

**面向对象程序设计上机实验报告**

实验题目1：一个简单游戏算法的实现

**学院名称 智能与计算学部**

**专 业 软件工程**

**学生姓名 陈昊昆**

**学 号 3021001196**

**年 级 2021级**

**班 级 3班**

**时 间 2022年 3月22日**

1. **实验目的**

理解面向对象的概念；

理解继承、封装、多态的实现；

练习使用DEBUG工具调试程序；

1. **实验内容**
2. 某一款游戏，其主要角色如下:
3. HeavyTank 重型坦克 初始生命值200，攻击力 20
4. Medium Tank 轻型坦克 初始生命值100，攻击力 10
5. War Factroy 兵工厂 初始生命值100，无攻击力
6. Barrack 兵营，可以训练出步枪兵、 RPG兵、军犬,初始生命值100，无攻击力
7. Rifle soldier 步枪兵 初始生命值50(对战 军犬除外)，攻击力 5（对战军犬可以一次击毙军犬)
8. Rocket soldier 火箭兵 初始生命值50(对战 军犬除外)，攻击力 10
9. Dog 军犬 ,初始生命值50，攻击力5(对战人类时候一口毙命)
10. 此外还要能通过Solider.getLivingSoilderCount/getDeadedSoilderCount 统计现在有多少个活着的和死去的士兵数量
11. 攻击，两个对象之间相互攻击，削减每个对象的健康值。
12. 请遵循以上游戏规则，设计并实现游戏代码，使其能够通过测试用例
13. **程序实现**

游戏基本类

**public** **class** Fightob {

**public** **int** health;

**public** **int** x;

**public** **int** y;

**public** **int** attcnt; // 防止重复统计死亡的标记

**public** **int** range; // 攻击范围

**public** **int** attackhp; // 攻击力

**public** Fightob() {

attcnt = 0;

}

**public** **int** getHealth() { // 返回生命值

**return** health;

}

**public** **boolean** isDestroyed() { //判断是否死亡

**if** (health <= 0) {**return** **true**;}

**else** {**return** **false**;}

}

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

x += dx;

y += dy;

}

//攻击目标

**public** **void** attack(Fightob victim) {

// 判断是否在攻击范围内

**double** sqdistance = (**this**.x - victim.x)\*(**this**.x - victim.x)+(**this**.y - victim.y)\*(**this**.y - victim.y);

**if**(sqdistance <= **this**.range\***this**.range) {

victim.health -= **this**.attackhp;}

// 如果是士兵 统计其死亡与存活情况

**if** (victim **instanceof** Soldier ) {

**if** (victim.health <= 0 && victim.attcnt == 0) {

Soldier.*livingcnt*--;

Soldier.*deadcnt*++;

victim.attcnt++;}

}

}

}

士兵类

**public** **class** Soldier **extends** Fightob {

**public** **int** attcnt;

**public** **static** **int** *livingcnt* = 0;

**public** **static** **int** *deadcnt* = 0;

// 初始化士兵 未死亡标记和存活士兵+1

**public** Soldier() {

attcnt = 0;

Soldier.*livingcnt* ++;

}

**public** **static** **int** getLivingSoldierCount() {

**return** *livingcnt*;

}

**public** **static** **int** getDeadedSoldierCount() {

**return** *deadcnt*;

}

}

枪兵类

**public** **class** RifleSoldier **extends** Soldier {

**public** RifleSoldier() {

range = 5;

health = 50;

attackhp = 5;

}

@Override

**public** **void** attack(Fightob victim) {

**double** sqdistance = (**this**.x - victim.x)\*(**this**.x - victim.x)+(**this**.y - victim.y)\*(**this**.y - victim.y);

**if**(sqdistance <= range\*range) {

victim.health -= 5;}

// 枪兵能一击毙命狗

**if** (victim **instanceof** Dog) {

**if**(sqdistance <= range\*range) {

victim.health = 0;

}

}

**if** (victim **instanceof** Soldier ) {

**if** (victim.health <= 0 && victim.attcnt == 0) {

Soldier.*livingcnt*--;

Soldier.*deadcnt*++;

victim.attcnt++;}

}

}

火箭兵类

**public** **class** RPGSoldier **extends** Soldier {

**public** RPGSoldier() {

health = 50;

range = 10;

attackhp = 5;

