黑白棋大概19世紀被發明
java大概28年前開始

所以以下如有雷同

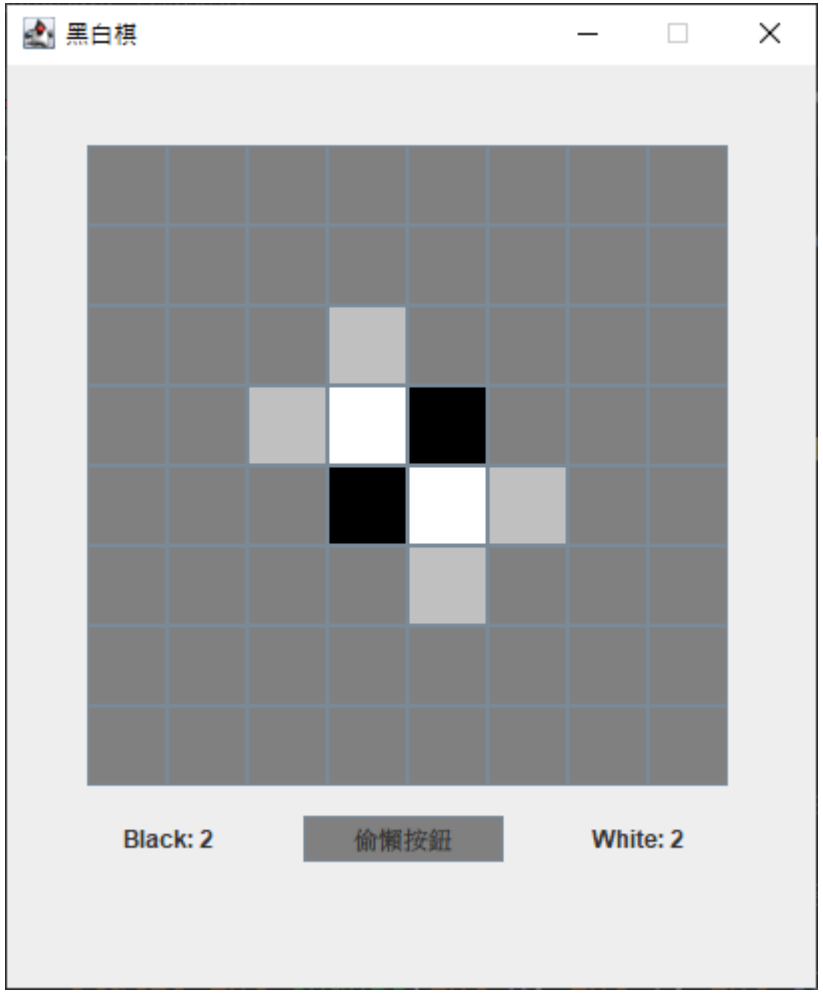
絕對抄襲

黑白棋 - Java實作

S10350114 S10350149 S10350131 都是2A

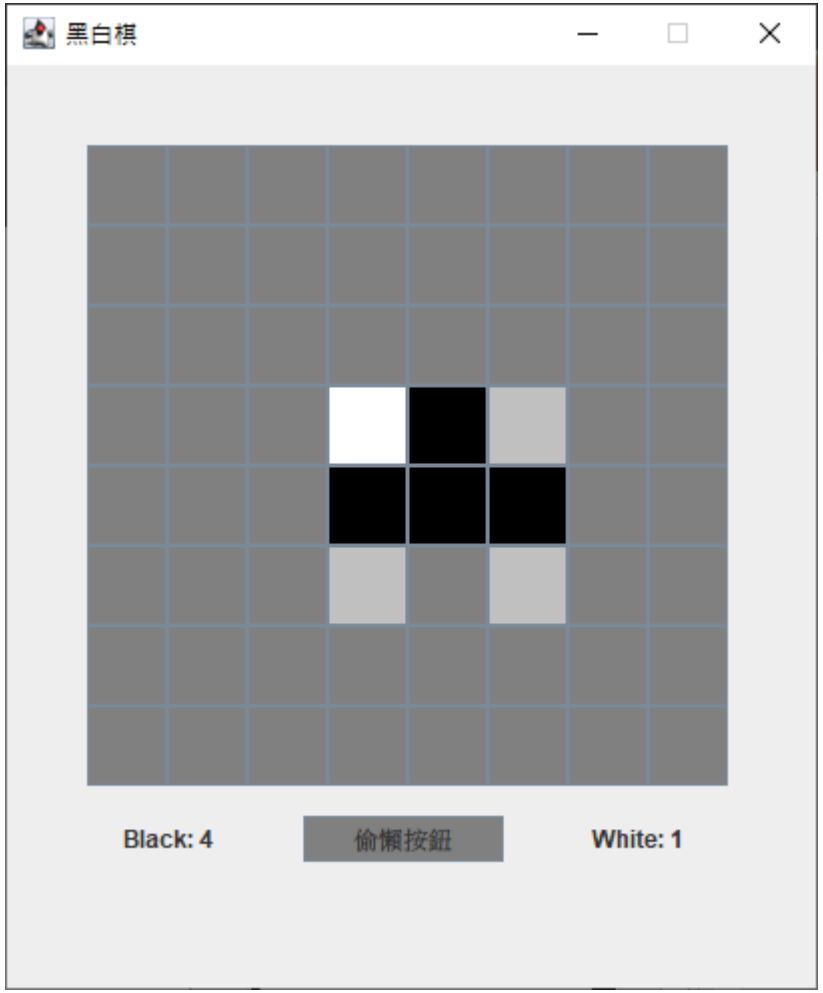
遊戲畫面

- 基本的黑棋白棋
- 可下棋的位置
- 簡單的計分
- 用來敷衍還沒做好的功能的按鈕



基本玩法

- 起始中央4顆棋子
- 黑棋先手
- 類似跳棋跨過對方棋子的概念
- 可以斜、直、橫三種方向
- 兩點一線翻轉中間對方的棋
- 不能有空格
- 下滿或雙方不能下棋時結束
- 棋多的人獲勝



專案介紹

- 主要使用Java Swing 製作
- Frame, Container, Button…

```
import java.awt.*;
import java.awt.Button;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

專案介紹

專案介紹

```
static int check(int x, int y, int color) { // 檢查是否可下子
static void Print() { // 印出棋盤 寫在change裡面
static int hasZero(String p) { // 找空棋格在哪, 搭配change函式使用
static int change(int x, int y, int color) { // 全方位遍歷 找可以轉的棋子
static void score() { // 遍歷計分
static int possible(int color) { // 看是否有東西下並記錄
static void setPossibleColor() { // 顯示可以下棋的位置
public game() { // 遊戲主體
```

```
public class game extends JFrame {
    // 遊戲初始化
    static int blackscore = 0;
    static int whitescore = 0;
    // 先用矩陣紀錄棋譜 0灰色 1白棋 2黑棋
    static int chess[][] = new int[8][8];
    // 再用按鈕加判斷式輸出棋盤
    static JButton block[][] = new JButton[8][8];
    // 可下棋的地方 possible
    static int poss[];
    static String pp;
