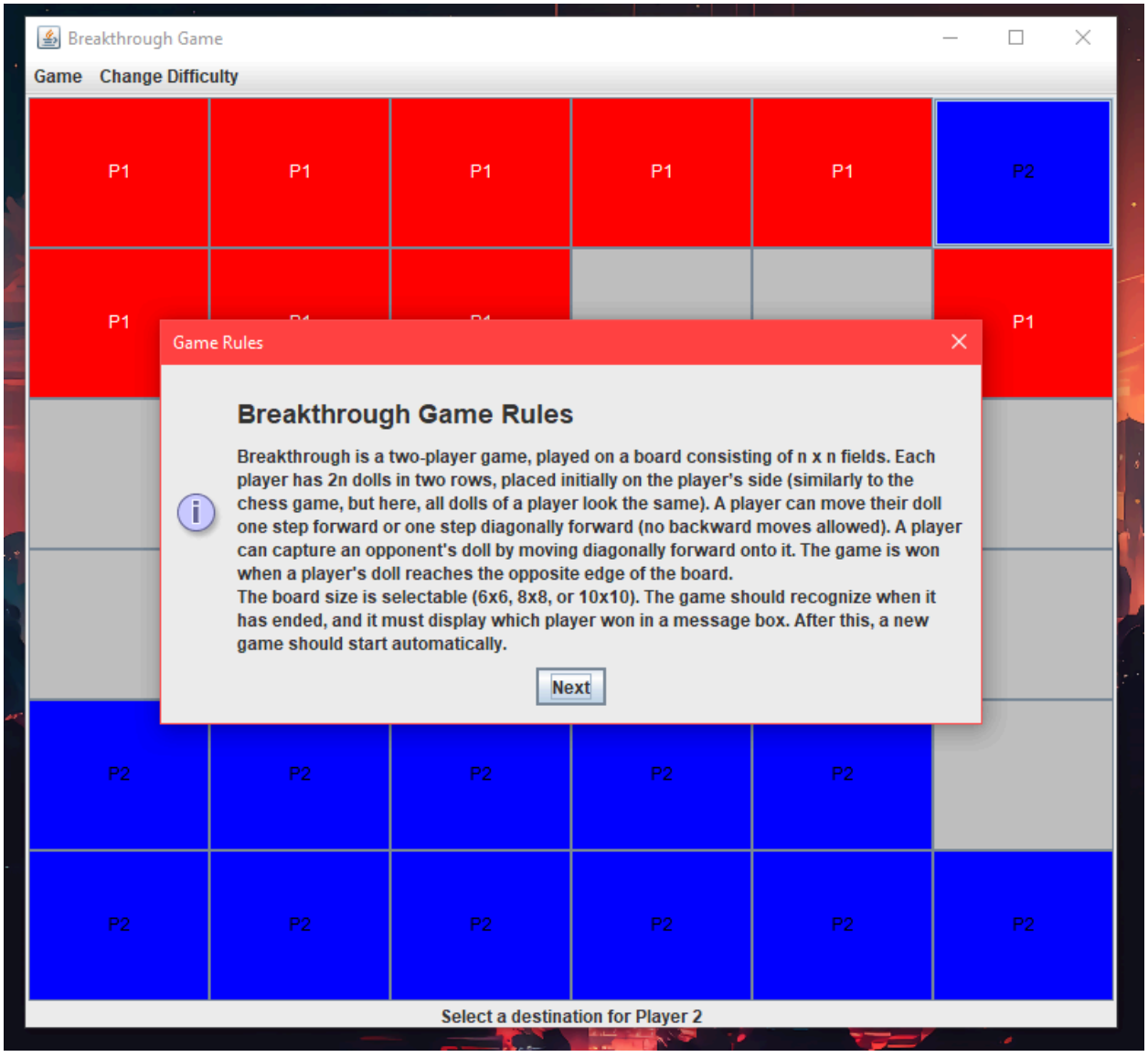
10. Win Condition
    - AS A player
    - I WANT TO win when my pawn reaches the opposite edge
    - GIVEN my pawn is one move away from the last row
    - WHEN I move my pawn to the last row
    - THEN the game announces me as the winner and prompts to restart.

- Post-Game
    11. Restart Game
        - AS A user
        - I WANT TO restart the game after it ends
        - GIVEN the game is over with a winner declared
        - WHEN I confirm to restart the game
        - THEN the board resets, and a new game begins moving users back to the rules section.




P1
P1
P1
P2
P2
P2
P2
P2
Game Rules
Breakthrough Game Rules
Breakthrough is a two-player game, played on a board consisting of n x n fields. Each player has 2n dolls in two rows, placed initially on the player's side (similarly to the chess game, but here, all dolls of a player look the same). A player can move their doll one step forward or one step diagonally forward (no backward moves allowed). A player can capture an opponent's doll by moving diagonally forward onto it. The game is won when a player's doll reaches the opposite edge of the board.
The board size is selectable (6x6, 8x8, or 10x10). The game should recognize when it has ended, and it must display which player won in a message box. After this, a new game should start automatically.
Next

12. Exit Game
    - AS A user
    - I WANT TO exit the game anytime
    - GIVEN the game is running
    - WHEN I close the game window
    - THEN the game shuts down cleanly without errors.



Button Handlers:

1. CellClickListener
    - Event: `ActionEvent` triggered when a game board cell is clicked.
    - Handler: `CellClickListener.actionPerformed(ActionEvent e)`
    - Connection: Determines the player's action:
        - If no pawn is selected, it selects the pawn at the clicked cell if valid.
        - If a pawn is already selected, it attempts to move the pawn to the clicked cell.
        - Updates the game state (e.g., board display, player turn) and checks for the win condition.

Menu Item Handlers:

2. Restart Menu Item
    - Event: `ActionEvent` triggered when the "Restart" menu item is selected.
    - Handler: `GameMenuBar.restartItem.actionPerformed(ActionEvent e)`
    - Connection: Calls `BreakthroughGameGUI.restartGame()` to reset the current game board and start fresh.
3. Exit Menu Item
    - Event: `ActionEvent` triggered when the "Exit" menu item is selected.
    - Handler: `GameMenuBar.exitItem.actionPerformed(ActionEvent e)`
    - Connection: Exits the application by calling `System.exit(0)`.
4. Difficulty Change Menu Items (6x6, 8x8, 10x10)
    - Event: `ActionEvent` triggered when a difficulty option is selected.
    - Handler: Each menu item (e.g., `GameMenuBar.size6x6.actionPerformed(ActionEvent e)`)
    - Connection: Prompts the user to confirm the board size change. On confirmation, calls `BreakthroughGameGUI.changeDifficulty(int size)` to initialize the game with the new board size.