# 作业一

1. 利用双重`for`循环把二维数组的所有元素都赋值为1，便于验证
    - 所有行和所有列中所有元素的累加和为256
    - 左上角到右下角和右上角到左下角所有元素的累加和为32
2. 利用双重`for`循环把二维数组的所有元素进行累加求和
3. 利用双重`for`循环把二维数组的对角线元素进行累加求和
    - 左上角到右下角：横纵坐标相等

    ```
    if(r == c){
      diagonalSum += arr[r][c];
    }
    ```

    - 右上角到左下角：横坐标加纵坐标等于数组的下标之和

    ```
    if(r + c == arr.length - 1){
      diagonalSum += arr[r][c];
    }
    ```

4. 运行结果截图：

```
/Library/Java/JavaVirtualMachines/jdk-11.0.8.jdk/Contents/Home/bin/java -javaagent:/Appli
所有元素之和：256
对角线所有元素之和：32

Process finished with exit code 0
```

# 作业二

1. 创建一个方法用来绘制绘制棋盘。这里可以借鉴模块一的作业。这里绘制出的棋盘是包含横纵坐标的，便于用户定位。

```
private static char[][] createChessBoard(int size){

}
```

    - 用户可以自定义棋盘大小，但必须要在合理的范围之间。方法规定棋盘的大小要在5x5～35x35之间。

```
if(size >= 5 && size <= 35){

}else{
    System.out.println("--------------------------");
    System.out.println("棋盘大小不合理，请重新输入！");
    System.out.println("棋盘大小应在5*5到35*35之间！");
}
```

- 绘制棋盘的逻辑与模块一的作业一样，可以参考借鉴。
- 注意⚠️：在游戏中规定棋盘大小为16x16。不需要用户自定义棋盘的大小。

2. 创建一个方法用来打印棋盘，用户每下一次棋，都要调用这个方法去打印更新棋盘

```
private static void printChessBoard(char[][] chessBoard){

}
```

- 利用`foreach`打印棋盘

```
if(chessBoard != null){
  for (char[] chars : chessBoard) {
    System.out.println(Arrays.toString(chars));
  }
}
```

```
[ , 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g]
[1, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[2, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[3, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[4, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[5, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[6, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[7, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[8, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[9, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[a, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[b, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[c, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[d, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[e, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[f, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[g, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
```

3. 创建一个方法用来加载游戏。这个方法是游戏的入口。主要用于显示游戏前的准备工作，提示用户准备开始。

```
private static void landingGame(){

}
```

   - 简单说明游戏规则

```
System.out.println("--------------------------");
System.out.println("欢迎来到五子棋大战！");
System.out.println("黑方为先手方，显示x；白方为后手方，显示o！");
```

   - 利用`while`循环检查用户是否准备开始游戏

```
boolean startFlag = false;
while(!startFlag){

}
```

     - 提示用户准备

```
System.out.println("--------------------------");
System.out.println("开始游戏请按Y确认：");
```

```
/Library/Java/JavaVirtualMachines/jdk-11.0.8.jdk/
--------------------------
欢迎来到五子棋大战!
黑方为先手方，显示X；白方为后手方，显示O!
--------------------------
开始游戏请按Y确认：
```

- 如果用户输入Y/y，创建棋盘，调用`playGame()`方法开始游戏；否则继续让用户准备

```
if(confirmInfo.equals("Y") || confirmInfo.equals("y")){
  startFlag = true;
  char[][] chessBoard = createChessBoard( 16);
  playGame(chessBoard);
}else{
   System.out.println("准备中......");
}
```

```
开始游戏请按Y确认:
Y
-------------------------
游戏开始！黑方先下！
[ , 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g]
[1, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[2, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[3, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[4, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[5, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[6, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[7, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[8, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[9, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[a, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[b, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[c, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[d, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[e, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[f, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[g, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
-------------------------
请黑方输入横纵坐标:
```

```
--------------------------
开始游戏请按Y确认:
N
准备中......
--------------------------
开始游戏请按Y确认:
```

4. 创建一个方法开始游戏。这个方法是主要用于流程控制，让黑白双方轮流下棋，直到一方胜出为止。

```
private static void playGame(){

}
```

   - 首先提示黑方先下

```
System.out.println("--------------------------");
System.out.println("游戏开始！黑方先下！");
```

- 定义5个变量用来记录状态信息

```
//表示用户下的棋的横纵坐标。
int x = 0;
int y = 0;
//表示用户下的棋是否有效。
boolean isValidFlag;
//表示黑白双方，false代表黑方，true代表白方。
boolean playerFlag = false;
//表示比赛输赢结果，false代表输，true代表赢。
boolean winFlag = false;
```

