

**面向对象程序设计上机实验报告**

实验题目6：动态游戏规则的存取实现

**学院名称 智能与计算学部**

**专 业 软件工程**

**学生姓名 王杨森**

**学 号 3020244116**

**年 级 2020级**

**班 级 软件工程3班**

**时 间 2022年 4月28日**

1. **实验目的**
2. 理解文件的读写过程；
3. 培养参数统一配置的思维；
4. 练习使用DEBUG工具调试程序；
5. **实验内容**
6. 修改前面实验的一款游戏，游戏中每个对象配置从一个配置文件中读取，配置文件格式类似如下：

#整个游戏的配置文件

#这个文件里面可能有注释，以#开头表示注释

#基地的属性

#每行形式为 [对象类型].[属性]=[整数] 其中对象类型和EnumObjectType 中的枚举类型字符完全相同。

#[属性]取值为 health range strength

#测试的时候，属性的整数值可能会改变

base.health=1000

base.range=0

base.strength=0

#重型坦克的属性

heavyTank.health=500

heavyTank.range=100

heavyTank.strength=100

mediumTank.health=300

mediumTank.range=50

mediumTank.strength=50

......

1. 程序必须从配置文件中动态的读取配置规则，完成每个对象的攻击情况，并且通过测试用例。
2. **程序实现**

**public** **class** Barrack **extends** Buildings{

**public** Barrack(Player player, **int** x, **int** y) {

**super**(player, x, y);

**this**.health = Params.*barrackHealth*;

**this**.range = Params.*barrackRange*;

**this**.strength = Params.*barrackStrength*;

}

**public** Roles traing(EnumObjectType type) {

Roles newRole = **null**;

**switch** (type) {

**case** ***dog*** :

newRole = **new** Dog(**this**.player, **this**.x, **this**.y);

**break**;

**case** ***RPGSoldier*** :

newRole = **new** RPGSoldier(**this**.player, **this**.x, **this**.y);

**break**;

**case** ***rifleSoldier*** :

newRole = **new** RifleSoldier(**this**.player, **this**.x, **this**.y);

**break**;

}

**if**(newRole != **null**) {

newRole.setType(type.toString());

BattleField.*gamesObjects*.add(newRole);

}

**return** newRole;

}

}

**public** **class** BattleField {

**private** **static** List<Player> *players* = **new** ArrayList<>();

**public** **static** List<GameObject> *gamesObjects* = **new** ArrayList<>();

**public** **static** List<String> *results* = **new** ArrayList<>();

**public** BattleField() {

*players* = **null**;

}

**public** **static** **void** init(String fileName) {

List<String[]> rules;

rules = *readIn*(fileName);

**for**(String[] rule: rules) {

**if**(rule[0].equals(EnumObjectType.***base***.toString())) {

**switch** (rule[1]) {

**case** "health" :

Params.*baseHealth* = Integer.*parseInt*(rule[2]);

**break**;

**case** "range" :

Params.*baseRange* = Integer.*parseInt*(rule[2]);

**break**;

**case** "strength" :

Params.*baseStrength* = Integer.*parseInt*(rule[2]);

**break**;

}

} **else** **if** (rule[0].equals(EnumObjectType.***heavyTank***.toString())) {

**switch** (rule[1]) {

**case** "health" :

Params.*heavyTankHealth* = Integer.*parseInt*(rule[2]);

**break**;

**case** "range" :

Params.*heavyTankRange* = Integer.*parseInt*(rule[2]);

**break**;

**case** "strength" :

Params.*heavyTankStrength* = Integer.*parseInt*(rule[2]);

**break**;

}

} **else** **if** (rule[0].equals(EnumObjectType.***mediumTank***.toString())) {

**switch** (rule[1]) {

**case** "health" :

Params.*mediumTankHealth* = Integer.*parseInt*(rule[2]);

**break**;

**case** "range" :

Params.*mediumTankRange* = Integer.*parseInt*(rule[2]);

