俞楚凡 OOP Lab3

在`Lab2`的基础上进行了一些修改，实现了同时维护三个棋盘并记录状态的功能。

首先，每个棋盘应该是一个独立的`Board`实例，每个`Board`实例在维护当前盘面状态的同时还需要维护当前执棋玩家以及id（以显示到棋盘上）。

```java
package entities;

public class Board {  15 usages  AInfinity-LilacDream *
    public Piece[][] board = new Piece[9][9];  16 usages
    public Player currentPlayer;  11 usages
    public int id;  2 usages

    public void init(Player player, int id) {  3 usages  AInfinity-LilacDream *
        currentPlayer = player;
        this.id = id;

        for (int i = 1; i <= 8; ++i) {
            for (int j = 1; j <= 8; ++j) {
                board[i][j] = Piece.EMPTY;
            }
        }
        board[4][5] = Piece.BLACK;
        board[5][4] = Piece.BLACK;
        board[4][4] = Piece.WHITE;
        board[5][5] = Piece.WHITE;
    }
}

```

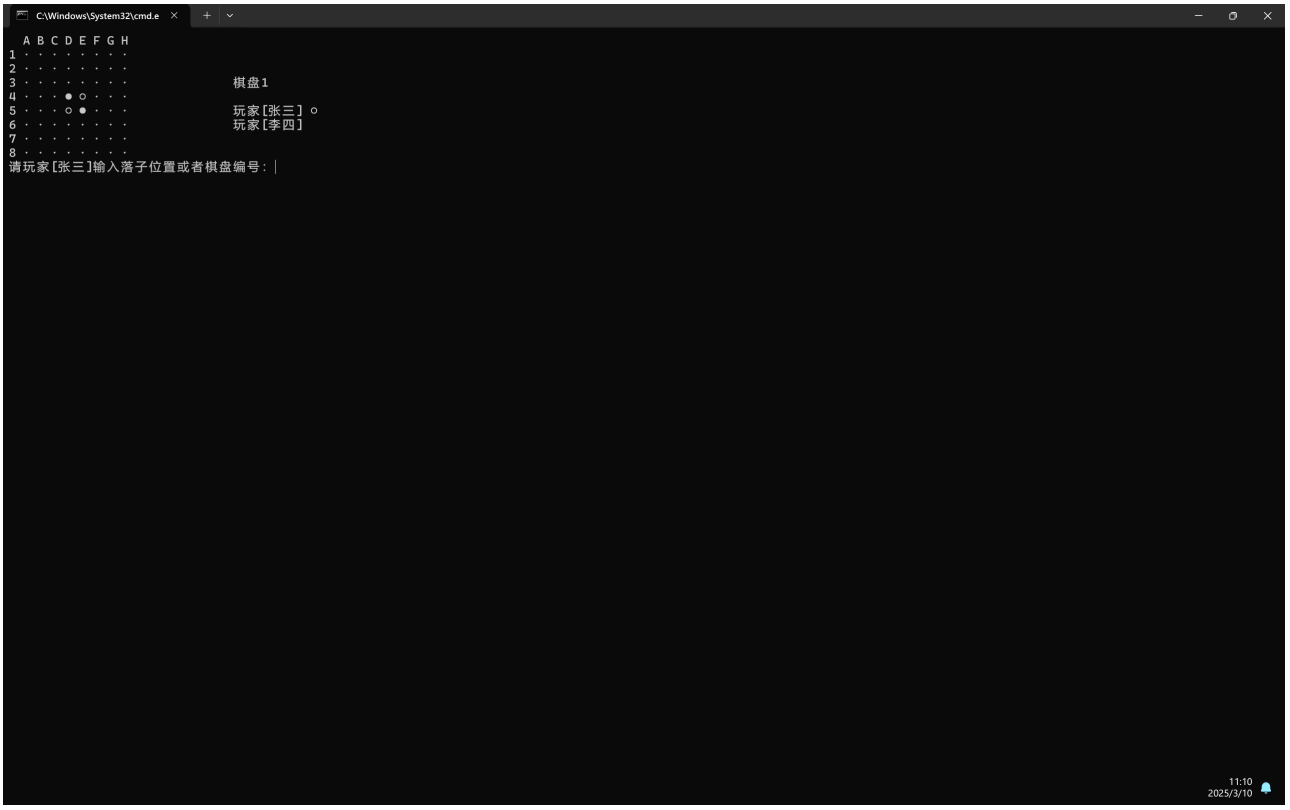
`Board`类 成员变量及函数

其次，由于`Lab2`中我的所有代码都将Board实例以参数形式传入函数中，因此在`Lab3`的修改变得较为方便：重新定义一个`CurrentBoard`变量记录当前棋盘，并将`CurrentBoard`代替`board1`传入函数即可。