- 利用`while`循环检查输赢结果，一方获胜，游戏结束；否则游戏继续。

```
while(!winFlag){

}
```

  - 黑白双方轮流下棋

```
if(!playerFlag){

}else{

}
```

    - 调用`isValidAndAssign()`方法判断棋子是否有效

```
while(!isValidFlag){
  ...
  if(isValidAndAssign(x, y, chessBoard, false)){
    isValidFlag = true;
  }
}
```

  - 用户每下一次，棋盘更新

```
printChessBoard(chessBoard);
```

  - 并调用`isWin()`方法进行输赢的判断

```
if(isWin(x, y, chessBoard, playerFlag)){
  winFlag = true;
}
```

- 最后调换选手

```
playerFlag = !playerFlag;
```

5. 创建一个方法判断棋子是否有效并赋值

```
 private static boolean isValidAndAssign(int x, int y, char[][] chessBoard,
boolean playerFlag){

 }
```

- 棋子的横纵坐标应在1～16之间

```
if(x <= 0 || x > 16 || y <= 0 || y > 16){

}
```

```
[ , 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g]
[1, X, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[2, +, O, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[3, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[4, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[5, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[6, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[7, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[8, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[9, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[a, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[b, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[c, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[d, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[e, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[f, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[g, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
--------------------------
请黑方输入横纵坐标:
17 17
--------------------------
横纵坐标应在1～16之间，请输入有效坐标!
请黑方输入横纵坐标:
```

- 应被占用的棋子，不能重复占用

```
else if(chessBoard[x][y] == 79 || chessBoard[x][y] == 88){

}
```

```
请黑方输入横纵坐标:
1 1
[ , 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g]
[1, X, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[2, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[3, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[4, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[5, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[6, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[7, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[8, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[9, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[a, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[b, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[c, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[d, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[e, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[f, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[g, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
--------------------------
请白方输入横纵坐标:
1 1
--------------------------
该坐标已被占用，请输入有效坐标!
请白方输入横纵坐标:
```

- 最后对有效的坐标赋值

```
else{
  if(!playerFlag){
    chessBoard[x][y] = 88;
  }else{
    chessBoard[x][y] = 79;
  }
}
```

6. 创建一个方法判断输赢。每下一次棋，都会调用这个方法。

```
 private static boolean isWin(int x, int y, char[][] chessboard, boolean
playerFlag){

 }
```

- 根据黑方或白方判断输赢情况

```
if(!playerFlag){

}else{

}
```

  - 根据坐标分别调用`sum()`方法计算8个方向的和，若其中有一个满足要求，即退出循环

```
for(int i = 1; i <= 8; i++){
  if(sum(x, y, chessboard, i) == 440){
    ...
    return true;
   }
}

for(int i = 1; i <= 8; i++){
  if(sum(x, y, chessboard, i) == 395){
    ...
    return true;
  }
}
```

7. 创建一个方法计算以一个坐标为起点，一个方向的元素之和

```
private static int sum(int x, int y,  char[][] chessboard, int direction){
  int sum = 0;
  int updatedX = x;
  int updatedY = y;
}
```

- 上下方向：横坐标加/减一，纵坐标不变

```
//上
updatedX = x - i;
//下
updatedX = x + i;
```

- 左右方向：横坐标不变，纵坐标加/减一；

```
//左
updatedY = y - i;
//右
updatedY = y + i;
```

- 左上右下方向：横纵坐标加/减一；

```
//左上
updatedX = x - i;
updatedY = y - i;
//右下
updatedX = x + i;
updatedY = y + i;
```

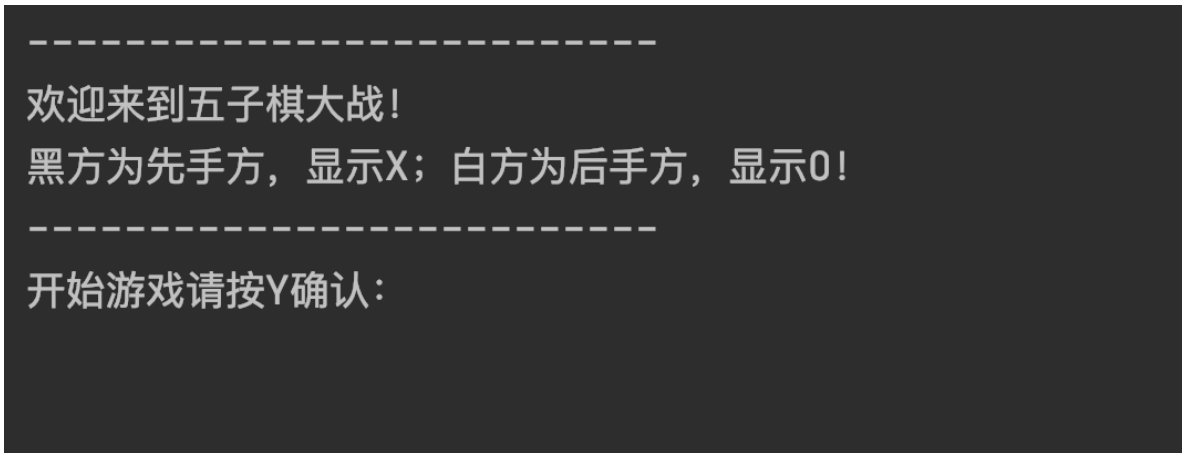
- 右上方向：横坐标减一，纵坐标加一；

```
//右上
updatedX = x - i;
updatedY = y + i;
```

