**深 圳 大 学 实 验 报 告**

**课程名称： Java程序设计**

**实验项目名称： 必实验2 类的高级应用**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 蔡树彬**

**报告人： 岳海涛 学号： 2022152020**

**实验时间： 2023年9月26日- 10月10日**

**实验报告提交时间： 2023年10月16日**

**教务部制**

|  |
| --- |
| **实验目的与要求：**  **实验目的：**熟悉面向对象编程中类的编写及应用。  **实验要求：**  编写代码，解决下述“炉石传说”问题（40分），并且要求类设计良好（10分）、代码提交到Git上（10分），通过代码扫描（10分），使用JUL或Log4j进行日志输出（10分），完成实验报告撰写，可读性好（20分）。  **问题描述**  《炉石传说：魔兽英雄传》（Hearthstone: Heroes of Warcraft，简称炉石传说）是暴雪娱乐开发的一款集换式卡牌游戏。游戏在一个战斗棋盘上进行，由两名玩家轮流进行操作，本题所使用的炉石传说游戏的简化规则如下：  \* 玩家会控制一些角色(Role)，每个角色有自己的生命值和攻击力。当生命值小于等于 0 时，该角色死亡。角色分为英雄和随从。  \* 玩家各控制一个英雄(Hero)，游戏开始时，英雄的生命值为 30，攻击力为 0。当英雄死亡时，游戏结束，英雄未死亡的一方获胜。  \* 玩家可在游戏过程中召唤随从(Servant)。棋盘上每方都有 7 个可用于放置随从的空位，从左到右一字排开，被称为战场。当随从死亡时，它将被从战场上移除。  \* 游戏开始后，两位玩家轮流进行操作（Command），每个玩家的连续一组操作称为一个回合(Turn)。  \* 每个回合中，当前玩家可进行零个或者多个以下操作：  1) 召唤随从：玩家召唤一个随从进入战场，随从具有指定的生命值和攻击力。  　　2) 随从攻击：玩家控制自己的某个随从攻击对手的英雄或者某个随从。  　　3) 结束回合：玩家声明自己的当前回合结束，游戏将进入对手的回合。该操作一定是一个回合的最后一个操作。  \* 当随从攻击时，攻击方和被攻击方会同时对彼此造成等同于自己攻击力的伤害。受到伤害的角色的生命值将会减少，数值等同于受到的伤害。例如，随从 X 的生命值为 HX、攻击力为 AX，随从 Y 的生命值为 HY、攻击力为 AY，如果随从 X 攻击随从 Y，则攻击发生后随从 X 的生命值变为 HX - AY，随从 Y 的生命值变为 HY - AX。攻击发生后，角色的生命值可以为负数。  本题将给出一个游戏的过程，要求编写程序模拟该游戏过程并输出最后的局面。  **输入格式**  输入第一行是一个整数 n，表示操作的个数。接下来 n 行，每行描述一个操作，格式如下：  　　<action> <arg1> <arg2> ...  其中<action>表示操作类型，是一个字符串，共有 3 种：summon表示召唤随从，attack表示随从攻击，end表示结束回合。这 3 种操作的具体格式如下：  \* summon <position> <attack> <health>：当前玩家在位置<position>召唤一个生命值为<health>、攻击力为<attack>的随从。其中<position>是一个 1 到 7 的整数，表示召唤的随从出现在战场上的位置，原来该位置及右边的随从都将顺次向右移动一位。  \* attack <attacker> <defender>：当前玩家的角色<attacker>攻击对方的角色 <defender>。<attacker>是 1 到 7 的整数，表示发起攻击的本方随从编号，<defender>是 0 到 7 的整数，表示被攻击的对方角色，0 表示攻击对方英雄，1 到 7 表示攻击对方随从的编号。  \* end：当前玩家结束本回合。  注意：随从的编号会随着游戏的进程发生变化，当召唤一个随从时，玩家指定召唤该随从放入战场的位置，此时，原来该位置及右边的所有随从编号都会增加 1。而当一个随从死亡时，它右边的所有随从编号都会减少 1。任意时刻，战场上的随从总是从1开始连续编号。  **输出格式**  输出共 5 行。  第 1 行包含一个整数，表示这 n 次操作后（以下称为 T 时刻）游戏的胜负结果，1 表示先手玩家获胜，-1 表示后手玩家获胜，0 表示游戏尚未结束，还没有人获胜。  第 2 行包含一个整数，表示 T 时刻先手玩家的英雄的生命值。  第 3 行包含若干个整数，第一个整数 p 表示 T 时刻先手玩家在战场上存活的随从个数，之后 p 个整数，分别表示这些随从在 T 时刻的生命值（按照从左往右的顺序）。  第 4 行和第 5 行与第 2 行和第 3 行类似，只是将玩家从先手玩家换为后手玩家。  **样例输入**  8  summon 1 3 6  summon 2 4 2  end  summon 1 4 5  summon 1 2 1  attack 1 2  end  attack 1 1  **样例输出**  0  30  1 2  30  1 2  **样例说明**  按照样例输入从第 2 行开始逐行的解释如下：  1. 先手玩家在位置 1 召唤一个生命值为 6、攻击力为 3 的随从 A，是本方战场上唯一的随从。  2. 先手玩家在位置 2 召唤一个生命值为 2、攻击力为 4 的随从 B，出现在随从 A 的右边。  3. 先手玩家回合结束。  4. 后手玩家在位置 1 召唤一个生命值为 5、攻击力为 4 的随从 C，是本方战场上唯一的随从。  5. 后手玩家在位置 1 召唤一个生命值为 1、攻击力为 2 的随从 D，出现在随从 C 的左边。  6. 随从 D 攻击随从 B，双方均死亡。  7. 后手玩家回合结束。  8. 随从 A 攻击随从 C，双方的生命值都降低至 2。  **测试用例**  **CSP201609-3** |
| **实验过程：**   1. JUL或Log4j的使用及截图说明（说明程序开发调试过程中日志的作用，不再使用控制台调式输出，JUL或Log4j可自行搜索学习，也可查阅bb上的讲义）   思路：设计函数  1.判断是否胜利；  2.打印函数，  3.summon  4.attacker  5. end  题目就是按照题意一条指令一条指令的执行  通过测试的的记录    这道题的思路其实挺直的，按照题意一步一步来设计即可，我当时卡的是一下细节和数据的处理。  提交git的截图      第一次p3c扫描    改正后的p3c扫描    Log4j 介绍  Log4j 是 Apache 下的一款开源的日志框架，通过在项目中使用 log4j，我们可以控制日志信息输出的位置（控制台、文件、数据库）；也可以控制每一条日志的输出格式，通过定义日志的输出级别，可以更灵活的控制日志的输出过程，方便项目的调试。  Log4j 组件  Log4j 主要由 Loggers（日志记录器）、Appenders（输出端）、Layout（日志格式化器）组成。  Loggers：控制日志的输出级别与日志是否输出  Appenders：指定日志的输出方式（控制台、文件）  Layout：控制台日志信息的输出格式  1. Loggers  Loggers：日志记录器，负责处理日志记录。实例的命名就是类的全限定名。Logger 的名字大小写敏感。其命名有继承机制。如：org.apache.commons 的 logger 会继承 org.apache 的 logger。  Log4j 中，有一个特殊的 logger，叫 root，它是所有的 logger 的根。root logger 可以通过 Logger.getRootLogger() 方法获取。  2. Appenders  Appenders 用来指定日志输出到哪个地方，可以同时指定日志的输出目的地。常用的目的地有以下几种：  ConsoleAppender：将日志输出到控制台  FileAppender：将日志输出到文件中  DailyRollingFileAppender：将日志输出到一个日志文件，并且每天输出到一个新的文件  RollingFileAppender：将日志输出到一个日志文件，并且指定文件的尺寸，当文件大小达到指定尺寸时，会自动把文件改名，同时，产生一个新的文件  JDBCAppender：将日志保存到数据库  3. Layouts  Layouts 用于控制日志输出内容的格式，让我们可以使用各种需要的格式输出日志。常用的 Layouts：  HTML Layout：格式化日志输出为 HTML 表格形式  Simple Layout：简单的日志输出格式化。打印的日志格式为（info - message）  Pattern Layout：最强大的格式化器。可以根据自定义格式输出日志，如果没有指定转换格式，则使用默认的转换格式 |
| **实验结论：**  首先是题目一开始的理解问题，还有数据的理解不准确，一开始设计的时候我就设计了三个函数，1.判断是否胜利；2.打印函数 3.操作函数  后来在问题出现的时候只定位到3函数中，而3函数其实一大堆的代码，非常心塞，然后我又把3细致分为 (1)summon (2)attacker （3）end 函数 重新定位问题，搞完之后发现没有考虑随从的生命值和攻击值是多位数的情况（心态崩了，前面一直检查不出问题），最后发现没辙了，去看帖子，看一下别人的实现和经验帖子才发现这个问题，在改代码。 |
| **心得体会：**  可能是我自己的coding能力不过关，做完这题耗费了一天时间，其实细心一点的话其实还好。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：蔡树彬  2023年10月20日 |