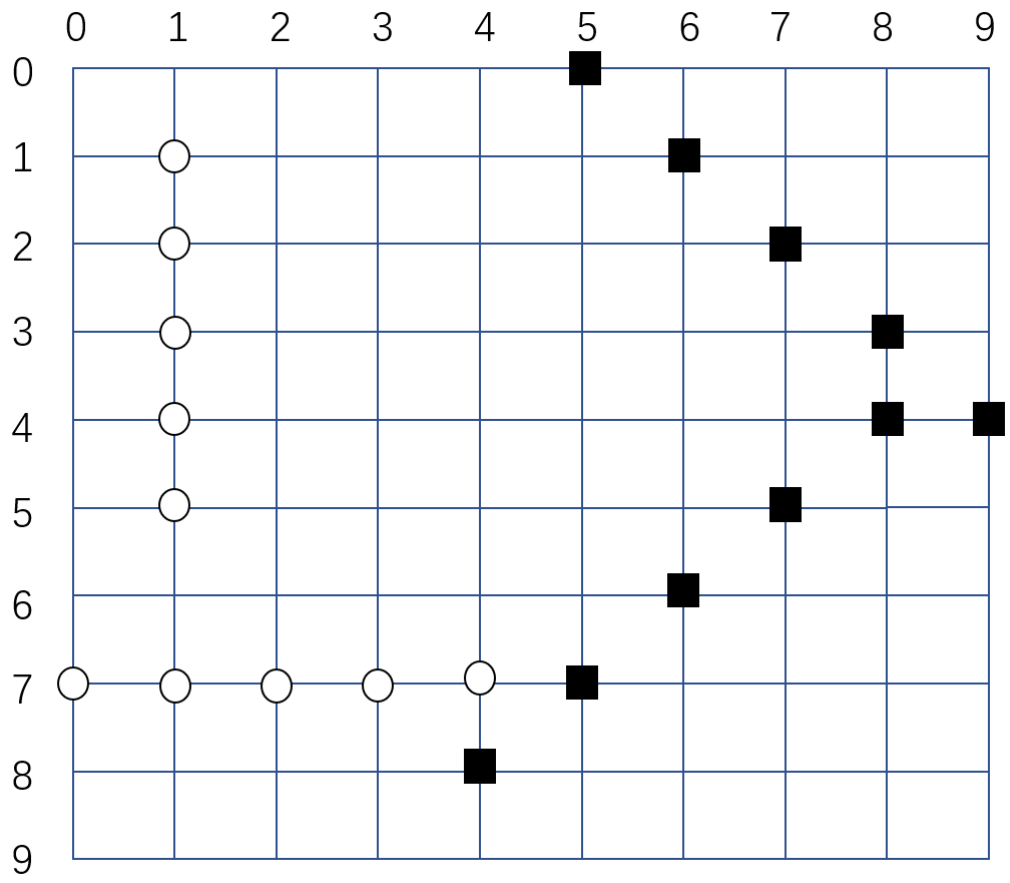
阶段项目--五子棋

目的

串联所学的知识点，从而检测出自己的不足，然后弥补自己的不足

需求说明

五子棋棋盘为一个10 x 10的棋盘，五子棋玩家共2个（这里分别称为A和B），A在棋盘上落子后，B再落子，依次往复，直到一方胜利或者棋盘空间用完为止。判断胜利的条件是一条直线或者任意斜线上同时存在A或者B的连续5颗棋子（见下图）。



技术实现

1. 静态变量

语法

```
public static 数据类型 变量名 = 变量值;
```

解释说明

静态变量只能定义类中，不能定义在方法中。静态变量可以在static修饰的方法中使用，也可以在非静态的方法中访问。主要解决在静态方法中不能访问非静态的变量。

2. 静态方法

语法

```
public static 返回值类型 方法名(){

}
```

解释说明

静态方法就相当于一个箱子，只是这个箱子中装的是代码，需要使用这些代码的时候，就把这个箱子放在指定的位置即可。

示例

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

public static void show(){
    System.out.println("张三");
    System.out.println("男");
    System.out.println("20");
}
```

实现步骤分析

1. 制作棋盘

a. 使用输入法中的制表符在控制台直接打印出棋盘，然后寻找落子位置的特征