**break**;

**case** "strength" :

Params.*mediumTankStrength* = Integer.*parseInt*(rule[2]);

**break**;

}

} **else** **if** (rule[0].equals(EnumObjectType.***rifleSoldier***.toString())) {

**switch** (rule[1]) {

**case** "health" :

Params.*rifleSoldierHealth* = Integer.*parseInt*(rule[2]);

**break**;

**case** "range" :

Params.*rifleSoldierRange* = Integer.*parseInt*(rule[2]);

**break**;

**case** "strength" :

Params.*rifleSoldierStrength* = Integer.*parseInt*(rule[2]);

**break**;

}

} **else** **if** (rule[0].equals(EnumObjectType.***RPGSoldier***.toString())) {

**switch** (rule[1]) {

**case** "health" :

Params.*RPGSoldierHealth* = Integer.*parseInt*(rule[2]);

**break**;

**case** "range" :

Params.*RPGSoldierRange* = Integer.*parseInt*(rule[2]);

**break**;

**case** "strength" :

Params.*RPGSoldierStrength* = Integer.*parseInt*(rule[2]);

**break**;

}

} **else** **if** (rule[0].equals(EnumObjectType.***dog***.toString())) {

**switch** (rule[1]) {

**case** "health" :

Params.*dogHealth* = Integer.*parseInt*(rule[2]);

**break**;

**case** "range" :

Params.*dogRange* = Integer.*parseInt*(rule[2]);

**break**;

**case** "strength" :

Params.*dogStrength* = Integer.*parseInt*(rule[2]);

**break**;

}

} **else** **if** (rule[0].equals(EnumObjectType.***barrack***.toString())) {

**switch** (rule[1]) {

**case** "health" :

Params.*barrackHealth* = Integer.*parseInt*(rule[2]);

**break**;

**case** "range" :

Params.*barrackRange* = Integer.*parseInt*(rule[2]);

**break**;

**case** "strength" :

Params.*barrackStrength* = Integer.*parseInt*(rule[2]);

**break**;

}

} **else** **if** (rule[0].equals(EnumObjectType.***warFactory***.toString())) {

**switch** (rule[1]) {

**case** "health" :

Params.*warFactoryHealth* = Integer.*parseInt*(rule[2]);

**break**;

**case** "range" :

Params.*warFactoryRange* = Integer.*parseInt*(rule[2]);

**break**;

**case** "strength" :

Params.*warFactoryStrength* = Integer.*parseInt*(rule[2]);

**break**;

}

}

}

}

**public** **static** **void** createPlayer(String name) {

**if**(name == **null**) **return**;

Player newPlayer = **new** Player(name);

*players*.add(newPlayer);

}

**public** **static** List<Player> getAllPlayer() {

**return** *players*;

}

**public** **static** GameBase createGameBase(Player player, **int** x, **int** y) {

GameBase newGameBase = **new** GameBase(player, x, y);

newGameBase.setType(EnumObjectType.***base***.toString());

*gamesObjects*.add(newGameBase);

**return** newGameBase;

}

**public** **static** List<String[]> readIn(String fileName) {

List<String[]> result = **new** ArrayList<>();

**try**(Reader reader = **new** FileReader(fileName);

LineNumberReader lineReader = **new** LineNumberReader(reader)) {

String line = lineReader.readLine();

**while**(line != **null**) {

**if**(line.trim().length() != 0 && line.trim().charAt(0) != '#') {

result.add(line.trim().split("[\\.=]"));

}

line = lineReader.readLine();

}

} **catch** (IOException e) {

e.printStackTrace();

}

**return** result;

}

}

**public** **abstract** **class** Buildings **extends** GameObject{

**public** Buildings(Player player, **int** x, **int** y) {

**super**(player, x, y);

}

}

**public** **class** Dog **extends** Roles{

**public** Dog(Player player, **int** x, **int** y) {

**super**(player, x, y);

**this**.health = Params.*dogHealth*;

**this**.range = Params.*dogRange*;

**this**.strength = Params.*dogStrength*;

}

}

**public** **enum** EnumObjectType {

***barrack***, ***warFactory***, ***rifleSoldier***, ***RPGSoldier***, ***mediumTank***, ***heavyTank***, ***dog***, ***base***,