    // 九宮格遍歷
    static int dir[][] = { { 0, 1 }, { 1, 0 }, { -
    // 黑棋先手
    static int color = 2;
    // 結束遊戲
    static int endCount = 0;
```

遊戲主體

- 以矩陣為依據建立按鈕
- 建立監聽器
- 在成功下棋之後換手
- 下一手找可以下棋的位置
- 刷新分數
- 最後記得把按鈕加進視窗

```
// 建立棋子
for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
        int ii = i;
        int jj = j;
        chess[i][j] = 0; // 0灰色
        block[i][j] = new JButton();
        block[i][j].setVisible(true);
        block[i][j].setBounds(40 + 40 * i, 40 + 40 * j, 40, 40);
        block[i][j].setBackground(Color.GRAY);
        block[i][j].addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (check(ii, jj, color) == 1) {
                    if (change(ii, jj, color) == 1) {
                        color = 3 - color; // 用三減一的方式去輪替1, 2
                    }
                }
                // 尋找可以下的棋格並上色
                int c = possible(color);
                setPossibleColor();
                // 計分
                score();
                bCount.setText("Black: " + blackscore);
                wCount.setText("White: " + whitescore);
                // 結束遊戲
            }
        });
        mainWin.add(block[i][j]);
```

零碎細節