```
System.out.println("┌────┬────┬────┬────┬────┬────┬────┬────┬────┐");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("├────┼────┼────┼────┼────┼────┼────┼────┼────┤");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("├────┼────┼────┼────┼────┼────┼────┼────┼────┤");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("├────┼────┼────┼────┼────┼────┼────┼────┼────┤");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("├────┼────┼────┼────┼────┼────┼────┼────┼────┤");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("├────┼────┼────┼────┼────┼────┼────┼────┼────┤");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("├────┼────┼────┼────┼────┼────┼────┼────┼────┤");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("├────┼────┼────┼────┼────┼────┼────┼────┼────┤");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("├────┼────┼────┼────┼────┼────┼────┼────┼────┤");
System.out.println("│    │    │    │    │    │    │    │    │    │");
System.out.println("└────┴────┴────┴────┴────┴────┴────┴────┴────┘");
```

b. 利用二维数组重新制作棋盘

```
public class Gobang {
```

```
2
3       public static char[][] chessboard = {
4               {'┌','┬','┬','┬','┬','┬','┬','┬','┬','┐'},
5               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
6               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
7               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
8               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
9               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
10              {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
11              {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
12              {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
13              {'└','┴','┴','┴','┴','┴','┴','┴','┴','┘'}
14      };
15
16      public static String separator = "——————";
17
18      public static void main(String[] args) {
19          System.out.println("   0     1     2     3     4     5     6     7
        8     9");
20          for(int i=0; i<chessboard.length; i++){ //外层循环控制行
21              System.out.print(i + "   ");
22              for(int j=0; j<chessboard[i].length; j++){//内层循环控制列
23                  if(j == chessboard[i].length - 1){ //最后一列
24                      System.out.print(chessboard[i][j]);
25                  } else {
26                      System.out.print(chessboard[i][j] + separator);//使用
    print打印，不换行
27                  }
28              }
29              System.out.println();
30              if(i < chessboard.length -1){//排除最后一行
31                  System.out.println("    |     |     |     |     |
    |     |     |     |");
32              }
33          }
34      }
35  }
```

c. 棋盘在玩家使用过程中会反复展示，需要使用方法来进行优化

```
1   public class Gobang {
2
3       public static char[][] chessboard = {
4               {'┌','┬','┬','┬','┬','┬','┬','┬','┬','┐'},
5               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
6               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
7               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
8               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
9               {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
10              {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
11              {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
12              {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
13              {'└','┴','┴','┴','┴','┴','┴','┴','┴','┘'}
14      };
15
16      public static String separator = "——————";
17
```

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

    public static void showChessboard(){
        System.out.println("   0     1     2     3     4     5     6     7     8     9");
        for(int i=0; i<chessboard.length; i++){ //外层循环控制行
            System.out.print(i + "   ");
            for(int j=0; j<chessboard[i].length; j++){//内层循环控制列
                if(j == chessboard[i].length - 1){ //最后一列
                    System.out.print(chessboard[i][j]);
                } else {
                    System.out.print(chessboard[i][j] + separator);//使用print打印，不换行
                }
            }
            System.out.println();
            if(i < chessboard.length -1){//排除最后一行
                System.out.println("    |     |     |     |     |     |     |     |     |");
            }
        }
    }
}
```

