**A 程序设计思路**

1. 读取地图并打印
2. 创建找到动物位置的函数并使得动物移动
3. 动物互吃的判断
4. 动物进出小河的判断
5. 胜负条件的判断
6. 悔棋，反悔棋功能的实现
7. 错误指令等细节的处理
8. 不断地运行 测试以及Debug

**B 大体实现过程**

1. 先从助教给的txt文件中读取动物和地图数据，并将其赋值到数组中。
2. 通过读取数组并用switch语句来打印出地图。
3. 读入指令，并根据打入的指令执行相应的操作，若指令不正确弹出相应提示，指令正确通过各种方法正确执行相应操作（undo，redo等或者让动物移动或提示这种做法违规）
4. 打印地图，进入新的一层循环。

**C 内容**

1. 先读取文件并将动物与地形信息录入二维数组map与animalData（我将左方的动物改成了abcd）。
2. 使用printTheMap(animalData, map)方法打印地图。
3. 进入大循环，String类型的input用于储存用户的指令。
4. 创建int类型的line与row变量，line为动物的行坐标，row为列坐标。

（theAnimal为指令的第一个字母，用于findPosition方法）

1. 如果指令为help，输出指令介绍

如果指令为exit，退出程序

如果指令为restart，重新开始，将animalData数组还原为memory[0]所对应的数组。

如果指令为undo或redo，进行悔棋与反悔棋（之后会详细介绍）

如果该指令错误，输出不能识别指令。

以上过程结束后除exit外，都将重新进行主循环（continue）

该指令为动物移动指令时，用switch判断要走的方向，对于每个方向大体过程如下：

先判断是否越界（即数组是否越界）

使用judgeWater方法判断要进入水中的动物是否可以入水。

若为狮虎跳河，则判断狮虎在跳河时是否有对面的鼠拦路，以及再跳河时对面的动物是否是对方的以及是否能吃，都没有问题，则狮虎可以跳河:

若不为狮虎跳河，则使用judgeTheMove方法判断是否违规，若违规，打出违规原因，否则成功走棋。

若指令不对应任何方向，打出提示。

成功走棋后打印棋盘，用whetherWin方法判断是否胜利，胜利结束游戏，否则更换玩家，进入下一层循环。

**悔棋与反悔棋的实现：**

首先设置了一个三维数组memory来储存历史信息，两个重要变量是currentTime和toProtectRedo(均为int类型)，在正常行走时，前者不断自增，后者一直为零。

当undo操作下达后，首先判断是否已经退回至开头（currentTime-1是否为零），没有的话就将memory[currentTime - 1]的内容复制到animalData中，然后currentTime自减，toProtectRedo自增，打印棋盘，功能实现。

当redo操作下达后，通过toProtectRedo的值（是否为0）来判断该操作的上一步是否为悔棋，因为只有上一步悔了棋，redo才有意义，之后将memory[currentTime + 1]复制到animalData中，打印地图，toProtectRedo自减。

**方法介绍：**

String findPosition(char theAnimal, boolean player, char[][] animal)

此方法是用来在输入指令后来找到所要移动的动物所对应的坐标的，但因为左方的动物在数组中是字母，所以要用theAnimal = (char) (theAnimal - '1' + 'a');，右方可以不直接转化，再通过animal数组都进行扫描，找到相应动物，将横纵坐标一起储存在字符串str中返回（如6和1返回61），找不到动物，返回can’t find

void printTheMap(char[][] animalData, char[][] map)

此方法专门用于打印地图，因此无返回值类型，输入动物与地形地图，开始两层循环，若这个格子上有动物，则打印动物，没有动物则打印地形。（具体通过switch实现）

int judgeTheMove(char yourOwnAnimal, char theAnimalNextCell, char theTerrainNextCell, char yourOwnTerrain, boolean player)

此方法主要用于判断动物间的互吃关系

拿对于左方举例，先判断该动物的行为是否是水中鼠要吃大象，如果是，返回5

之后判断要进的格子是否为自己的窝，如果是，则返回2，再判断要吃的棋子是否为自己的棋子，如果是，返回3，之后，若下一格有动物，判断他是否在陷阱里，是的话直接吃掉返回1，不在陷阱中，判断此动物是否可吃（象与鼠单独写出），可吃返回1，不可吃返回4

PS 违规操作的提示都在主方法中。

int judgeWater(int line, int row, char direction, char[][] animalData, char[][] map, boolean player)

此方法用于判断有关小河方面的规则，这里我依然采用分方向解决的办法，开始是有一个判断句，说明一旦要移动的是鼠，这个方法永远只输出0，如果要移动的棋子不是鼠，则分为四个方向，拿w方向举例第一个判断是要判断进入的格子是否为水，不是返回0，是的话接着判断是否要移动的是狮虎，不是的话返回1，禁止跳河，不是则判断之间是否有敌方鼠挡路（所以开始的时候要以player作为参数之一），有鼠挡路返回3，无鼠返回2，在主界面中执行跳河操作。

int whetherWin(char[][] animal, char[][] map)