}

**public** **class** GameBase **extends** Buildings{

**public** GameBase(Player player, **int** x, **int** y) {

**super**(player, x, y);

**this**.health = Params.*baseHealth*;

**this**.range = Params.*baseRange*;

**this**.strength = Params.*baseStrength*;

}

**public** Buildings building(EnumObjectType type, **int** x, **int** y) {

Buildings newBuilding = **null**;

**switch** (type) {

**case** ***barrack*** :

newBuilding = **new** Barrack(**this**.player, x, y);

**break**;

**case** ***warFactory*** :

newBuilding = **new** WarFactory(**this**.player, x, y);

**break**;

}

**if**(newBuilding != **null**) {

newBuilding.setType(type.toString());

BattleField.*gamesObjects*.add(newBuilding);

}

**return** newBuilding;

}

}

**public** **class** GameObject {

**protected** Player player;

**protected** **int** x, y;

**protected** **int** health;

**protected** **int** range;

**protected** **int** strength;

**protected** String type;

**public** GameObject(Player player, **int** x, **int** y) {

**this**.player = player;

**this**.x = x;

**this**.y = y;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** **int** getHealth() {

**return** **this**.health;

}

**public** **boolean** isDestroyed() {

**return** **this**.health <= 0;

}

**public** **void** getHurt(**int** attack) {

**if**(**this**.health - attack >= 0) **this**.health -= attack;

**else** **this**.health = 0;

}

**public** **void** move(**int** dx, **int** dy) {

**this**.x += dx;

**this**.y += dy;

}

**public** **void** attack(GameObject obj) {

**if**(obj.isDestroyed() || **this**.isDestroyed() || **this**.strength <= 0) **return**;

**if**(Utils.*getDistance*(**this**, obj) <= **this**.range) {

obj.getHurt(**this**.strength);

}

}

**public** **void** attack() {

**if**(**this**.strength <= 0 || **this**.isDestroyed()) **return**;

**double** min = **this**.range;

**int** idx = -1;

**for**(**int** i = 0; i < BattleField.*gamesObjects*.size(); i++) {

GameObject obj = BattleField.*gamesObjects*.get(i);

**if**(obj.isDestroyed() || obj.player == **this**.player) {

**continue**;

}

**if**(Utils.*getDistance*(obj, **this**) <= min) {

min = Utils.*getDistance*(obj, **this**);

idx = i;

}

}

**if**(idx >= 0) {

String line = **this**.debug() + " 攻击 " + BattleField.*gamesObjects*.get(idx).debug();

attack(BattleField.*gamesObjects*.get(idx));

line += " 攻击后 health=" + BattleField.*gamesObjects*.get(idx).getHealth();

BattleField.*results*.add(line);

}

}

**public** String debug() {

**return** "[" + **this**.player.getName() + "." + **this**.type + " live=" + !**this**.isDestroyed() + " x=" + **this**.x + " y=" + **this**.y + " health=" + **this**.health + "]";