2. 落子

a. 玩家A、B会交替落子

```
/**
 * 五子棋
 */
public class Gobang {

    public static char[][] chessboard = {
            {'┌','┬','┬','┬','┬','┬','┬','┬','┬','┐'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'└','┴','┴','┴','┴','┴','┴','┴','┴','┘'}
    };

    public static String separator = "——————";

    public static void main(String[] args) {
        showChessboard();
        int totalPosition = chessboard.length * chessboard[0].length;
        for(int times=0; times < totalPosition; times++){
```

```
            System.out.println(times % 2 == 0 ? "请玩家A落子：" : "请玩家B落子");
        }
    }

    public static void showChessboard(){
        System.out.println("   0     1     2     3     4     5     6     7     8     9");
        for(int i=0; i<chessboard.length; i++){ //外层循环控制行
            System.out.print(i + "   ");
            for(int j=0; j<chessboard[i].length; j++){//内层循环控制列
                if(j == chessboard[i].length - 1){ //最后一列
                    System.out.print(chessboard[i][j]);
                } else {
                    System.out.print(chessboard[i][j] + separator);//使用print打印，不换行
                }
            }
            System.out.println();
            if(i < chessboard.length -1){//排除最后一行
                System.out.println("    |     |     |     |     |     |     |     |     |");
            }
        }

    }
}
```

b. 落子位置必须是0~100之间的整数，且不能使用已经存在棋子的位置

```
import java.util.Scanner;

/**
 * 五子棋
 */
public class Gobang {

    public static char[][] chessboard = {
            {'┌','┬','┬','┬','┬','┬','┬','┬','┬','┐'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'└','┴','┴','┴','┴','┴','┴','┴','┴','┘'}
    };

    public static String separator = "——————";

    public static char pieceA = 'o'; //玩家A的棋子
    public static char pieceB = '■';//玩家B的棋子

    public static void main(String[] args) {
        showChessboard();
```

```
        int totalPosition = chessboard.length * chessboard[0].length;
        Scanner sc = new Scanner(System.in);
        for(int times=0; times < totalPosition; times++){
            System.out.println(times % 2 == 0 ? "请玩家A落子：" : "请玩家B落子：");
            while(true){//保证落子成功的一个循环
                //检测Scanner中是否有输入的数据并且判断数据是否为整数，如果没有数据，就需要从控制台输入
                if(sc.hasNextInt()){
                    int position = sc.nextInt();
                    if(position >= 0 && position < totalPosition){
                        char currentPiece = (times % 2 == 0) ? pieceA : pieceB;
                        // position = 21   13x13      21 / 13 = 1  21 % 13 = 8
                        int row = position / chessboard.length; //位置除以棋盘数组的长度得到行号
                        int col = position % chessboard[0].length; //位置取模棋盘数组的总列数得到列号
                        if(chessboard[row][col] == pieceA || chessboard[row][col] == pieceB){
                            System.out.println("非法落子，请重新选择落子位置");
                            continue;
                        } else {
                            chessboard[row][col] = currentPiece;
                            break;
                        }
                    } else {
                        System.out.println("非法落子，请重新选择落子位置");
                    }
                } else {
                    System.out.println("非法落子，请重新选择落子位置");
                    sc.next();//将Scanner中存储的数据取出来，防止死循环
                }
            }
            //落子完成后，棋盘需要重新展示
            showChessboard();
        }
    }

    public static void showChessboard(){
        System.out.println("   0     1     2     3     4     5     6     7     8     9");
        for(int i=0; i<chessboard.length; i++){ //外层循环控制行
            System.out.print(i + "   ");
            for(int j=0; j<chessboard[i].length; j++){//内层循环控制列
                if(j == chessboard[i].length - 1){ //最后一列
                    System.out.print(chessboard[i][j]);
                } else {
                    System.out.print(chessboard[i][j] + separator);//使用print打印，不换行
                }
            }
            System.out.println();
            if(i < chessboard.length -1){//排除最后一行
                System.out.println("    |     |     |     |     |     |     |     |     |     |");
            }
```

```
        }

    }
}
```