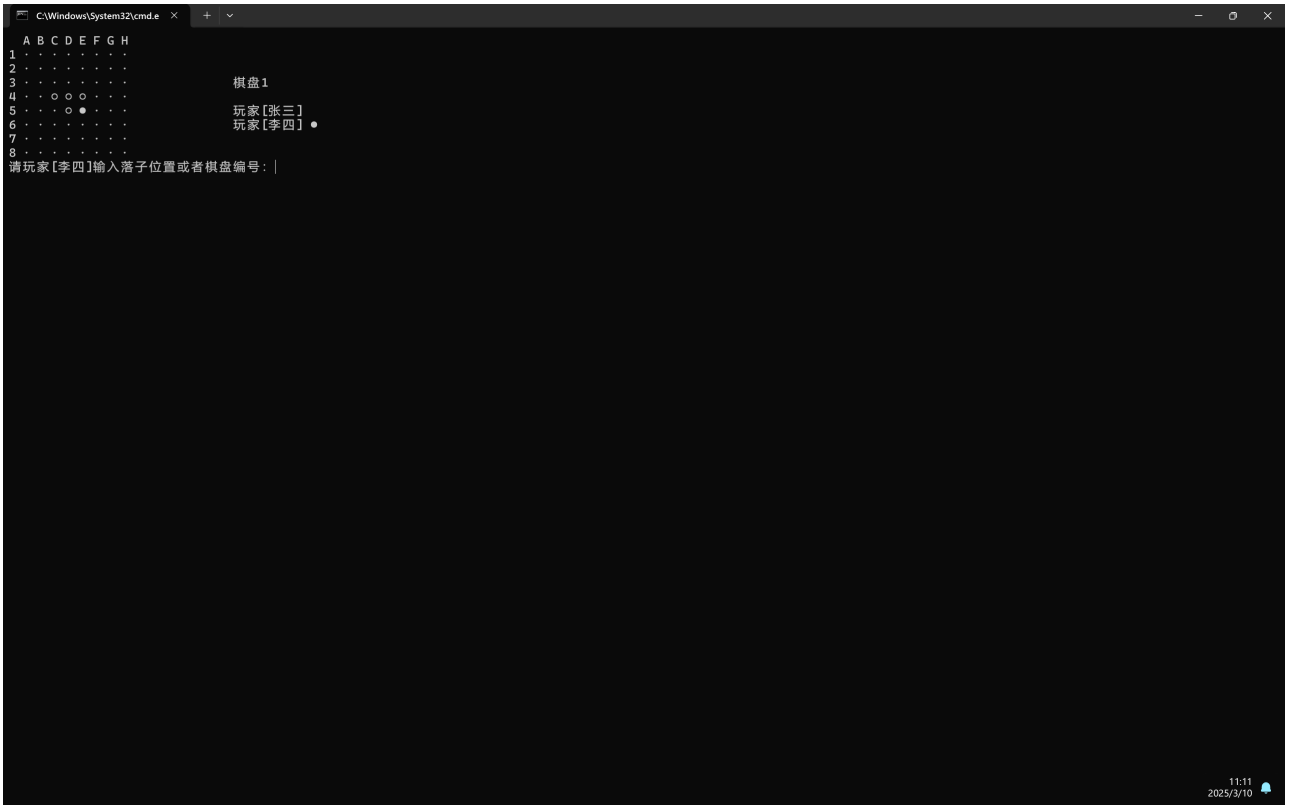
切换棋盘的功能主要是通过判断输入字符串的长度来实现的。

本实验报告承接`Lab2`的实验报告，因此只记录修改的思路。