```
JFrame mainWin = new JFrame();
Container con = mainWin.getContentPane();
con.setLayout(null);
// 結束視窗
JFrame endWin = new JFrame();
endWin.setSize(280, 100);
endWin.setTitle("結束");
endWin.setVisible(false);
// 分數顯示
JLabel bCount = new JLabel("", SwingConstants.LEFT);
bCount.setBounds(58, 375, 100, 23);
mainWin.add(bCount);
JLabel wCount = new JLabel("", SwingConstants.RIGHT);
wCount.setBounds(238, 375, 100, 23);
mainWin.add(wCount);
```

```
// 起始棋子
chess[3][3] = 1;
chess[4][4] = 1;
block[3][3].setBackground(Color.WHITE);
block[4][4].setBackground(Color.WHITE);
chess[3][4] = 2;
chess[4][3] = 2;
block[3][4].setBackground(Color.BLACK);
block[4][3].setBackground(Color.BLACK);
bCount.setText("Black: 2");
wCount.setText("White: 2");
possible(color);
setPossibleColor();
// 視窗設定
mainWin.setSize(420, 500);
mainWin.setTitle("黑白棋");
mainWin.setVisible(true);
mainWin.setResizable(false);
mainWin.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

函式介紹

- 檢查下棋的地方是否有效
- 1. 棋盤內 2. 旁邊有對手棋子 3.格子內沒有棋子

```
static int check(int x, int y, int color) { // 檢查是否可下子
    int diff = 3 - color; // 對手的棋子
    if (chess[x][y] != 1 && chess[x][y] != 2) { // 欲放格子沒有子
        for (int i = 0; i < 8; i++) {
            int tx = x + dir[i][0];
            int ty = y + dir[i][1];
            // 1. 範圍內 2. 有對手棋子
            if (tx >= 0 && tx < 8 && ty >= 0 && ty < 8 && chess[tx][ty] == diff) {
                return 1;
            }
        }
    }
    return 0;
}
```

函式介紹

- 尋找是否有空棋格

```
static int hasZero(String p) { // 找有沒有空棋格
    if (p.length() == 0) {
        return 0;
    }
    return p.indexOf("0");
}
```

- 每次刷新棋盤

```
static void Print() { // 印出棋盤 寫在change裡面
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            if (chess[i][j] == 1) {
                block[i][j].setBackground(Color.WHITE);
            } else if (chess[i][j] == 2) {
                block[i][j].setBackground(Color.BLACK);
            }
        }
    }
    return;
}
```

函式介紹

- 先同一個方向
- 紀錄有什麼棋子
- 遇到自己停下
- 如果有記錄到自己且沒有空棋格
- 成功翻轉
- 總共 8 個方向

```
static int change(int x, int y, int color) { // 全方位遍歷 找可以轉的棋子
    int isChange = 0;
    String p = "";
    int f = 0; // flip
    for (int i = x - 1; i >= 0; i--) {// west
        if (chess[i][y] == color) {
            f = 1;
            break;
        }
        p += String.valueOf(chess[i][y]);
    }
    if (hasZero(p) == -1 && f == 1) { // 不包含零且在有效空間裡
        isChange = 1;
        for (int i = x - 1; i >= 0; i--) { // 變色
            if (chess[i][y] == color) {
                break;
            }
            chess[i][y] = color;
        }
    }
```