c. 落子完成后，需要校验是否获胜

```
import java.util.Scanner;

/**
 * 五子棋
 */
public class Gobang {

    public static char[][] chessboard = {
            {'┌','┬','┬','┬','┬','┬','┬','┬','┬','┐'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'└','┴','┴','┴','┴','┴','┴','┴','┴','┘'}
    };

    public static String separator = "------";

    public static char pieceA = '○'; //玩家A的棋子
    public static char pieceB = '■';//玩家B的棋子

    public static void main(String[] args) {
        showChessboard();
        int totalPosition = chessboard.length * chessboard[0].length;
        Scanner sc = new Scanner(System.in);
        outer:
        for(int times=0; times < totalPosition; times++){
            System.out.println(times % 2 == 0 ? "请玩家A落子：" : "请玩家B落子：");
            char currentPiece = (times % 2 == 0) ? pieceA : pieceB;//当前使用的棋子
            while(true){//保证落子成功的一个循环
                //检测Scanner中是否有输入的数据并且判断数据是否为整数，如果没有数据，就需要从控制台输入
                if(sc.hasNextInt()){
                    int position = sc.nextInt();
                    if(position >= 0 && position < totalPosition){
                        // position = 21   13x13      21 / 13 = 1  21 % 13 = 8
                        int row = position / chessboard.length; //位置除以棋盘数组的长度得到行号
                        int col = position % chessboard[0].length; //位置取模棋盘数组的总列数得到列号
                        if(chessboard[row][col] == pieceA || chessboard[row][col] == pieceB){
                            System.out.println("非法落子，请重新选择落子位置");
```

```
                            continue;
                        } else {
                            chessboard[row][col] = currentPiece;
                            break;
                        }
                    } else {
                        System.out.println("非法落子，请重新选择落子位置");
                    }
                } else {
                    System.out.println("非法落子，请重新选择落子位置");
                    sc.next();//将Scanner中存储的数据取出来，防止死循环
                }
            }
            //落子完成后，棋盘需要重新展示
            showChessboard();
            //判断获胜情况，一共4种
            for(int i=0; i<chessboard.length; i++){
                for(int j=0; j<chessboard[i].length; j++){
                    //第一种：水平方向上存在同一玩家的连续5颗棋子
                    //(i, j) (i, j+1) (i, j+2) (i, j+3) (i, j+4)
                    boolean case1 = (j + 4 < chessboard[i].length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i][j+1] == currentPiece
                            && chessboard[i][j+2] == currentPiece
                            && chessboard[i][j+3] == currentPiece
                            && chessboard[i][j+4] == currentPiece;
                    //第二种：锤子方向上存在同一玩家的连续5颗棋子
                    //(i, j) (i+1, j) (i+2, j) (i+3, j) (i+4, j)
                    boolean case2 = (i + 4 < chessboard.length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i+1][j] == currentPiece
                            && chessboard[i+2][j] == currentPiece
                            && chessboard[i+3][j] == currentPiece
                            && chessboard[i+4][j] == currentPiece;
                    //第三种：135°角上存在同一玩家的连续5颗棋子
                    //(i, j) (i+1, j+1) (i+2, j+2) (i+3, j+3) (i+4, j+4)
                    boolean case3 = (i + 4 < chessboard.length)
                            && (j + 4 < chessboard[i].length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i+1][j+1] == currentPiece
                            && chessboard[i+2][j+2] == currentPiece
                            && chessboard[i+3][j+3] == currentPiece
                            && chessboard[i+4][j+4] == currentPiece;
                    //第四种：45°角上存在同一玩家的连续5颗棋子
                    //(i, j) (i-1, j+1) (i-2, j+2) (i-3, j+3) (i-4, j+4)
                    boolean case4 = (i > 4) && (j + 4 <
chessboard[i].length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i-1][j+1] == currentPiece
                            && chessboard[i-2][j+2] == currentPiece
                            && chessboard[i-3][j+3] == currentPiece
                            && chessboard[i-4][j+4] == currentPiece;
                    if(case1 || case2 || case3 || case4){
                        System.out.println(times % 2 == 0 ? "玩家A获得胜利"
: "玩家B获得胜利");
                        break outer;
                    }
                }
```

```
            }
        }
    }

    public static void showChessboard(){
        System.out.println("   0     1     2     3     4     5     6     7
    8     9");
        for(int i=0; i<chessboard.length; i++){ //外层循环控制行
            System.out.print(i + "   ");
            for(int j=0; j<chessboard[i].length; j++){//内层循环控制列
                if(j == chessboard[i].length - 1){ //最后一列
                    System.out.print(chessboard[i][j]);
                } else {
                    System.out.print(chessboard[i][j] + separator);//使用
print打印，不换行
                }
            }
            System.out.println();
            if(i < chessboard.length -1){//排除最后一行
                System.out.println("    |     |     |     |     |     |
  |     |     |     |");
            }
        }

    }
}
```

