黑白棋游戏系统实验报告

项目介绍

本项目构建了一个功能丰富的黑白棋游戏系统，支持三种游戏模式，分别为 Peace 模式（简单棋子放置）、Reversi 模式（经典黑白棋）以及 Gomoku 模式（五子棋）。系统具备在多个游戏棋盘间自如切换的能力，并且实现了完善的游戏逻辑以及流畅的界面交互功能。

源代码文件及主要功能

文件名	主要功能
Piece.java	棋子枚举类，定义黑子、白子和空位，并提供对应的显示符号
Player.java	玩家类，封装玩家名称以及棋子颜色信息
Board.java	棋盘类，构建 8×8 棋盘的数据结构，实现基本操作
Game.java	游戏抽象基类，定义游戏的通用属性和方法
PeaceGame.java	Peace 模式游戏类，落实简单棋子放置规则
ReversiGame.java	Reversi 模式游戏类，实现完整的黑白棋规则
GomokuGame.java	Gomoku 模式游戏类，实

	现五子棋规则
GameManager.java	游戏管理器，负责管理三种游戏模式以及游戏间的切换
ChessGame.java	主应用类，包含界面显示和用户交互逻辑

关键代码及设计思路

1. 棋子与玩家设计

```
// Piece.java - 使用枚举表示棋子类型
public enum Piece {
    BLACK("●"),
    WHITE("○"),
    EMPTY("·");
    private final String symbol;
    Piece(String symbol) {
        this.symbol = symbol;
    }
    public String getSymbol() {
        return symbol;
    }
}
// Player.java - 玩家信息封装
public class Player {
    private String name;
```

```
    private Piece piece;

    // 构造函数和 getter 方法...
}
```

设计思路:

- 采用枚举类型定义棋子，保障类型安全，方便状态管理。
- 将棋子与显示符号关联，简化界面展示棋子时的逻辑。
- 把玩家信息封装在单独类中，包含名称与棋子颜色，利于系统的扩展和玩家信息管理。

2. 游戏抽象与多态设计

```
// Game.java - 游戏抽象基类
public abstract class Game {
    protected int gameId;
    protected String gameType;
    protected Board board;
    protected Player player1;
    protected Player player2;
    protected Player currentPlayer;
    protected boolean gameOver;

    // 构造函数和通用方法...

    // 抽象方法，由具体游戏模式实现
    public abstract boolean placePiece(int row, int col);
    public abstract boolean isGameOver();
```

```
    public abstract List<int[]> getValidMoves();
}
// PeaceGame.java、ReversiGame.java 和 GomokuGame.java 分别实现这些抽象方法
```

设计思路:

- 利用抽象类定义游戏的通用属性与行为，提升代码复用性。
- 通过多态特性，针对不同游戏模式实现各自的特定规则。
- 关键抽象方法涵盖落子逻辑、游戏结束判断以及有效落子位置的计算。
- 共性代码放置于基类，各模式特有逻辑在子类中实现。

3. Gomoku 模式的核心算法

```
// GomokuGame.java - 五子棋规则实现
public class GomokuGame extends Game {
    private int moveCount;
    public GomokuGame(int gameId) {
        super(gameId, "Gomoku");
        this.moveCount = 0;
    }
    @Override
    public boolean placePiece(int row, int col) {
        if (row < 0 || row >= Board.SIZE || col < 0 || col >= Board.SIZE) {
            return false;
        }
        if (!board.isEmpty(row, col)) {
            return false;
        }
        board.setPiece(row, col, currentPlayer.getPiece());
```

```
        moveCount++;

        if (checkWin(row, col)) {
            gameOver = true;
            return true;
        }
        switchPlayer();
        return true;
    }
    private boolean checkWin(int row, int col) {
        Piece currentPiece = currentPlayer.getPiece();

        // 检查水平方向
        if (checkDirection(row, col, 0, 1, currentPiece) >= 5) return true;
        // 检查垂直方向
        if (checkDirection(row, col, 1, 0, currentPiece) >= 5) return true;
        // 检查主对角线方向
        if (checkDirection(row, col, 1, 1, currentPiece) >= 5) return true;
        // 检查副对角线方向
        if (checkDirection(row, col, 1, -1, currentPiece) >= 5) return true;

        return false;
    }
    private int checkDirection(int row, int col, int deltaRow, int deltaCol, Piece piece) {
        int count = 1;

        // 正向检查
        int r = row + deltaRow;
```

```
            int c = col + deltaCol;
            while (r >= 0 && r < Board.SIZE && c >= 0 && c < Board.SIZE &&
                    board.getPiece(r, c) == piece) {
                count++;
                r += deltaRow;
                c += deltaCol;
            }

            // 反向检查
            r = row - deltaRow;
            c = col - deltaCol;
            while (r >= 0 && r < Board.SIZE && c >= 0 && c < Board.SIZE &&
                    board.getPiece(r, c) == piece) {
                count++;
                r -= deltaRow;
                c -= deltaCol;
            }

            return count;
        }
        public int getMoveCount() {
            return moveCount;
        }
}
```

