OOP Lab2 俞楚凡 实验报告

所有源代码位于`src`文件夹下。目录结构如下：

----`src`

    ----`game.java` 游戏主要逻辑部分

    ----`entites`

        ----`Board.class` 棋盘类

        ----`Piece.class` 棋盘格状态枚举类

        ----`Player.class` 玩家类

在Lab2中，我对课堂要求作了一些简单的延伸，实现了一个**完整的**黑白棋游戏。一些功能包括玩家落子位置合法性判断，游戏结束条件判断以及获胜方判断。

现将`game.java`下的关键代码做简要的说明和分析：

游戏首先初始化棋盘和玩家实例，随后在一个主循环函数`mainLoop`下运行。

```java
public static void main(String[] args) {  AInfinity-LilacDream
    // create new players
    Player player1 = new Player( name: "张三");
    Player player2 = new Player( name: "李四");

    // create new board
    Board board = new Board();
    board.init();

    mainLoop(player1, player2, board);
}
```

`mainLoop`函数中，首先判断游戏是否结束。如果棋盘已经被填满或双方玩家均没有了合法的落子位置，游戏结束。调用`endGame()`函数，进行获胜方判断，并输出结束信息。

```
if (isAllFilledFlag) {
    endGame(board);
    return;
}

if (!checkLegalStep(playerFlag ? Piece.BLACK : Piece.WHITE, board)) {
    if (!checkLegalStep(playerFlag ? Piece.WHITE : Piece.BLACK, board)) {
        endGame(board);
        return;
    }
    else {
        playerFlag = !playerFlag;
        isNoLegalStepFlag = true;
    }
}
```

随后，打印游戏棋盘，并等待玩家输入。若输入非法，游戏将会一直循环直到玩家下出了合法的一步。

```
// print frame
printWindow(player1, player2, board, playerFlag);
// Input
while (true) {
    System.out.printf("请玩家[%s]输入落子位置：", playerFlag ? player1.getName() : player2.getName());
    if (sc.hasNextLine()) {
        String str = sc.next();
        curX = Character.getNumericValue(str.charAt(0));
        curY = str.charAt(1) - 'A' + 1;

        if (board.board[curX][curY] != Piece.EMPTY) {
            System.out.printf("[%s]已经有棋子了！\n", str);
            continue;
        }

        if (!updateBoard(board, curX, curY)) {
            System.out.printf("[%s]是非法的落子位置！\n", str);
            continue;
        }
        break;
    }
}

playerFlag = !playerFlag;
```

玩家每下出一步，我们根据黑白棋规则更新整个棋盘。*具体地*，我们定义了一个方向数组`directions`，并依次朝八个方向判定，对于每个方向，我们将翻转当前棋子与同色棋子之间夹住的对手棋子。

```java
// after a move is performed, update the whole board
static boolean updateBoard(Board board, int x, int y) {  1 usage  AInfinity-LilacDream
    Piece currentPiece = playerFlag ? Piece.BLACK : Piece.WHITE;
    Piece oppositePiece = playerFlag ? Piece.WHITE : Piece.BLACK;

    boolean flipFlag = false;

    // direction enum
    int[][] directions = {
            {-1, 0},
            {1, 0},
            {0, -1},
            {0, 1},
            {-1, -1},
            {-1, 1},
            {1, -1},
            {1, 1}
    };

    // check each direction
    for (int[] dir : directions) {
        int dx = dir[0];
        int dy = dir[1];

        int tx = x + dx;
        int ty = y + dy;

        java.util.ArrayList<int[]> toFlip = new java.util.ArrayList<>();
        boolean canFlip = false;

        while (tx >= 1 && tx <= 8 && ty >= 1 && ty <= 8) {
            if (board.board[tx][ty] == oppositePiece) {
                toFlip.add(new int[]{tx, ty});
            } else if (board.board[tx][ty] == currentPiece && !toFlip.isEmpty()) {
                canFlip = true;
                break;
            } else break;
            tx += dx;
            ty += dy;
```

上为部分代码示例。

更多具体细节您可以从源码中查看，或者运行程序查看游戏运行情况。