链滴

华容道数组实现

作者：ZephyrJung
原文链接：https://ld246.com/article/1483014200476
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

```
import java.util.Scanner;

/**
 * Created by Zephyr on 2016/12/29.
 */
public class HuaRongMap{
    private static int[][] map={
            {211,401,402,241},
            {213,403,404,243},
            {221,311,312,251},
            {223,111,121,253},
            {131,000,000,141}
    };
    private static final int width=4;
    private static final int height=5;
    private static int direction=1;//1,2,3,4，上下左右

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);

        while(true){
            click(sc.nextInt());
            if(map[4][1]==403){
                System.out.println("You win!");
            }else{
                printMap();
            }
        }
    }
    public static void click(int grid){
        if(grid==0)
            return;
        int pos=getPosition(grid);
        direction = grid%10;
        checkMove(grid);
    }

    public static boolean checkMove(int grid){
        //根据不同种类的方格检查当前方向是否可以移动
        //是则按照当前方向移动，否则修改方向
            //4（类型）*4（方向）
            int i=0,i1=0,i2=0,i3=0;
            int j=0,j1=0,j2=0,j3=0;
            switch(grid/100*10+direction) {
                case 11:
                    i = getPosition(grid) / 10;
                    j = getPosition(grid) % 10;
                    if (i - 1 >= 0 && map[i - 1][j] == 0) {
                        map[i - 1][j] = map[i][j];
                        map[i][j] = 0;
                        return true;
                    } else {
                        return false;
                    }
```

```
case 12:
    i = getPosition(grid/10*10+1) / 10;
    j = getPosition(grid/10*10+1) % 10;
    if (i + 1 < height && map[i + 1][j] == 0) {
        map[i + 1][j] = map[i][j];
        map[i][j] = 0;
        return true;
    } else {
        return false;
    }
case 13:
    i = getPosition(grid/10*10+1) / 10;
    j = getPosition(grid/10*10+1) % 10;
    if (j - 1 >= 0 && map[i][j - 1] == 0) {
        map[i][j - 1] = map[i][j];
        map[i][j] = 0;
        return true;
    } else {
        return false;
    }
case 14:
    i = getPosition(grid/10*10+1) / 10;
    j = getPosition(grid/10*10+1) % 10;
    if (j + 1 < width && map[i][j + 1] == 0) {
        map[i][j + 1] = map[i][j];
        map[i][j] = 0;
        return true;
    } else {
        return false;
    }
case 21://211
    i = getPosition(grid/10 * 10 + 1) / 10;
    j = getPosition(grid/10 * 10 + 1) % 10;
    if (i - 1 >= 0 && map[i - 1][j] == 0) {
        map[i - 1][j] = map[i][j];
        map[i][j] = map[i + 1][j];
        map[i + 1][j] = 0;
        return true;
    } else {
        return false;
    }
case 22://213
    i = getPosition(grid/10 * 10 + 3) / 10;
    j = getPosition(grid/10 * 10 + 3) % 10;
    if (i + 1 < height && map[i + 1][j] == 0) {
        map[i + 1][j] = map[i][j];
        map[i][j] = map[i - 1][j];
        map[i - 1][j] = 0;
        return true;
    } else {
        return false;
    }
case 23://211,213
    i1 = getPosition(grid/10 * 10 + 1) / 10;
```