d. 棋盘使用完毕还未分出胜负，需要提示

```
import java.util.Scanner;

/**
 * 五子棋
 */
public class Gobang {

    public static char[][] chessboard = {
            {'┌','┬','┬','┬','┬','┬','┬','┬','┬','┐'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'└','┴','┴','┴','┴','┴','┴','┴','┴','┘'}
    };

    public static String separator = "——";

    public static char pieceA = '○'; //玩家A的棋子

    public static char pieceB = '■';//玩家B的棋子

    public static int times = 0; //记录棋盘使用次数，偶数次玩家A落子，奇数次玩家B
落子
```

```
28
29      public static void main(String[] args) {
30          showChessboard();
31          int totalPosition = chessboard.length * chessboard[0].length;
32          Scanner sc = new Scanner(System.in);
33          outer:
34          while (times < totalPosition){
35              System.out.println(times % 2 == 0 ? "请玩家A落子：" : "请玩家B落
子：");
36              char currentPiece = (times % 2 == 0) ? pieceA : pieceB;//当前使
用的棋子
37              while(true){//保证落子成功的一个循环
38                  //检测Scanner中是否有输入的数据并且判断数据是否为整数，如果没有数
据，就需要从控制台输入
39                  if(sc.hasNextInt()){
40                      int position = sc.nextInt();
41                      if(position >= 0 && position < totalPosition){
42                          // position = 21   13x13      21 / 13 = 1  21 % 13
= 8
43                          int row = position / chessboard.length; //位置除以棋
盘数组的长度得到行号
44                          int col = position % chessboard[0].length; //位置取
模棋盘数组的总列数得到列号
45                          if(chessboard[row][col] == pieceA ||
chessboard[row][col] == pieceB){
46                              System.out.println("非法落子，请重新选择落子位置");
47                              continue;
48                          } else {
49                              chessboard[row][col] = currentPiece;
50                              break;
51                          }
52                      } else {
53                          System.out.println("非法落子，请重新选择落子位置");
54                      }
55                  } else {
56                      System.out.println("非法落子，请重新选择落子位置");
57                      sc.next();//将Scanner中存储的数据取出来，防止死循环
58                  }
59              }
60              //落子完成后，棋盘需要重新展示
61              showChessboard();
62              //判断获胜情况，一共4种
63              for(int i=0; i<chessboard.length; i++){
64                  for(int j=0; j<chessboard[i].length; j++){
65                      //第一种：水平方向上存在同一玩家的连续5颗棋子
66                      //(i, j) (i, j+1) (i, j+2) (i, j+3) (i, j+4)
67                      boolean case1 = (j + 4 < chessboard[i].length)
68                              && chessboard[i][j] == currentPiece
69                              && chessboard[i][j+1] == currentPiece
70                              && chessboard[i][j+2] == currentPiece
71                              && chessboard[i][j+3] == currentPiece
72                              && chessboard[i][j+4] == currentPiece;
73                      //第二种：锤子方向上存在同一玩家的连续5颗棋子
74                      //(i, j) (i+1, j) (i+2, j) (i+3, j) (i+4, j)
75                      boolean case2 = (i + 4 < chessboard.length)
76                              && chessboard[i][j] == currentPiece
77                              && chessboard[i+1][j] == currentPiece
78                              && chessboard[i+2][j] == currentPiece
```

```
                            && chessboard[i+3][j] == currentPiece
                            && chessboard[i+4][j] == currentPiece;
                    //第三种：135°角上存在同一玩家的连续5颗棋子
                    //(i, j) (i+1, j+1) (i+2, j+2) (i+3, j+3) (i+4, j+4)
                    boolean case3 = (i + 4 < chessboard.length)
                            && (j + 4 < chessboard[i].length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i+1][j+1] == currentPiece
                            && chessboard[i+2][j+2] == currentPiece
                            && chessboard[i+3][j+3] == currentPiece
                            && chessboard[i+4][j+4] == currentPiece;
                    //第四种：45°角上存在同一玩家的连续5颗棋子
                    //(i, j) (i-1, j+1) (i-2, j+2) (i-3, j+3) (i-4, j+4)
                    boolean case4 = (i > 4) && (j + 4 <
chessboard[i].length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i-1][j+1] == currentPiece
                            && chessboard[i-2][j+2] == currentPiece
                            && chessboard[i-3][j+3] == currentPiece
                            && chessboard[i-4][j+4] == currentPiece;
                    if(case1 || case2 || case3 || case4){
                        System.out.println(times % 2 == 0 ? "玩家A获得胜利"
: "玩家B获得胜利");
                        break outer;
                    }
                }
            }
            times++;
        }
        if(times == totalPosition){//说明棋盘已经用完还未分出胜负
            System.out.println("平均");
        }
    }

    public static void showChessboard(){
        System.out.println("   0     1     2     3     4     5     6     7
    8     9");
        for(int i=0; i<chessboard.length; i++){ //外层循环控制行
            System.out.print(i + "   ");
            for(int j=0; j<chessboard[i].length; j++){//内层循环控制列
                if(j == chessboard[i].length - 1){ //最后一列
                    System.out.print(chessboard[i][j]);
                } else {
                    System.out.print(chessboard[i][j] + separator);//使用
print打印，不换行
                }
            }
            System.out.println();
            if(i < chessboard.length -1){//排除最后一行
                System.out.println("    |     |     |     |     |     |
  |     |     |     |");
            }
        }

    }
}
```