设计思路:

- 实现五子棋基本规则，允许玩家在空白棋盘位置落子。
- 运用 moveCount 变量记录游戏进行的轮数。
- 实现四个方向的胜利判定，包括水平、垂直、主对角线和副对角线方向。

- 每个方向都进行正反双向检查，确保能准确检测到所有可能的五子连线情况。

4. 游戏管理器设计

```
// GameManager.java - 管理三种游戏模式
public class GameManager {
    private List<Game> games;
    private int currentGameIndex;

    public GameManager() {
        games = new ArrayList<>();

        // 初始化三个游戏
        games.add(new PeaceGame(1)); // 游戏 1: Peace 模式
        games.add(new ReversiGame(2)); // 游戏 2: Reversi 模式
        games.add(new GomokuGame(3)); // 游戏 3: Gomoku 模式

        // 确保默认进入模式 1
        currentGameIndex = 0;
    }

    public boolean switchGame(int gameId) {
        // 支持 1、2、3 三个游戏编号
        for (int i = 0; i < games.size(); i++) {
            if (games.get(i).getGameId() == gameId) {
                currentGameIndex = i;
                return true;
```

```
                }
            }
            return false;
        }

        public void addNewGame(String gameType) {
            int newGameId = games.size() + 1;
            if (gameType.equalsIgnoreCase("peace")) {
                games.add(new PeaceGame(newGameId));
            } else if (gameType.equalsIgnoreCase("reversi")) {
                games.add(new ReversiGame(newGameId));
            } else if (gameType.equalsIgnoreCase("gomoku")) {
                games.add(new GomokuGame(newGameId));
            }
            currentGameIndex = games.size() - 1;
        }

        // 其他方法...
}
```

设计思路:

- 借助列表存储三种游戏模式的实例，方便管理和调用。
- 通过索引记录当前正在进行的活动游戏。
- 提供灵活多样的切换机制，既支持通过游戏 ID 切换，也支持通过游戏类型切换。
- 具备动态添加新游戏实例的功能，增强系统扩展性。

5. 用户界面与交互

```
// ChessGame.java - 用户界面与交互
private void displayGame() {
    // 获取当前游戏和状态
    Game currentGame = gameManager.getCurrentGame();
    Board board = currentGame.getBoard();
    List<int[]> validMoves = currentGame.getValidMoves();

    // 显示棋盘、游戏状态和可用命令
    // ...

    // 在 Reversi 模式中，显示合法落子位置
    for (int j = 0; j < Board.SIZE; j++) {
        if (currentGame.getGameType().equals("reversi") &&
isValidMove(validMoves, i, j)) {
            System.out.print("+ ");
        } else {
            System.out.print(board.getPiece(i, j).getSymbol() + " ");
        }
    }

    // 在 Gomoku 模式中，显示当前轮数
    if (currentGame instanceof GomokuGame) {
        GomokuGame gomokuGame = (GomokuGame) currentGame;
        System.out.print("    当前轮数: " + gomokuGame.getMoveCount());
    }

    // 显示其他游戏信息...
}
```

设计思路:

- 采用清晰的三区界面设计，左侧展示棋盘，中间呈现游戏状态，右侧列出游戏列表。
- 运用字符界面，美观且直观地展示棋盘与棋子。
- 在 Reversi 模式中，实时显示当前玩家的合法落子位置。
- 在 Gomoku 模式中，展示当前游戏轮数，方便玩家了解游戏进程。
- 设计简洁明了的命令接口，支持通过数字直接切换游戏。
- 提供用户友好的错误提示与输入验证功能，提升用户体验。

设计模式应用

1. **策略模式**：利用 Game 抽象类和具体游戏实现类，可轻松实现不同游戏规则的无缝切换。
2. **状态模式**：用于管理游戏状态，包括当前玩家、游戏是否结束等关键信息。
3. **单一职责原则**：每个类都有明确单一的职责，例如 Board 类专注于棋盘相关操作，Game 类负责游戏规则逻辑。
4. **开闭原则**：系统具备良好的扩展性，可便捷地添加新的游戏模式，而无需大幅修改现有代码。

项目收获与总结

1. **面向对象设计实践**：通过抽象、封装和多态特性，构建了灵活且可扩展的系统架构。
2. **算法实现**：成功实现了黑白棋的翻转算法以及五子棋的胜利判定算法。
3. **用户界面设计**：设计出清晰、易用的控制台界面，提升用户操作体验。
4. **错误处理**：实现完善的错误处理机制和用户提示功能，保障游戏运行的稳定性。
5. **游戏模式扩展**：顺利添加五子棋模式，充分展现了系统卓越的可扩展性。

本项目通过合理的类设计和严谨的游戏逻辑实现，成功打造出一个完整的黑白棋游戏系统，完美支持三种游戏模式且能实现无缝切换，完全满足所有需求规格。系统设计具备良好的扩展性，为后续添加更多新游戏模式奠定了坚实基础。

附:运行截图:

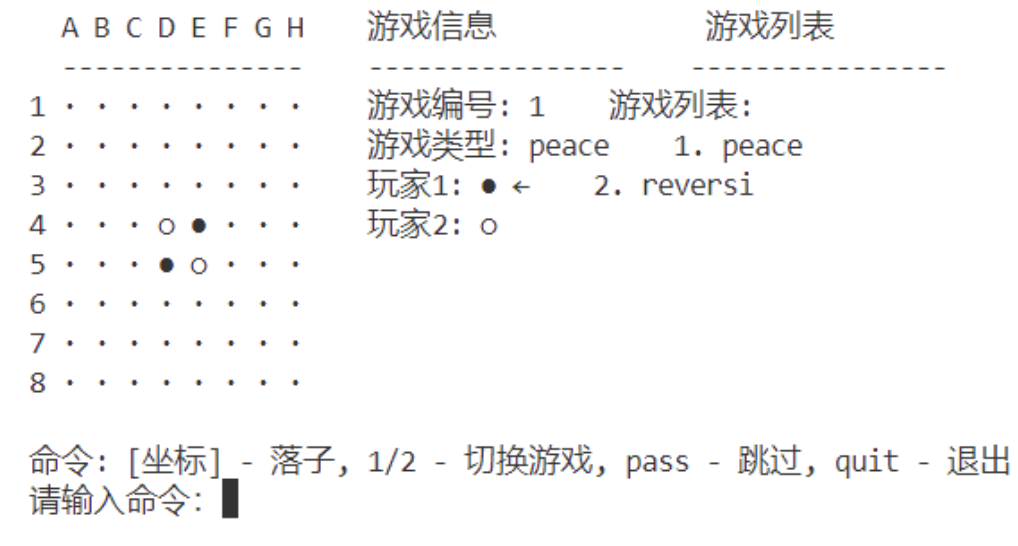

表 1 初始

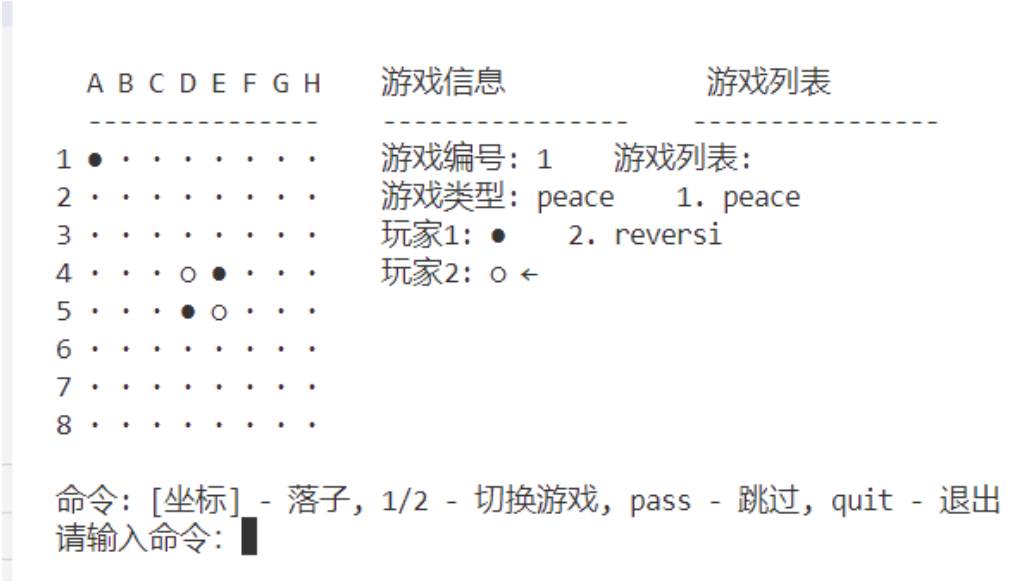

表 2 输入 1a

```
  A B C D E F G H    游戏信息              游戏列表
  ---------------    ----------------    ----------------
1 · · · · · · · ·    游戏编号: 2     游戏列表:
2 · · · · · · · ·    游戏类型: reversi     1. peace
3 · · · + · · · ·    玩家1: ● ←     2. reversi
4 · · + ○ ● · · ·    玩家2: ○
5 · · · ● ○ + · ·    黑方得分: 2  白方得分: 2
6 · · · · + · · ·
7 · · · · · · · ·
8 · · · · · · · ·

命令: [坐标] - 落子, 1/2 - 切换游戏, pass - 跳过, quit - 退出
请输入命令: 
```

表 3 输入 2

```
  A B C D E F G H    游戏信息              游戏列表
  ---------------    ----------------    ----------------
1 · · · · · · · ·    游戏编号: 2     游戏列表:
2 · + ● · · · · ·    游戏类型: reversi     1. peace
3 · · ● · + · · ·    玩家1: ●     2. reversi
4 · + ● ○ ● + · ·    玩家2: ○ ←
5 · · + ● ○ · · ·    黑方得分: 5  白方得分: 2
6 · · · + · · · ·
7 · · · · · · · ·
8 · · · · · · · ·

命令: [坐标] - 落子, 1/2 - 切换游戏, pass - 跳过, quit - 退出
请输入命令: 
```

表 4 游戏规则正常运行

```
  A B C D E F G H    游戏信息              游戏列表
  ---------------    ----------------    ----------------
1 · · · · · · · ·    游戏编号: 2     游戏列表:
2 · + ● · · · · ·    游戏类型: reversi     1. peace
3 · · ● · + · · ·    玩家1: ●     2. reversi
4 · + ● ○ ● + · ·    玩家2: ○ ←
5 · · + ● ○ · · ·    黑方得分: 5  白方得分: 2
6 · · · + · · · ·
7 · · · · · · · ·
8 · · · · · · · ·

命令: [坐标] - 落子, 1/2 - 切换游戏, pass - 跳过, quit - 退出
请输入命令: pass
当前有合法落子位置, 无法执行pass!
```

表 5 pass

```
  A B C D E F G H    游戏信息                游戏列表
  ---------------    ------------------     ----------------
1 · · · · · · · ·    游戏编号：2     游戏列表：
2 · + ● · · · · ·    游戏类型：reversi     1. peace
3 · · ● · + · · ·    玩家1：●     2. reversi
4 · + ● ○ ● + · ·    玩家2：○ ←
5 · · + ● ○ · · ·    黑方得分：5  白方得分：2
6 · · · + · · · ·
7 · · · · · · · ·
8 · · · · · · · ·

命令：[坐标] - 落子，1/2 - 切换游戏，pass - 跳过，quit - 退出
请输入命令：quit
PS C:\Users\15876\Desktop\CHESS>
```