}

}

**public** **class** HeavyTank **extends** Tank{

**public** HeavyTank(Player player, **int** x, **int** y) {

**super**(player, x, y);

**this**.health = Params.*heavyTankHealth*;

**this**.range = Params.*heavyTankRange*;

**this**.strength = Params.*heavyTankStrength*;

}

}

**public** **class** MediumTank **extends** Tank{

**public** MediumTank(Player player, **int** x, **int** y) {

**super**(player, x, y);

**this**.health = Params.*mediumTankHealth*;

**this**.range = Params.*mediumTankRange*;

**this**.strength = Params.*mediumTankStrength*;

}

}

**public** **class** Params {

**public** **static** **int** *baseHealth*;

**public** **static** **int** *baseRange*;

**public** **static** **int** *baseStrength*;

**public** **static** **int** *heavyTankHealth*;

**public** **static** **int** *heavyTankRange*;

**public** **static** **int** *heavyTankStrength*;

**public** **static** **int** *mediumTankHealth*;

**public** **static** **int** *mediumTankRange*;

**public** **static** **int** *mediumTankStrength*;

**public** **static** **int** *rifleSoldierHealth*;

**public** **static** **int** *rifleSoldierRange*;

**public** **static** **int** *rifleSoldierStrength*;

**public** **static** **int** *RPGSoldierHealth*;

**public** **static** **int** *RPGSoldierRange*;

**public** **static** **int** *RPGSoldierStrength*;

**public** **static** **int** *dogHealth*;

**public** **static** **int** *dogRange*;

**public** **static** **int** *dogStrength*;

**public** **static** **int** *barrackHealth*;

**public** **static** **int** *barrackRange*;

**public** **static** **int** *barrackStrength*;

**public** **static** **int** *warFactoryHealth*;

**public** **static** **int** *warFactoryRange*;

**public** **static** **int** *warFactoryStrength*;

}

**public** **class** Player {

**private** String name;

**public** Player() {

**this**.name = **null**;

}

**public** Player(String name) {

**this**.name = name;

}

**public** String getName() {

**return** **this**.name;

}

**public** **void** setName(String newName) {

**this**.name = newName;

}

}

**public** **class** RifleSoldier **extends** Soldiers{

**public** RifleSoldier(Player player, **int** x, **int** y) {

**super**(player, x, y);

**this**.health = Params.*rifleSoldierHealth*;

**this**.range = Params.*rifleSoldierRange*;

**this**.strength = Params.*rifleSoldierStrength*;

}

}

**public** **abstract** **class** Roles **extends** GameObject{

**public** Roles(Player player, **int** x, **int** y) {

**super**(player, x, y);

}

}

**public** **class** RPGSoldier **extends** Soldiers{

**public** RPGSoldier(Player player, **int** x, **int** y) {

**super**(player, x, y);

**this**.health = Params.*RPGSoldierHealth*;

**this**.range = Params.*RPGSoldierRange*;

**this**.strength = Params.*RPGSoldierStrength*;

}

}

**public** **abstract** **class** Soldiers **extends** Roles{

**public** Soldiers(Player player, **int** x, **int** y) {

**super**(player, x, y);

}

}

**public** **abstract** **class** Tank **extends** Roles{

**public** Tank(Player player, **int** x, **int** y) {

**super**(player, x, y);

}

}

**public** **class** Utils {

**public** **static** **double** getDistance(GameObject obj1, GameObject obj2) {

**return** Math.*sqrt*((obj1.x - obj2.x) \* (obj1.x - obj2.x) + (obj1.y - obj2.y) \* (obj1.y - obj2.y));

}

}

**public** **class** WarFactory **extends** Buildings{

**public** WarFactory(Player player, **int** x, **int** y) {

**super**(player, x, y);

**this**.health = Params.*warFactoryHealth*;

**this**.range = Params.*warFactoryHealth*;

**this**.strength = Params.*warFactoryStrength*;

}

**public** Tank building(EnumObjectType type) {

Tank newTank = **null**;

**switch** (type) {

**case** ***heavyTank*** :

newTank = **new** HeavyTank(**this**.player, **this**.x, **this**.y);

**break**;

**case** ***mediumTank*** :

newTank = **new** MediumTank(**this**.player, **this**.x, **this**.y);

**break**;

}

**if**(newTank != **null**) {

newTank.setType(type.toString());

BattleField.*gamesObjects*.add(newTank);

}

**return** newTank;

}

}

1. **实验结果**

测试代码：

**public** **class** Test {

**public** Test() {

// **TODO** Auto-generated constructor stub

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

// 某一款游戏，其主要角色如下:

// 游戏中每个对象有当前 x,y坐标，坐标值 取值范围为整数。

// \*\*\*\*每种对象的攻击力strength、攻击范围range、初始健康值health通过BattleField.init(String

// fileName)从从配置文件中初始化

// 非建筑物可以通过move(dx,dy)来移动坐标值,dx,dy表示x轴,y轴增量, 取值范围为整数。

// 对象A攻击B的时候，要满足两个对象之间直线距离小于等于A的攻击范围,否则攻击无效（被攻击方不减健康值）

// 任何对象有getHealth() 方法，返回当前生命值，如果已经死亡则返回 <=0的一个数字

// 任何对象有isDestroyed() 方法，如果生命值<=0则true,否则false

// GameBase 初始x,y值在创建时指定

// Barrack 兵营，可以训练出步枪兵、 RPG兵、军犬，初始x,y值在创建时指定

// RifleSoldier 步枪兵 ，初始x,y值就是兵营的x,y

// RPGSoldier 火箭兵 ，初始x,y值就是兵营的x,y

// Dog 军犬 ,初始x,y值就是兵营的x,y

// A.attack(B)，表示A攻击B对象，B.health=B.health-A.strength 注意这次没有一击毙命方法

// \*\*\*\* A.attack()，表示A寻找距离他最近、非己方、且活着的对象B 进行攻击，如果攻击范围内没有符合要求对象则什么也不做。

// 如果A.isDestroyed() 则A.attack() 没有任何效果

**try** {

// 测试点 初始化各种战斗单元的攻击力、健康值、攻击范围。具体格式见rules.ini

String filename = "D:\\北小洋\\21222\\面向对象程序设计\\OOP-Homework\\src\\com\\huawei\\classroom\\student\\h17\\rules.ini";

BattleField

.*init*("D:\\北小洋\\21222\\面向对象程序设计\\OOP-Homework\\src\\com\\huawei\\classroom\\student\\h17\\rules.ini");

// 总共有几个玩家，测试的时候最少2个玩家，最多10个玩家

String[] playerNames = { "A", "B", "C" };

**for** (**int** i = 0; i < playerNames.length; i++) {

// 测试点 传入玩家名字， 逐个创建玩家

BattleField.*createPlayer*(playerNames[i]);

}

// 测试点 返回先前创建的所有玩家

List<Player> players = BattleField.*getAllPlayer*();

// 基地、兵工厂、兵营、坦克、士兵、狗 这些可以创建的东西必须是 GameObject 子类

// List<GameObject> gameObjects 完全是我为了测试创建的对象，你们可以在程序里面不创建这种东西

List<GameObject> gameObjects = **new** ArrayList<GameObject>();

**for** (**int** i = 0; i < players.size(); i++) {

// 随机确定一个基地位置

**int** x = i \* 100;

**int** y = i \* 100;

// 测试点 传入玩家、坐标，创建基地

GameBase base = BattleField.*createGameBase*(players.get(i), x, y);

gameObjects.add(base);

// 测试点 传入玩家、坐标，创建兵营

Barrack barrack = (Barrack) base.building(EnumObjectType.***barrack***, x + 30, y + 30);

gameObjects.add(barrack);

// 测试点 传入玩家、坐标，创建兵工厂

WarFactory warFactory = (WarFactory) base.building(EnumObjectType.***warFactory***, x + 60, y + 60);

gameObjects.add(warFactory);

// 创建各种战斗对象 移动位置

gameObjects.add(barrack.traing(EnumObjectType.***rifleSoldier***));

gameObjects.get(gameObjects.size() - 1).move(3, 3);

gameObjects.add(barrack.traing(EnumObjectType.***RPGSoldier***));

gameObjects.get(gameObjects.size() - 1).move(2, 2);

gameObjects.add(barrack.traing(EnumObjectType.***dog***));

gameObjects.get(gameObjects.size() - 1).move(-2, -2);

gameObjects.add(warFactory.building(EnumObjectType.***heavyTank***));

gameObjects.get(gameObjects.size() - 1).move(30, 30);

gameObjects.add(warFactory.building(EnumObjectType.***mediumTank***));

gameObjects.get(gameObjects.size() - 1).move(60, 60);