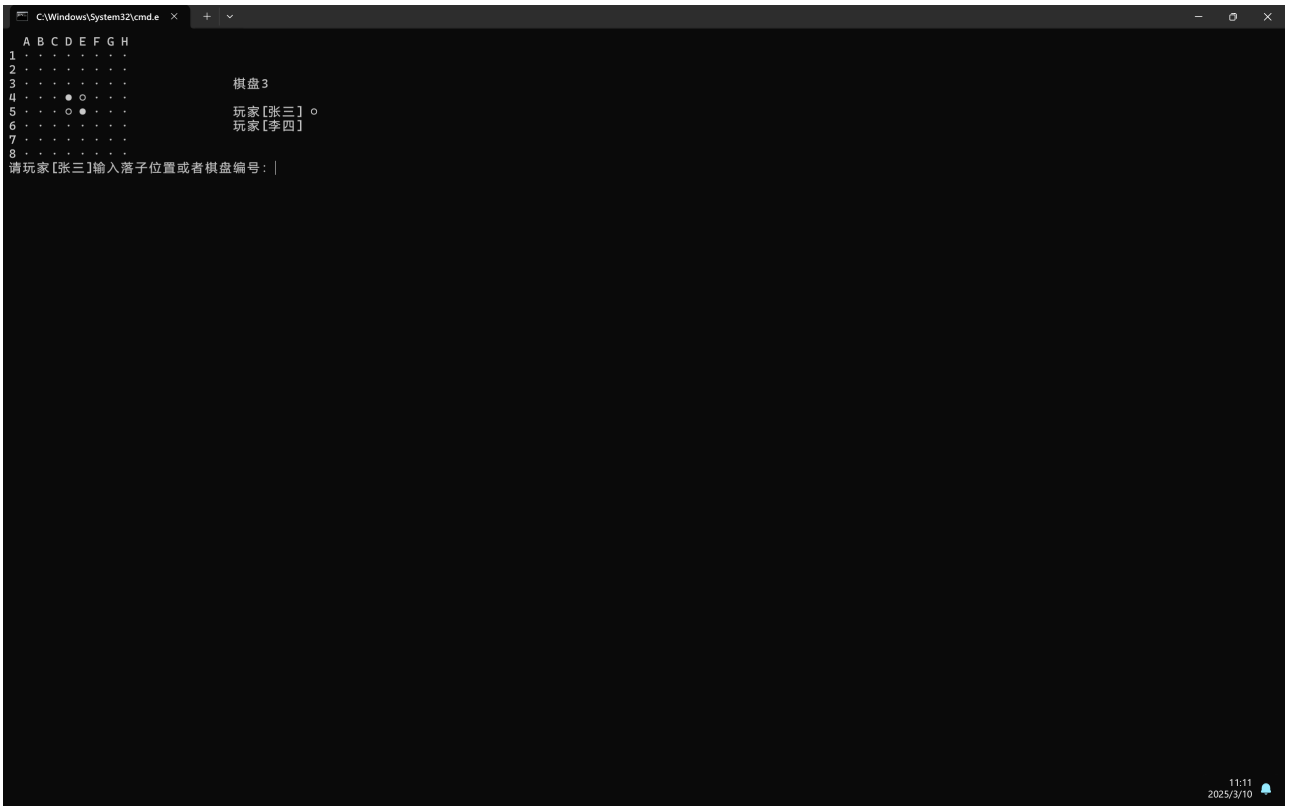
以下是运行时截图：



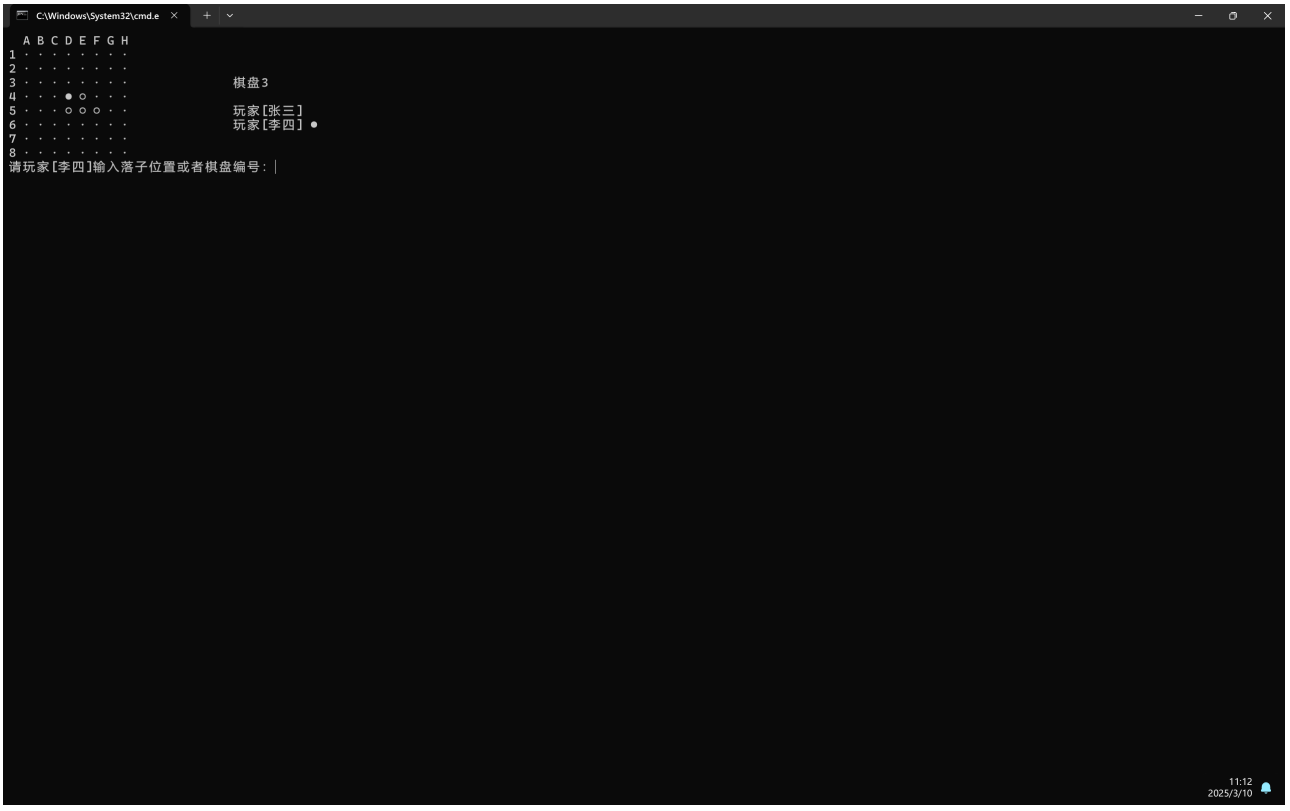
游戏开始时，默认显示棋盘1。



