**斗兽棋设计文档**

1. 关于读入与初始化地图的方式

我定义了几个数组：animalMap, tileMap, printMap, alAnimalMap

首先使用Scanner逐行输入地图文件，用substring方法进行拆分，分别存入两个二维数组中。将左侧动物值加上10存入alAnimalMap对应位置，右侧动物值加上20。然后再用选择语句和循环语句遍历判断此处是否有动物，若有，则将动物存入打印地图printMap（左方动物数字打印于右侧，右方动物数字打印于左侧），若无，则将地形存入打印地图。

1. 关于类的使用

在本程序的实现过程中，我使用了两个类。Jungle为带有main函数的主类，Animal为作为数据类型的动物类，用于定义与操作动物对象。

Jungle类：主类，作为程序运行的主场地。拥有当前阵营，地图与历史地图,当前步数等类变量以及保存地图，更新地图，游戏主体等方法。设计思路是将程序主体模块化和封装化，使其逻辑清晰，运行流畅。

Animal类：动物类，定义一种动物数据类型。拥有值，位置，阵营，生存状况，是否在水等属性，各类geter和seter，三类移动（普通动物，鼠，狮虎）与相应的对抗方法。设计思路是将零散的动物属性封装聚集起来，使其更易操作与变动，逻辑更加明晰。

1. 使用到的关键变量

Jungle类中：

int[][] animalMap 存放动物地图

ArrayList[][] preAnimalMap 存放历史动物地图

int[][] alAnimalMap 存放全信息动物地图（包含动物的值以及阵营）

int[][] tileMap 地形地图

int nowCamp 现在准备移动的阵营（用1代表左阵营，用2代表右阵营）

int leftDead 左方的动物死亡数

int rightDead 右方的动物死亡数

int leftTrap 左方被困死动物数

int rightTrap 右方被困死动物数

int undoStep 悔棋的步数

int step 总步数

boolean isFirst 是否是开局

boolean isMove 是否发生实质性移动

boolean isPlay 是否继续玩下去

String[][] printMap 存放用于打印的地图

Animal[] leftcamp 左阵营的动物

Animal[] rightcamp 右阵营的动物

Animal类中：

private int value 存储动物的值

private int locationR 存储动物所在的行数

private int locationC 存储动物所在的列数

private boolean isAlive 存储动物的存活情况

private int camp 存储动物的阵营

private boolean isWater 存储动物的在水情况

1. 主要函数

Jungle类中:

boolean play(Scanner input)

游戏主体玩耍方法。在其中我调整了isMove变量的值，用于判断此次命令是否进行了实质性移动。调整了leftDead, rightDead, leftTrap, rightTrap的值，每轮游戏都要重新计数一次左右死亡与被困数量。调整了nowCamp的值，交换了两方阵营。返回了isPlay的值，判断是否还玩。

void saveMap()

将地图存入历史地图方法。在ArrayList对象数组preAnimalMap中多加了一张地图。

void updateMap()

更新地图，把位置属性为地图上对应坐标的动物值存入地图对应位置。更新了alAnimalMap与animalMap的值。

void putAnimal()

更新动物，把动物对象的位置属性设置成动物地图上指示的对应位置值。更改了leftCamp与rightCamp各只动物的位置属性。

void preMap()

将历史地图中对应step-undoStep的地图调出来，存入现在的地图中。更改了alAnimalMap与animalMap的值。

void setAlValue()

初始化动物对象以及历史动物地图。初始化了所有关键变量（好多啊，我就不一一赘述了）。

void readMap()

从文件里读取地图存入数组。初始化了tileMap，animalMap，alAnimalMap。

void initAnimal()

初始化动物位置。初始化了leftCamp和rightCamp各动物对象的位置属性。

void undo()

悔棋方法。调整了isFirst的值（也许会回到开头），调整了undoStep的值，交换了阵营（nowCamp）。

void redo()

撤销悔棋方法。调整了undoStep的值，交换了阵营（nowCamp）。

boolean camping(String str, Animal[] camp)

某方动物进行移动的方法。调整了step和undoStep的值，调整且返回了isMove的值。

Animal findAnimal(int lor, int loc)

在已知坐标的状况下返回那个坐标位置的动物对象

void transPrint()

生成一张用于打印的地图。给printMap赋值。

void mapPrint()

打印用于打印的地图

void help()

打印帮助

Animal类中：

移动方法

boolean move(String direction)

普通动物的移动方法。

boolean ratMove(String direction)

老鼠的移动方法

boolean tiliMove(String direction)

老虎狮子的移动方法

单方向移动能力判断方法

boolean subMovable(int r, int c)

判断普通动物某个方向是否能动

boolean subRatMovable(int r, int c)

判断老鼠某个方向是否能动

boolean subTiliMovable(int r, int c, int r2, int c2)