函式介紹

```
f = 0;
p = "";
for (int i = x + 1; i < 8; i++) {// east
    if (chess[i][y] == color) {
        f = 1;
        break;
    }
    p += String.valueOf(chess[i][y]);
}
if (hasZero(p) == -1 && f == 1) {
    isChange = 1;
    for (int i = x + 1; i < 8; i++) {
        if (chess[i][y] == color) {
            break;
        }
        chess[i][y] = color;
    }
}
```

```
f = 0;
p = "";
for (int i = 1; i <= 8; i++) {// east-north
    int tx = x + i;
    int ty = y - i;
    if (tx >= 0 && tx < 8 && ty >= 0 && ty < 8) {
        if (chess[tx][ty] == color) {
            f = 1;
            break;
        }
        p += String.valueOf(chess[tx][ty]);
    }
}
if (hasZero(p) == -1 && f == 1) {
    isChange = 1;
    for (int i = 1; i <= 8; i++) {
        int tx = x + i;
        int ty = y - i;
        if (tx >= 0 && tx < 8 && ty >= 0 && ty < 8) {
            if (chess[tx][ty] == color) {
                break;
            }
            chess[tx][ty] = color;
        }
    }
}
```

函式介紹

- 成功翻轉後才下棋
- 刷新棋盤
- 回傳給監聽器

```
if (isChange == 1) {
    chess[x][y] = color;
    Print();
}

return isChange;
```

函式介紹

- 遍歷每格計分
- 就是計分

```
static void score() { // 遍歷計分
    blackscore = 0;
    whitescore = 0;
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            if (chess[i][j] == 1) {
                whitescore += 1;
            } else if (chess[i][j] == 2) {
                blackscore += 1;
            }
        }
    }
}
```

函式介紹

- 與上述函式雷同
- 先同一個方向
- 紀錄有什麼棋子
- 遇到自己停下
- 如果有記錄到自己且沒有空棋格
- 找到有效位置
- 紀錄
- 總共 8 個方向

```
static int possible(int color) { // 看是否有東西下並記錄
    // 0 == 沒東西下
    // possible position
    String pp = "";
    for (int x = 0; x < 8; x++) {
        for (int y = 0; y < 8; y++) {
            String p = "";
            int f = 0; // flip
            for (int i = x - 1; i >= 0; i--) {// west
                if (chess[i][y] == color) {
                    f = 1;
                    break;
                }
                p += String.valueOf(chess[i][y]);
            }
            if (hasZero(p) == -1 && f == 1) { // 不包含零且在有效空間裡
                // 紀錄有效子位置
                pp += x + "," + y + ",";
            }
```

函式介紹

- 最後輸出成陣列供後續使用
- 回傳監聽器還可以下棋

```
String temp[] = pp.split(",");
if (temp.length == 0) {
    return 0;
} else {
    poss = new int[temp.length];
    for (int i = 0; i < temp.length; i++) {
        poss[i] = Integer.valueOf(temp[i]);

    }
    return 1;
}
```

函式介紹

- 用剛剛找到的有效位置陣列和按鈕輸出

```
static void setPossibleColor() { // 顯示可以下棋的位置
    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            if (chess[i][j] == 0) {
                block[i][j].setBackground(Color.GRAY);
            }
        }
    }
    for (int k = 0; k < poss.length - 1; k += 2) {
        if (chess[poss[k]][poss[k + 1]] != 1 && chess[poss[k]][poss[k + 1]] != 2) {
            block[poss[k]][poss[k + 1]].setBackground(Color.LIGHT_GRAY);
        }
    }
}
```

函式介紹

■啟動遊戲

```
public static void main(String[] args) {
    new game();
}
```

偷懶按鈕

- 案下去會重開一局新遊戲
- 有問題就按兩次

```
// 偷懶按鍵
JButton newGame = new JButton("偷懶按鈕");
newGame.setVisible(true);
newGame.setBounds(148, 375, 100, 23);
newGame.setBackground(Color.GRAY);
newGame.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new game();
    }
});
mainWin.add(newGame);
```

改進部分

- 少做漂亮的結算畫面
- 本來要做Bot但出問題了沒辦法Demo

```
// 偷懶按鍵
JButton newGame = new JButton("偷懶按鈕");
newGame.setVisible(true);
newGame.setBounds(148, 375, 100, 23);
newGame.setBackground(Color.GRAY);
newGame.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        new game();
    }
});
mainWin.add(newGame);
```

謝謝謝大家