}

**for** (**int** round = 0; round < 5; round++) {

**for** (**int** i = 0; i < gameObjects.size(); i++) {

// 测试点 攻击

gameObjects.get(i).attack();

//System.out.println(gameObjects.get(i));

}

String line = "第" + Integer.*toString*(round) + " 轮攻击";

BattleField.*results*.add(line);

//System.out.println("第" + round + " 轮攻击");

}

//测试结果见 reuslt.txt文件 供参考

**for** (**int** i = 0; i < gameObjects.size(); i++) {

String line = gameObjects.get(i).debug();

BattleField.*results*.add(line);

// System.out.println(gameObjects.get(i).debug());

}

// for(int i = 0; i < BattleField.results.size(); i++) {

// System.out.println(BattleField.results.get(i));

// }

} **catch** (Exception e) {

e.printStackTrace();

}

**try** (Reader reader = **new** FileReader("src/com/huawei/classroom/student/h17/result.txt");

LineNumberReader lineReader = **new** LineNumberReader(reader)){

List<String> correct = **new** ArrayList<>();

List<String> result = BattleField.*results*;

String line = lineReader.readLine();

**while**(line != **null**) {

correct.add(line);

line = lineReader.readLine();

}

// for(int i = 0; i < BattleField.results.size(); i++) {

// System.out.println(BattleField.results.get(i));

// }

**if**(correct.size() != result.size()) {

System.***out***.println("NO! 数量都不一样啦！" + correct.size() +" " + result.size());

} **else** {

**boolean** flag = **true**;

**for**(**int** i = 0; i < result.size(); i++) {

**if**(!result.get(i).equals(correct.get(i))) {

System.***out***.println("NO! 做错啦！idx = " + i);

System.***out***.println(result.get(i));

flag = **false**;

**break**;

}

}

**if**(flag) {

System.***out***.println("恭喜你做对啦！去做h18吧");

}

}

} **catch** (IOException e) {

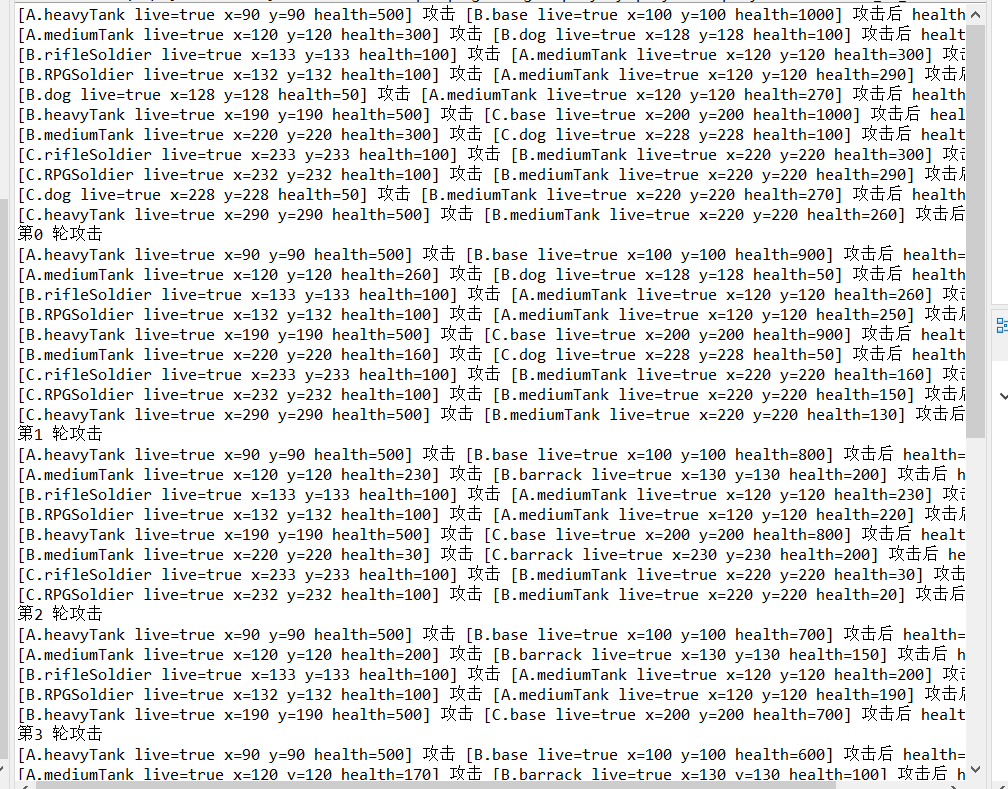
e.printStackTrace();