判断狮虎某个方向是否能动

全方位移动能力判断方法

boolean isMovable()

判断普通动物是否能动

boolean isRatMovable()

判断老鼠是否能动

boolean isTiliMovable()

判断狮虎是否能动

行走与对抗方法

boolean fight(int r, int c)

普通动物行走以及与其他动物对抗。调整了locationC和locationR，isAlive的值，判断其位置与存活情况。

boolean ratFight(int r, int c)

老鼠行走以及与其他动物对抗。调整了locationC和locationR，isAlive，isWater的值，判断其位置与存活情况以及在水情况。

boolean tiliFight(int r, int c, int r2, int c2)

狮虎行走以及与其他动物对抗。调整了locationC和locationR，isAlive的值，判断其位置与存活情况。（局部变量中有一个是hasRat，用于判别跳河时水路上是否有敌方老鼠）

1. 程序优劣分析

优势：使用了面向对象编程的思想与技术，定义了动物类封装了属性与方法，程序代码更加清晰一些。将各类重复代码进行了有效整合，用方法的多次调用来实现（如动物的移动，我将向各个方向行走的特征值作为参数传入，使这一部分代码减为原先的四分之一）。使用updateMap与putAnimal这样一组共轭对应的方法实现地图与动物对象位置的互相转换，使用findAnimal方法可以通过位置坐标找到对应位置的动物，如此之后，位置，地图与动物对象建立了对应的联系，使处理方便了很多。

劣势：使用面向对象编程使单个个体易于处理的同时，也增加了代码的长度，使代码变得不必要的繁杂，程序的精简度发生了较大损耗。由于编程前期整体编程思路的不明晰，我使用了一个不加阵营信息的animalMap，之后证明这是不必要的（可以用alAnimalMap或是preAnimalMap的第一张进行替代），减弱了代码的统一性。过于大量的在主类中使用类变量（有意义却意义不大的那种），以后要尽量避免这种不好的编程习惯，尽量用方法间传值来替代类变量的滥用。

番外：一路上的问题们和解决策略

1. 呀我用的是对象啊，死了的动物位置该放在那里呢？

答：这个问题轻松地解决了，我把它扔在了地图之外。行数和列数都是-2的地方。

1. 关于各类移动函数中多段类似代码的精简

答：由于移动是有四个方向的，所以不精简前相似的代码需要打上四遍。然而由于四段代码有较大的差异，整合起来也比较麻烦。不过想到代码的不同点至少该比共同点少许多，于是我决定创建一个方法，将代码的不同点都作为参数传入方法，如此就能完成整合。对于普通动物与老鼠而言，传入的应该是它们将要到达的坐标的表达式，而对于狮虎而言，还要传入它们跳河时将要到达的坐标的表达式。这个问题也算是解决了。

1. 无限步悔棋的实现

答：关于这个悔棋，TA给了我三种思路，其余两种是类似于记录变化的方式。于我而言挺难的，于是我使用了最为暴力的方式：将历史地图全然存入数组。但是对于无限步悔棋（至少是理论上的），我需要一个不定长数组。于是通过搜索引擎，我发现了ArrayList这个类，一个加强版的数组。通过这个历史地图数组，理论上的无尽悔棋得以实现。

1. 悔棋引出的地图记录信息缺失问题解决

答：我新建了一张地图，使每一个元素既包含值信息也包含阵营信息。如左方老鼠我用11来表示，而右方老鼠则是21来表示。这张地图是信息缺失得以解决。（至于为什么我到悔棋才遇到这样的问题，可能是因为我用了自定义类的缘故吧(￣旦￣；)）

1. 关于已知位置处动物对象寻找的问题解决

答：我创建了一个findAnimal的方法。然后就是通过遍历的方法，一个个去找寻位置坐标对应的动物，然后返回那只动物（此时我发现原来动物也是可以当返回值用的，真棒）。

1. 关于斗兽棋中两类异常胜负判断的实现

答：我说的异常胜负，就是那种吃光或者困死对方动物的胜利方式。我使用了在每轮开始时清点的方式。通过一个foreach循环遍历每一只动物，判断其是否已死或被困死。判断是否已死很简单，每一只动物都有生存情况的属性，简单的改变即可。而判断是否被困死就复杂一些了。我先判断了动物往某一方向的可动情况，通过把所有可移动条件取非一遍就可以得到不可移动条件，而只要遇到一个不可移动条件，动物往这一方向就不可动。然后把四个方向的可动性都判断一遍，若都不可动，则已被困死。如此就可以实现被困死的判断。好啦，这个问题也解决了，而我的文档也终于写到这里了，长达一个月的PJ任务终于大功告成了，想想还有些不舍呢(\*/ω╲\*)

（哦对了接下来就是PJ2……啊啊啊啊）