- 左下方向：横坐标加一，纵坐标减一；

```
//左下
updatedX = x + i;
updatedY = y - i;
```

8. 运行代码截图
   - 准备阶段




   - 按Y/y开始游戏

```
黑方为先手方，显示X；白方为后手方，显示O！
--------------------------
开始游戏请按Y确认：
Y
--------------------------
游戏开始！黑方先下！
[ , 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g]
[1, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[2, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[3, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[4, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[5, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[6, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[7, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[8, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[9, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[a, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[b, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[c, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[d, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[e, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[f, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[g, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
--------------------------
请黑方输入横纵坐标：
```

- 黑方输入坐标

```
请黑方输入横纵坐标:
2 2
[ , 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g]
[1, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[2, +, X, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[3, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[4, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[5, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[6, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[7, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[8, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[9, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[a, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[b, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[c, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[d, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[e, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[f, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[g, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
```

- 白方输入坐标

```
请白方输入横纵坐标:
3 2
[ , 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g]
[1, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[2, +, X, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[3, +, O, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[4, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[5, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[6, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[7, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[8, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[9, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[a, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[b, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[c, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[d, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[e, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[f, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[g, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
```

- 黑白双方依次轮流下棋，直到一方获胜

```
请黑方输入横纵坐标:
2 6
[ , 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g]
[1, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[2, +, X, X, X, X, X, +, +, +, +, +, +, +, +, +, +]
[3, +, O, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[4, +, O, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[5, +, O, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[6, +, O, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[7, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[8, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[9, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[a, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[b, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[c, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[d, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[e, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[f, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
[g, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +, +]
恭喜黑方获胜!
```

## 作业三

1. 创建抽象类提取上网类和电话类的共有成员。

```
public abstract class Plan {
   ...
}
```

2. 创建电话服务接口

```
public interface TalkService {
    void talking(double talk, SimCard simCard);
}
```

3. 创建上网服务接口

```
public interface DataService {
    void surfing(double data, SimCard simCard);
}
```

4. 创建电话类继承抽象类实现电话服务接口，重写show()和talking()方法

```
public class TalkPlan extends Plan implements TalkService {
  ...
  @Override
  public void show() {
    System.out.println("[通话时长：" + super.getAmount() + "，短信条数：" +
this.textNumber + "，每月资费：" +       super.getFee() + "]");
  }

  @Override
  public void talking(double talk, SimCard simCard) {
    System.out.println(simCard.getUsername() + "有"+talk+"分钟通话时长。他正在打电
话...");
  }
}
```

5. 创建上网类继承抽象类实现上网服务接口，重写show()和surfing()方法

```
public class DataPlan extends Plan implements DataService {
  ...
  @Override
  public void show() {
    System.out.println("[上网流量：" + super.getAmount() + "，每月资费：  " +
super.getFee() + "]");
  }

  @Override
  public void surfing(double talk, SimCard simCard) {
    System.out.println(simCard.getUsername() + "有"+data+"GB流量。他正在上
网...");
  }
}
```

6. 创建电话卡类
7. 创建电话卡的枚举类
8. 创建用户消费类
9. 创建测试类

```
public static void main(String[] args){
  SimCard card = new SimCard(
                            SimCardEnum.BIGCARD.getCardType(),
                            "*******9999",
                            "Zichen",
```

```
                                "*******",
                                1000.00,
                                900.00,
                                10.00
                            );

    TalkPlan talkPlan = new TalkPlan(card.getTalkLimit(), 100, 50.00);
    DataPlan dataPlan = new DataPlan(card.getDataLimit(), 10.00);
    talkPlan.show();
    dataPlan.show();
    talkPlan.talking(card.getTalkLimit(), card);
    dataPlan.surfing(card.getDataLimit(), card);
}
```

10. 运行代码截图：

```
/Library/Java/JavaVirtualMachines/jdk-11.0.8.jdk/Contents/Home/bin/ja
[通话时长：900.0，短信条数：100，每月资费：50.0]
[上网流量：10.0，每月资费：  10.0]
Zichen有900.0分钟通话时长。他正在打电话...
Zichen有10.0GB流量。他正在上网...
```