此方法是主要的判断胜负的方法，因为之前在judgeTheMove函数中已经确定了动物不能进入自己的窝，所以这里只需判断在相应窝的位置是否有动物即可。之后的两个胜利条件被我分装在3个其他方法里。

int whetherNoChess(char[][] animal)

这个方法用于判断是否有一方的动物被吃光了,用三层循环扫描动物，第一层是从鼠到象的循环，第二三层是对于地图的循环，如果发现了要找的动物，则使得相应变量+1，若有一方等于零，则游戏结束。（重要变量int theAliveAnimalOfRightPlayer , theAliveAnimalOfLeftPlayer 分别指右方和左方活着的动物的数量）

int whetherCanIMove(char[][] animal, char[][] map)

int toJudgeWhetherCanIMove(int line, int row, char[][] animal, char[][] map, boolean player)

这两个方法都是用来判断无子可动的方法（第二个是第一个的一部分）

定义int类型的重要变量theAnimalAliveOfRight , theAnimalAliveOfLeft , theNumberOfLeftAnimalsToMove , theNumberOfRightAnimalsToMove来表示不能动的动物数和活着的动物数，然后用和与whetherNoChess类似的方法算出存活的动物数，之后，在toJudgeWhetherCanIMove方法中向上下左右移动每一个棋子（其中调用了第三个和第四个方法），若上下左右都不可移动，则输出1，否则输出0，这样在whetherCanIMove的循环中就能够算出可移动动物的数量，有一方可移动的动物数与存活的数目相等，则返回1，有一方获胜。

**D问题**

1. 读取地图后难以区分开动物是左方的还是右方的。
2. 字母与数字间不能灵活的转换。
3. 单引号 双引号和没有引号的区分。（之前没有多加注意，运行的时候经常出错还找不到错在哪里）
4. 当用数组返回动物的坐标时，不知道如何返回两个值。
5. 悔棋与反悔棋不是很会弄~
6. 单个方法过长（尤其是判断胜负的方法，大约有150行）
7. 各种奇葩小错误的出现（比如在各个方向代码复制粘贴的过程中忘掉了一块，导致在走某些方向的时候出现bug）
8. 一开始的时候，错误提示（比如 “不能进入水中”）的打印在方法中，这样在进行多个判断时由于要多次使用这个方法，所以有时会打出两遍提示。
9. 在早期的时候对规则的理解不是很清楚，到后期发现再去更改代码发现很麻烦。
10. 无论如何，PJ的代码中总要分出4个方向，每个都有很多重复代码，想精简些。
11. 表达能力拙计。在注释 变量命名 方法名等方面感觉找不到精炼的话来表达自己的意思。

**E 解决策略**

1. 自己思考了下，最终决定把源文件左方的动物改成字母。
2. 自己查书，恶补了一下以前的知识，又通过百度学会了像char1-‘0’这样比较机智的办法。
3. 这个问题主要还是问同学解决的，知道了对于char string int 等类型引号的用法，棋子也就终于能动了！
4. 并没有想到助教说的返回数组的办法，但是考虑既然坐标是两个个位数，所以最后直接就返回了一个两位数（比如6和1就是61），后来又改成了返回一个String。
5. 通过lab7知道了如何悔棋与反悔棋，把井字棋的方法更改了一下用到了斗兽棋中。
6. 最后决定把单个方法拆开，比如判断胜负我直接就拆成了四个方法，方法间来回调用使得代码看起来顺眼了许多~但是有些方法需要5个甚至6个参数，看起来很冗赘，这个问题我到现在还没有解决。
7. 主要是通过比较出现bug方向的代码和其他方向的代码来debug，同时叮嘱一下自己下次细心点……
8. 开始的时候，我是将各种if语句嵌套了起来，但这样会使代码逻辑变得复杂，并且做无子可动时也会十分麻烦，所以我干脆把错误提示移到了主方法中。
9. 只能埋头修改啦~但修改之后的代码再修改的话会变得很容易，也掌握了一些让代码可塑性更高的方法。
10. 在判断动物大小等方法中删去了方向，但很多其他地方还是没有想出办法来去除方向方面的重复代码，我问的其他同学也没有什么好的办法，所有这个问题并没有完全解决。
11. 在code review中学到了一些技巧，也希望自己能通过多练多看来提高这种技能。

**F 反思与不足**

1. 觉得自己的代码在有些地方重复较多，比如在主方法和一些其他方法中要分四个方向判断指令，导致重复代码很多。
2. 在悔棋和反悔棋功能时有新创建了一个三维数组，觉得这么做有些浪费内存。
3. 有些变量名依然不是很直观，有些方法较长，需要的参数很多。
4. 注释写的不是很好。

**G 觉得自己写得好的地方**

1. 悔棋和反悔棋的我觉得实现的比较清晰
2. 代码比较容易修改，各个if语句的层次很分明
3. 方法间的调用很灵活~