3. 声音特效

为落子、非法落子及获胜添加音效

```
import java.applet.Applet;
import java.applet.AudioClip;
import java.net.URL;
import java.util.Scanner;

/**
 * 五子棋
 */
public class Gobang {

    public static char[][] chessboard = {
            {'┌','┬','┬','┬','┬','┬','┬','┬','┬','┐'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'├','┼','┼','┼','┼','┼','┼','┼','┼','┤'},
            {'└','┴','┴','┴','┴','┴','┴','┴','┴','┘'}
    };

    public static String separator = "————";

    public static char pieceA = '○'; //玩家A的棋子

    public static char pieceB = '■';//玩家B的棋子

    public static int times = 0; //记录棋盘使用次数，偶数次玩家A落子，奇数次玩家B落子

    public static void main(String[] args) {
        showChessboard();
        int totalPosition = chessboard.length * chessboard[0].length;
        Scanner sc = new Scanner(System.in);
        outer:
        while (times < totalPosition){
            System.out.println(times % 2 == 0 ? "请玩家A落子：" : "请玩家B落子：");
            char currentPiece = (times % 2 == 0) ? pieceA : pieceB;//当前使用的棋子
            while(true){//保证落子成功的一个循环
                //检测Scanner中是否有输入的数据并且判断数据是否为整数，如果没有数据，就需要从控制台输入
                if(sc.hasNextInt()){
                    int position = sc.nextInt();
                    if(position >= 0 && position < totalPosition){
                        // position = 21   13x13      21 / 13 = 1  21 % 13 = 8
                        int row = position / chessboard.length; //位置除以棋盘数组的长度得到行号
```

```
                        int col = position % chessboard[0].length; //位置取模棋盘数组的总列数得到列号
                        if(chessboard[row][col] == pieceA || chessboard[row][col] == pieceB){
                            System.out.println("非法落子，请重新选择落子位置");
                            playAudio("illegal.wav");
                            continue;
                        } else {
                            chessboard[row][col] = currentPiece; //落子
                            playAudio("fill.wav");
                            break;
                        }
                    } else {
                        System.out.println("非法落子，请重新选择落子位置");
                        playAudio("illegal.wav");
                    }
                } else {
                    System.out.println("非法落子，请重新选择落子位置");
                    playAudio("illegal.wav");
                    sc.next();//将Scanner中存储的数据取出来，防止死循环
                }
            }
            //落子完成后，棋盘需要重新展示
            showChessboard();
            //判断获胜情况，一共4种
            for(int i=0; i<chessboard.length; i++){
                for(int j=0; j<chessboard[i].length; j++){
                    //第一种：水平方向上存在同一玩家的连续5颗棋子
                    //(i, j) (i, j+1) (i, j+2) (i, j+3) (i, j+4)
                    boolean case1 = (j + 4 < chessboard[i].length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i][j+1] == currentPiece
                            && chessboard[i][j+2] == currentPiece
                            && chessboard[i][j+3] == currentPiece
                            && chessboard[i][j+4] == currentPiece;