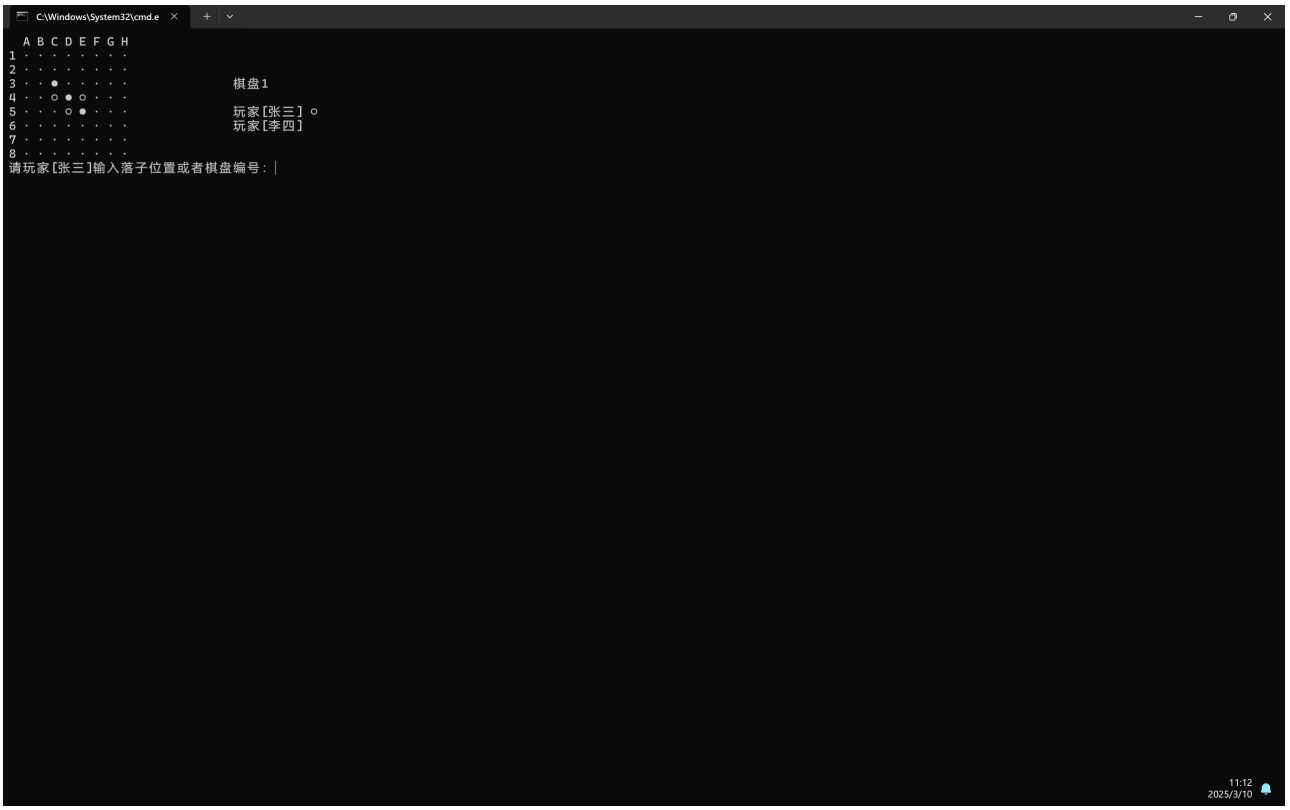
在输入“4c”后，更新棋盘并显示。



在输入“3”后，切换到棋盘3。



在棋盘3上走一步。



切换到棋盘1，再走一步。

在任意一个棋盘上决出胜负后，游戏结束。

END