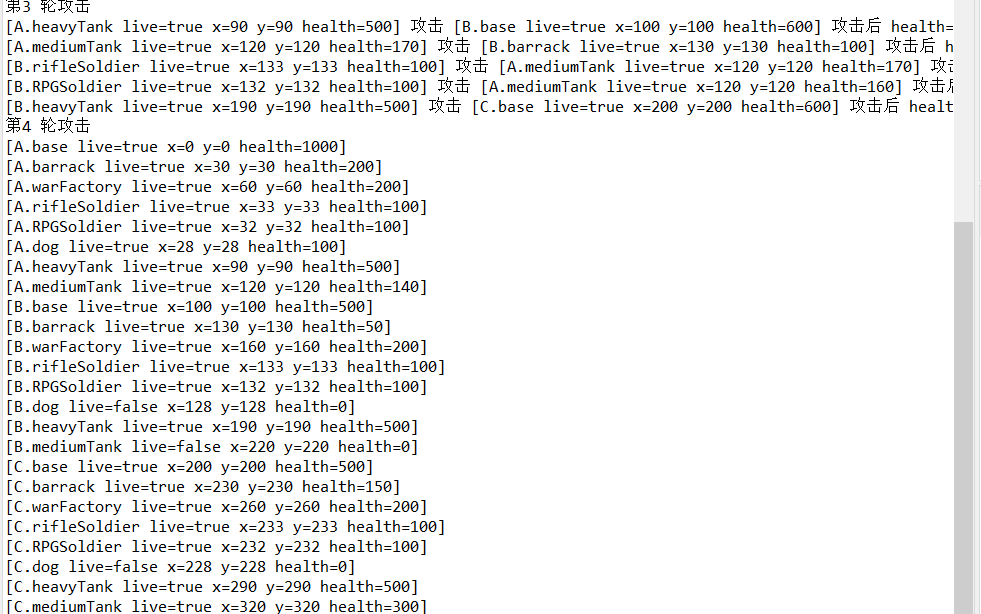
}

}

}

下面是自己输出的游戏运行过程：（我在程序中加了按照格式输出攻击过程的代码，存放在了一个List里）





肉眼比对其实是一样的，不过出于强迫症，我还写了从result.txt中读取数据的代码，可以直接比较运行结果和标准答案是否相同：

代码放出来了：

**try** (Reader reader = **new** FileReader("src/com/huawei/classroom/student/h17/result.txt");

LineNumberReader lineReader = **new** LineNumberReader(reader)){

List<String> correct = **new** ArrayList<>();

List<String> result = BattleField.*results*;

String line = lineReader.readLine();

**while**(line != **null**) {

correct.add(line);

line = lineReader.readLine();

}

**if**(correct.size() != result.size()) {

System.***out***.println("NO! 数量都不一样啦！" + correct.size() +" " + result.size());