                    //第二种：锤子方向上存在同一玩家的连续5颗棋子
                    //(i, j) (i+1, j) (i+2, j) (i+3, j) (i+4, j)
                    boolean case2 = (i + 4 < chessboard.length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i+1][j] == currentPiece
                            && chessboard[i+2][j] == currentPiece
                            && chessboard[i+3][j] == currentPiece
                            && chessboard[i+4][j] == currentPiece;
                    //第三种：135°角上存在同一玩家的连续5颗棋子
                    //(i, j) (i+1, j+1) (i+2, j+2) (i+3, j+3) (i+4, j+4)
                    boolean case3 = (i + 4 < chessboard.length)
                            && (j + 4 < chessboard[i].length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i+1][j+1] == currentPiece
                            && chessboard[i+2][j+2] == currentPiece
                            && chessboard[i+3][j+3] == currentPiece
                            && chessboard[i+4][j+4] == currentPiece;
                    //第四种：45°角上存在同一玩家的连续5颗棋子
                    //(i, j) (i-1, j+1) (i-2, j+2) (i-3, j+3) (i-4, j+4)
                    boolean case4 = (i > 4) && (j + 4 < chessboard[i].length)
                            && chessboard[i][j] == currentPiece
                            && chessboard[i-1][j+1] == currentPiece
```

```
                            && chessboard[i-2][j+2] == currentPiece
                            && chessboard[i-3][j+3] == currentPiece
                            && chessboard[i-4][j+4] == currentPiece;
                    if(case1 || case2 || case3 || case4){
                        System.out.println(times % 2 == 0 ? "玩家A获得胜利"
: "玩家B获得胜利");
                        playAudio("win.wav");
                        break outer;
                    }
                }
            }
            times++;
        }
        if(times == totalPosition){//说明棋盘已经用完还未分出胜负
            System.out.println("平均");
        }
    }

    public static void showChessboard(){
        System.out.println("   0     1     2     3     4     5     6     7
    8     9");
        for(int i=0; i<chessboard.length; i++){ //外层循环控制行
            System.out.print(i + "   ");
            for(int j=0; j<chessboard[i].length; j++){//内层循环控制列
                if(j == chessboard[i].length - 1){ //最后一列
                    System.out.print(chessboard[i][j]);
                } else {
                    System.out.print(chessboard[i][j] + separator);//使用
print打印，不换行
                }
            }
            System.out.println();
            if(i < chessboard.length -1){//排除最后一行
                System.out.println("    |     |     |     |     |     |
  |     |     |     |");
            }
        }
    }

    /**
     * 播放声音
     * @param fileName
     */
    public static void playAudio(String fileName){
        URL url = Gobang.class.getResource(fileName);
        AudioClip clip = Applet.newAudioClip(url);
        clip.play();
        try {
            Thread.sleep(50);
        } catch (InterruptedException e) {
        }
    }
}
```