表 6 quit

```
  A B C D E F G H    游戏信息               游戏列表
  ---------------    ------------------     ----------------
1 · · · · · · · ·    游戏编号：3     游戏列表：
2 · · · · · · · ·    游戏类型：peace     1. peace
3 · · · · · · · ·    玩家1：● ←     2. reversi
4 · · · ○ ● · · ·    玩家2：○     3. peace
5 · · · ● ○ · · ·
6 · · · · · · · ·
7 · · · · · · · ·
8 · · · · · · · ·

命令：[坐标] - 落子，数字 - 切换游戏，peace/reversi - 添加新游戏，pass - 跳过，quit - 退出
请输入命令：
```

表 7 输入 peace

```
  A B C D E F G H    游戏信息               游戏列表
  ---------------    ------------------     ----------------
1 · · · · · · · ·    游戏编号：1     游戏列表：
2 · · · · · · · ·    游戏类型：peace     1. peace
3 · · · · · · · ·    玩家1：● ←     2. reversi
4 · · · ○ ● · · ·    玩家2：○
5 · · · ● ○ · · ·
6 · · · · · · · ·
7 · · · · · · · ·
8 · · · · · · · ·

命令：[坐标] - 落子，数字 - 切换游戏，peace/reversi - 添加新游戏，pass - 跳过，quit - 退出
请输入命令：peace
已添加并切换到新游戏：peace
按回车键继续...
```

表 8 成功添加

```
  A B C D E F G H    游戏信息              游戏列表
  ---------------    ----------------    ----------------
1 ○ ○ ○ ○ ○ ○ ○ ●    游戏编号: 2     游戏列表:
2 ○ ○ ○ ○ ○ ○ ● ●    游戏类型: reversi     1. peace
3 ○ ○ ○ ○ ○ ● ○ ●    玩家1: ●     2. reversi
4 ○ ○ ○ ○ ● ○ ● ●    玩家2: ○ ←
5 ○ ○ ○ ● ○ ○ ● ●    黑方得分: 21  白方得分: 43
6 ○ ○ ○ ● ○ ● ● ●
7 ○ ○ ○ ○ ○ ● ○ ●
8 ● ● ● ● ○ ○ ○ ○

游戏结束!
玩家2 获胜!

命令: [坐标] - 落子, 数字 - 切换游戏, peace/reversi - 添加新游戏, pass - 跳过, quit - 退出
请输入命令:
```

表 9 2.游戏结束

```
  A B C D E F G H    游戏信息              游戏列表
  ---------------    ----------------    ----------------
1 ○ ● ○ ○ ● ○ ● ○    游戏编号: 1     游戏列表:
2 ● ○ ● ○ ● ○ ● ○    游戏类型: peace     1. peace
3 ● ○ ● ● ○ ● ○ ●    玩家1: ●     2. reversi
4 ○ ● ○ ○ ● ● ○ ●    玩家2: ○ ←
5 ○ ● ○ ● ○ ● ○ ●
6 ○ ● ○ ● ○ ● ○ ●
7 ○ ● ○ ● ○ ● ○ ●
8 ○ ● ○ ● ○ ● ○ ●

游戏结束!
```

表 10 1.游戏结束

```
  A B C D E F G H    游戏信息              游戏列表
  ---------------    ----------------    ----------------
1 · · · · · · · ·    游戏编号: 6     游戏列表:
2 · · · · · · · ·    游戏类型: Gomoku     1. peace
3 · · · · · · · ·    玩家1: ● ←     2. reversi
4 · · · · · · · ·    玩家2: ○     3. Gomoku
5 · · · · · · · ·    当前轮数: 0     4. peace
6 · · · · · · · ·     5. Gomoku
7 · · · · · · · ·     6. Gomoku
8 · · · · · · · ·

命令: [坐标] - 落子, 数字 - 切换游戏, peace/reversi/gomoku - 添加新游戏, pass - 跳过, quit - 退出
请输入命令:
```

表 11 正确创建

表 12 胜利判断

表 13 平局