} **else** {

**boolean** flag = **true**;

**for**(**int** i = 0; i < result.size(); i++) {

**if**(!result.get(i).equals(correct.get(i))) {

System.***out***.println("NO! 做错啦！idx = " + i);

System.***out***.println(result.get(i));

flag = **false**;

**break**;

}

}

**if**(flag) {

System.***out***.println("恭喜你做对啦！去做h18吧");

}

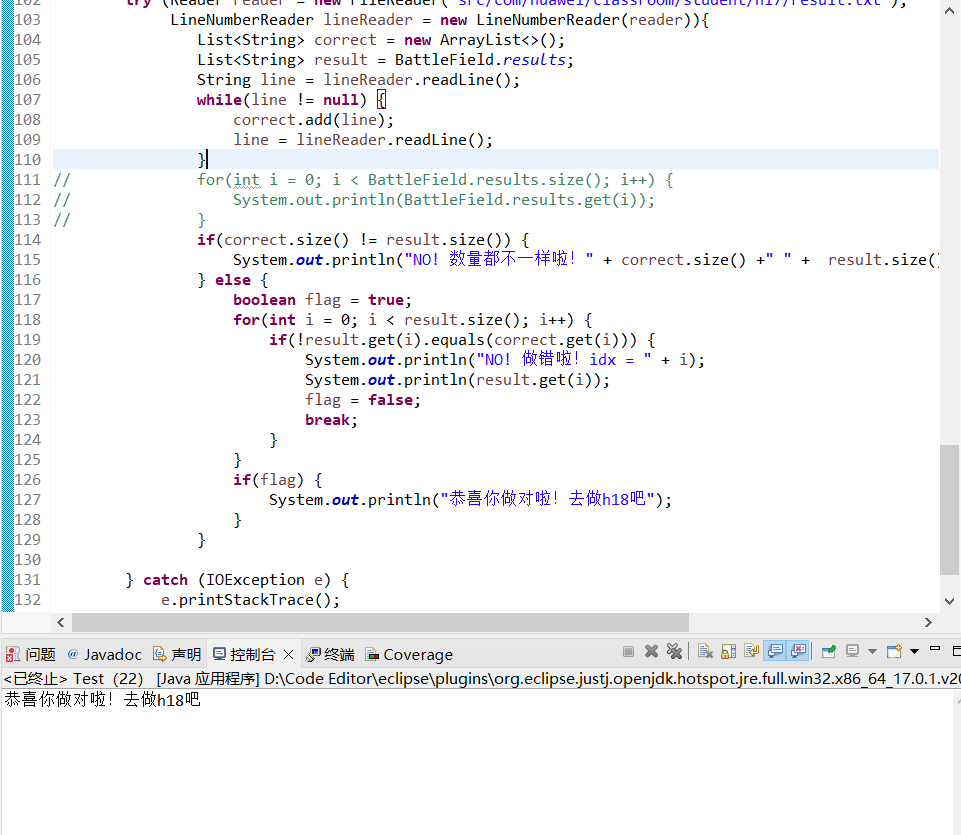
}

} **catch** (IOException e) {

e.printStackTrace();

}

上面是代码，下面是运行结果：



看上去我做对了，实际上也做对了。（因为输出的结果和result.txt是一样的）

1. **实验中遇到的问题及解决方法**

遇到的第一个问题是从配置文件中读取参数，这里我单独设置一个类Params，读取的参数都存入Params中，其他类在构造时直接访问Params类。

其余的便是对于之前已经实现的代码的更改，由于添加了玩家这一新属性，因此每个gameObject都需要设置一个新的属性player，在进行攻击时需要判断两个gameObject的player是否不同，此外由于添加了attack()方法，在这里我添加了一个List用于存放所有建立的gameObject，调用attack()时访问这个List找出距离最近的且存活的且不属于同一玩家的gameObject进行攻击。

通过这次实验，理解了参数统一配置的思维和文件读入的方法，同时复习了关于类的继承、封装知识，巧妙地设置父类继承可以极大优化之前的代码。