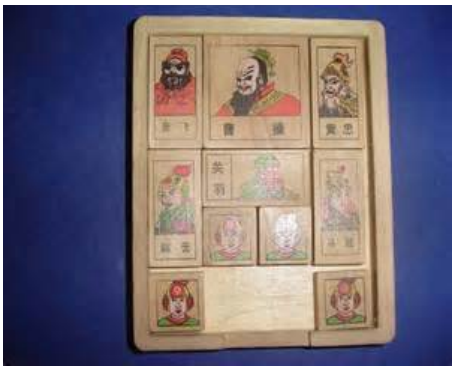
```
    i3 = getPosition(grid/10 * 10 + 3) / 10;
    j = getPosition(grid/10 * 10 + 1) % 10;
    if (j - 1 >= 0 && map[i1][j - 1] == 0 && map[i3][j - 1] == 0) {
        map[i1][j - 1] = map[i1][j];
        map[i3][j - 1] = map[i3][j];
        map[i1][j] = 0;
        map[i3][j] = 0;
        return true;
    } else {
        return false;
    }
case 24://211,213
    i1 = getPosition(grid/10 * 10 + 1) / 10;
    i3 = getPosition(grid/10 * 10 + 3) / 10;
    j = getPosition(grid/10 * 10 + 1) % 10;
    if (j + 1 < width && map[i1][j + 1] == 0 && map[i3][j + 1] == 0) {
        map[i1][j + 1] = map[i1][j];
        map[i3][j + 1] = map[i3][j];
        map[i1][j] = 0;
        map[i3][j] = 0;
        return true;
    } else {
        return false;
    }
case 31:
    i = getPosition(grid/10 * 10 + 1) / 10;
    j1 = getPosition(grid/10 * 10 + 1) % 10;
    j2 = getPosition(grid/10 * 10 + 2) % 10;
    if (i - 1 >= 0 && map[i - 1][j1] == 0 && map[i - 1][j2] == 0) {
        map[i - 1][j1] = map[i][j1];
        map[i - 1][j2] = map[i][j2];
        map[i][j1] = 0;
        map[i][j2] = 0;
        return true;
    } else {
        return false;
    }
case 32:
    i = getPosition(grid/10 * 10 + 1) / 10;
    j1 = getPosition(grid/10 * 10 + 1) % 10;
    j2 = getPosition(grid/10 * 10 + 2) % 10;
    if (i + 1 < height && map[i + 1][j1] == 0 && map[i + 1][j2] == 0) {
        map[i + 1][j1] = map[i][j1];
        map[i + 1][j2] = map[i][j2];
        map[i][j1] = 0;
        map[i][j2] = 0;
        return true;
    } else {
        return false;
    }
case 33:
    i = getPosition(grid/10 * 10 + 1) / 10;
    j = getPosition(grid/10 * 10 + 1) % 10;
    if (j - 1 >= 0 && map[i][j - 1] == 0) {
```

```
            map[i][j - 1] = map[i][j];
            map[i][j] = map[i][j + 1];
            map[i][j + 1] = 0;
            return true;
        } else {
            return false;
        }
    case 34:
        i = getPosition(grid/10 * 10 + 2) / 10;
        j = getPosition(grid/10 * 10 + 2) % 10;
        if (j + 1 < width && map[i][j + 1] == 0) {
            map[i][j + 1] = map[i][j];
            map[i][j] = map[i][j - 1];
            map[i][j - 1] = 0;
            return true;
        } else {
            return false;
        }
    case 41:
        i = getPosition(grid/10 * 10 + 1) / 10;
        j1 = getPosition(grid/10 * 10 + 1) % 10;
        j2 = getPosition(grid/10 * 10 + 2) % 10;
        if (i - 1 >= 0 && map[i - 1][j1] == 0 && map[i - 1][j2] == 0) {
            map[i - 1][j1] = map[i][j1];
            map[i][j1] = map[i + 1][j1];
            map[i + 1][j1] = 0;
            map[i - 1][j2] = map[i][j2];
            map[i][j2] = map[i + 1][j2];
            map[i + 1][j2] = 0;
            return true;
        } else {
            return false;
        }
    case 42:
        i = getPosition(grid/10 * 10 + 3) / 10;
        j1 = getPosition(grid/10 * 10 + 1) % 10;
        j2 = getPosition(grid/10 * 10 + 2) % 10;
        if (i + 1 < height && map[i + 1][j1] == 0 && map[i + 1][j2] == 0) {
            map[i + 1][j1] = map[i][j1];
            map[i][j1] = map[i - 1][j1];
            map[i - 1][j1] = 0;
            map[i + 1][j2] = map[i][j2];
            map[i][j2] = map[i - 1][j2];
            map[i - 1][j2] = 0;
            return true;
        } else {
            return false;
        }
    case 43:
        i1 = getPosition(grid/10 * 10 + 1) / 10;
        i3 = getPosition(grid/10 * 10 + 3) / 10;
        j = getPosition(grid/10 * 10 + 1) % 10;
        if (j - 1 >= 0 && map[i1][j - 1] == 0 && map[i3][j - 1] == 0) {
            map[i][j1 - 1] = map[i][j1];
```

```
                    map[i][j1] = map[i][j1 + 1];
                    map[i][j1 + 1] = 0;
                    map[i][j2 - 1] = map[i][j2];
                    map[i][j2] = map[i][j2 + 1];
                    map[i][j2 + 1] = 0;
                    return true;
                } else {
                    return false;
                }
            case 44:
                i1 = getPosition(grid/10 * 10 + 1) / 10;
                i3 = getPosition(grid/10 * 10 + 3) / 10;
                j = getPosition(grid/10 * 10 + 2) % 10;
                if (j + 1 < width && map[i1][j + 1] == 0 && map[i3][j + 1] == 0) {
                    map[i][j1 + 1] = map[i][j1];
                    map[i][j1] = map[i][j1 - 1];
                    map[i][j1 - 1] = 0;
                    map[i][j2 + 1] = map[i][j2];
                    map[i][j2] = map[i][j2 - 1];
                    map[i][j2 - 1] = 0;
                    return true;
                } else {
                    return false;
                }
        }
    return false;
}


public static int getPosition(int grid){
    for(int i=0;i<height;i++){
        for(int j=0;j<width;j++){
            if(grid==map[i][j]){
                return i*10+j;
            }
        }
    }
    return -1;
}

public static void printMap(){
    for(int i=0;i<height;i++){
        for(int j=0;j<width;j++){
            if(map[i][j]==0){
                System.out.print("000 ");
            }else {
                System.out.print(map[i][j] + " ");
            }
        }
        System.out.println();
    }
}
}
```



根据华容道的划分，将不同个字进行了编码，假设最小单位为一个方格（如下图左下角的小兵）

华容道的图形编码规则如下：

总共三位，

第一位；竖向两个单位的以2开头，曹操以4开头，横向两个单位的以3开头，一个单位的一1开头

第二位：同类型单位的编号，如曹操只有1，小兵则分别为11x,12x,13x,14x

第三位：方向编码，从上到下，从左到右编码为1,2,3,4，如果没有则跳过，如左上角竖向两个单位的表示为:211,213。

所以上图的图形编码可以如下表示（000只是为了看起来方便）：

{211,401,402,241},

{213,403,404,243},

{221,311,312,251},

{223,111,121,253},

{131,000,000,141}

运行后，需要输入三位数，第一位是格子类型，第二位是同类型的编号（前两位就是要移动的小兵编的前两位），第三位为方向，上下左右依次编码为1、2、3、4，比如我要111这个小兵向下移动，则入112

每次输入就是打印出数组当前状况。