PJ1 说明文档

193020100xx XXX

程序结构设计与分析

1. **地图的尺寸**由常数 FLOORS，ROWS，COLUMNS 确定，其他信息从 map 文件中读入。map 文件中包含玩家初始位置和地图。读入的值用双引号（””）标记，按顺序读入，给相应变量赋值。不在双引号内的数据将被地图读取忽略，它们只是为了便于在文本编辑器中查看和编辑地图。地图仅在游戏初始化时读入一次。

   存档为一个 save 文件（若有），它的结构与读取方式和 map 文件基本类似，不同的是存档在每次读档时都从硬盘中读入。

   怪物数据记录在类 Enemy 中。

2. 类的说明

   MagicTower：包含 main()的类。

   Colour：提供带颜色的打印。

   Enemy：根据怪物的 id 创建相应的 Enemy 实例，包含与怪物战斗时需要的怪物信息。

   Help：打印帮助文本。

   Player：Player 对象可储存玩家的所有信息，用于记录每一步操作后玩家的状态。

   Record：记录玩家状态和地图中被清除的元素的变化，用于撤销和重做。如果玩家的操作没有改变地图内容，则用.elementZ = -1 标记。

3. 关键变量说明

```
private static final int FLOORS = 5, ROWS = 13, COLUMNS = 13;
```

地图的尺寸。

```
private static final int LEN_RECORDS = 256;
```

历史记录的单位长度。

```
private static char[][][] mapData = new char[FLOORS][ROWS][COLUMNS];
```

游戏初始地图，只读入一次。

```
private static char[][][] map = new char[FLOORS][ROWS][COLUMNS];
```

当前地图。

```
private static int startX, startY;
```

玩家的初始位置。规定玩家只能从序章（即第 0 层）开始，所以没有 startZ 变量。

```
private static Player player;
```

玩家对象。

```
private static Record[] records;
```
历史记录数组，每进行一步操作将添加一个 Record 对象。

```
private static int recordPosition, originalRecordPosition;
```
recordPosition 指向下一个记录将要写入的位置。originalRecordPosition 指向该次撤销、重做操作前下一个记录将要写入的位置，即重做操作最多可达到的位置的下一个。

```
private static boolean trap = false;
```
玩家在进入魔王区域后将触发机关，打败魔王解除机关。记录机关的状态。

4. 主要方法列举

```
private static void info()
```
对应 F 指令，列表打印当前楼层怪物的参数。如果当前楼层已没有怪物，给出提示。

```
private static void help()
```
对应 HELP 指令，打印帮助文本，并显示地图和玩家状态。

```
private static void load() throws IOException
```
对应 LOAD 指令，读取存档，并显示地图和玩家状态。

```
private static void save() throws IOException
```
对应 SAVE 指令，保存进度，并提示保存成功。

```
private static void restart()
```
对应 RESTART 指令，开始新游戏，打印游戏介绍，并显示地图和玩家状态。

```
private static void exit() throws IOException
```
对应 EXIT 指令，自动保存进度，再退出游戏。如果玩家不希望保存，则可以直接关闭游戏程序，而不使用此指令。

```
private static void undo()
```
对应 UNDO 指令，撤销一步。没有撤销步数上限，除非已到达游戏开始、重新开始或读取进度的位置。

```
private static void redo()
```
对应 REDO 指令，取消撤销。

```
private static boolean parseWASD(String command) throws IOException
```
对应所有的移动操作。包括基本移动 W、A、S、D，以及两种快速移动方式：WASD 后加一位数字，表示连续多次向该方向移动；或者 WW、AA、SS、DD，表示向该方向走到底。为防止误操作，增强控制，快速移动只能在空地上连续移动，若遇到楼梯会在新的楼层的楼梯处停下，若遇到墙、物品、怪物会在它们之前停下。只有玩家第一步操作就触发以上事件时才会执行相应的操作，且完成后将停下。
所有不属于前 8 个指令的输入都通过 parseWASD()检查，如果输入不正确，该方法返回 false。

5. 自我评价

写得好的地方：增强移动操作等于长按方向键，使玩家可以关注游戏本身，而不是单调的操作。

写得不好的地方：每条记录只能包含一个地图元素的清除，如果要扩展为地图元素的改变就难以在此基础上实现。而且即使是本游戏中，魔王区域的机关就另外用了 trap 来表示，因为如果把机关门视为一种地图元素，那么玩家打败魔王时就要同时清除魔王和机关门，无法实现。trap 分布于多个方法，使得代码有一点混乱。

编程中遇到的问题和解决策略

1.没有学过带颜色的打印，API 和书上也都没有写，最后上网查找到了

2.历史记录的调试很困难，第一个 bug 是把 player 引用直接赋值给 Record 对象的 player 属性，第二个是 MagicTower 的第 528 行，如果撤销过之后再把记录覆盖上去，那么要写入的位置就一定不是 null，造成记录填写不正确，这两个 bug 都找了好几个小时

3.本来 parseWASD 是想要能解析任意多步移动操作的，后来发现实际使用不会用到，而且有一点难实现，改为了实用且容易写的单步操作

意见与建议

希望以后的 lab 批改完之后可以给出得分，简要说明一下代码的问题，便于我们及时改正。特别是代码风格这种电脑不认为是问题的问题，如果没有作业反馈就很难改。