}

}

狗类

**public** **class** Dog **extends** Fightob {

**public** Dog() {

health = 50;

range = 5;

attackhp = 5;

}

@Override

**public** **void** attack(Fightob victim) {

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

**double** sqdistance = (**this**.x - victim.x)\*(**this**.x - victim.x)+(**this**.y - victim.y)\*(**this**.y - victim.y);

**if**(sqdistance <= range\*range) {

victim.health -= 5;

}

// 狗能一口要死士兵

**if** (victim **instanceof** Soldier ) {

**if**(sqdistance <= range\*range) {

victim.health = 0;

}

}

**if** (victim **instanceof** Soldier) {

**if** (victim.health <= 0 && victim.attcnt == 0) {

Soldier.*livingcnt*--;

Soldier.*deadcnt*++;

victim.attcnt++;}

}

}

}

坦克类

**public** **class** Tank **extends** Fightob {

}

轻型坦克类

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

**public** MediumTank() {

health = 100;

attackhp = 10;

range = 10;

}

}

重型坦克类

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

**public** HeavyTank() {

health = 200;

attackhp = 20;

range = 10;

}

}

游戏基地类

**public** **class** GameBase **extends** Fightob{

**public** GameBase() {

health = 500;

}

// 初始化位置

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

GameBase b = **new** GameBase();

b.x = x;

b.y = y;

**return** b;

}

// 创建建筑物 兵营和兵工厂

**public** Fightob building(EnumObjectType out, **int** x, **int** y) {

**switch**(out) {

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

Barrack b = **new** Barrack();

b.x = x;

b.y = y;

**return** b;

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

WarFactory w = **new** WarFactory();

w.x = x;

w.y = y;

**return** w;

}

**return** **null**;

}

}

兵营类

**public** **class** Barrack **extends** Fightob{

**public** Barrack() {

health = 100;

}

**public** **int** bx;

**public** **int** by;

//创建兵的实例 并将兵营的位置赋给兵

**public** Fightob traing(EnumObjectType out) {

**switch**(out) {

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

bx = **this**.x;

by = **this**.y;

RifleSoldier a = **new** RifleSoldier();

a.x = bx;

a.y = by;

**return** a;

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

bx = **this**.x;

by = **this**.y;

RPGSoldier b = **new** RPGSoldier();

b.x = bx;

b.y = by;

**return** b;

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

bx = **this**.x;

by = **this**.y;

Dog c = **new** Dog();

c.x = bx;

c.y = by;

**return** c;

}

**return** **null**;

}

}

兵工厂类

**public** **class** WarFactory **extends** Fightob {

**public** **int** wx;

**public** **int** wy;

**public** WarFactory() {

health = 100;

}

//创建坦克的实例 并将兵工厂的位置赋给坦克

**public** Fightob building(EnumObjectType out) {

**switch**(out) {

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

wx = **this**.x;

wy = **this**.y;

MediumTank a = **new** MediumTank();

a.x = wx;

a.y = wy;

**return** a;

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

wx = **this**.x;

wy = **this**.y;

HeavyTank b = **new** HeavyTank();

b.x = wx;

b.y = wy;

**return** b;

}

**return** **null**;

}

}

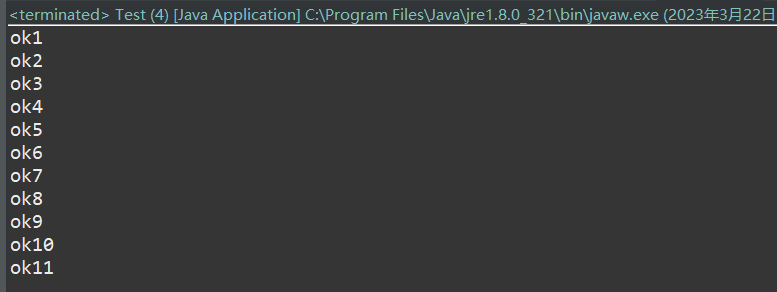
枚举类型类

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

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

}

1. **实验结果**



1. **实验中遇到的问题及解决方法**

我在这个实验中，学会了类的继承，如何实现接口和如何设置断点对代码实现调试。遇到的问题就是要重视对基本类的定义，将共有的属性和方法都写入基本类中，增加代码的复用性和可维护性。如果有特殊类的特殊功能，再在子类中重新写新的方法。