实验三报告

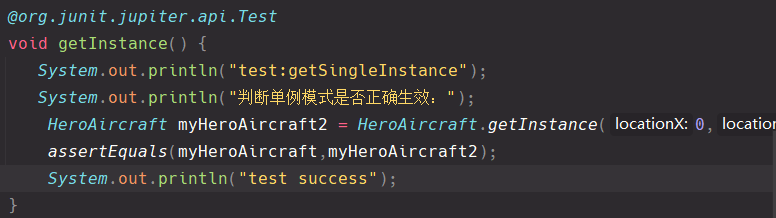
200111132-吴桐-11

# 单元测试

*结合飞机大战实例，在系统中选择英雄机、敌机、子弹和道具类的方法（包含其父类方法）作为单元测试的对象，为每个测试对象编写单元测试代码。要求至少选择3个类，每个类至少2个方法，并截图JUnit单元测试的结果。*

1. 测试用例*（一个方法一个测试用例）*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | H1 | | | |
| 待测试类及方法 | HeroAircraft.getInstance | | | |
| 测试类及方法 | HeroAircraftTest.getInstance | | | |
| 前提条件（如有） | 测试类运行开始有一个英雄机实例已经被创建 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 测试单例模式是否正常生效，即该方法只会产生唯一一个实例 | 再调用getInstance的方法用另一个英雄机引用接受，再使用assertEquals方法判断与一开始创建的对象二者是否相同  执行完毕打印test success信息 | 正常运行，无中断，打印test success信息 | 正常运行，无中断，打印test success信息 | 通过 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | H2 | | | |
| 待测试类及方法 | HeroAircraft.decreaseHp | | | |
| 测试类及方法 | HeroAircraftTest.decreaseHp | | | |
| 前提条件（如有） | 测试类运行开始有一个英雄机实例已经被创建 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 测试扣血函数是否会超过数据边界(hp<0) | 先将英雄机血量设置为max，然后调用decreaseHp函数,传参为两倍血量（如果没有健壮性处理则会使血量低于0），再用assertFalse函数判断英雄机的hp是否小于0，如果大于等于0，输出test success | 正常运行，无中断，打印test success信息 | 正常运行，无中断，打印test success信息 | 通过 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | H3 | | | |
| 待测试类及方法 | HeroAircraft.recoverHp | | | |
| 测试类及方法 | HeroAircraftTest.recoverHp | | | |
| 前提条件（如有） | 测试类运行开始有一个英雄机实例已经被创建 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 测试回血函数是否会超过边界（hp>maxHp） | 先将英雄机血量设置为1，然后调用recoverHp函数,传参为两倍血量（如果没有健壮性处理则会使血量大于回血最大值），再用assertFalse函数判断英雄机的hp是否大于maxHp，如果小于等于maxHp，输出test success | 正常运行，无中断，打印test success信息 | 正常运行，无中断，打印test success信息 | 通过 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | E1 | | | |
| 待测试类及方法 | EliteEnemy.vanish | | | |
| 测试类及方法 | EliteEnemyTest.vanish | | | |
| 前提条件（如有） | 测试该方法前生成一个精英敌机实例 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 测试vanish坠毁函数是否正常运行改变敌机存活状态 | 先通过notValid函数掉出刚刚生成敌机的状态，再调用vanish，后再次使用notValid函数查看敌机状态，都使用assertTrue函数判断返回的状态，如果无误，打印—Test Success-- | 正常运行，无中断，打印—Test Success--信息 | 正常运行，无中断，打印—Test Success--信息 | 通过 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | E2 | | | |
| 待测试类及方法 | EliteEnemy.forward | | | |
| 测试类及方法 | EliteEnemyTest.forward | | | |
| 前提条件（如有） | 测试该方法前生成一个精英敌机实例 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 测试forward函数在敌机飞跃超过屏幕下方后是否会自动判定失效 | 将敌机的y轴坐标先设置在超出屏幕设定值，即飞出屏幕下方。调用一次forward函数后，利用notValid函数返回飞机状态，利用assumeTrue判断，如果飞机失效符合预期，则打印—Test Success-- | 正常运行，无抛出异常，打印—Test Success--信息 | 正常运行，无抛出异常，打印—Test Success--信息 | 通过 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | E3 | | | |
| 待测试类及方法 | EliteEnemy.shoot | | | |
| 测试类及方法 | EliteEnemyTest.shoot | | | |
| 前提条件（如有） | 测试该方法前生成一个精英敌机实例 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 测试精英敌机的shoot射击函数是否正常返回子弹。 | 调用敌机的shoot函数，assumeFalse看返回的列表是否为空，再依次使用assumeFalse检查列表中的子弹设计方向是否为朝下（1）,如果符合预期正常执行，并且打印—Test Success-- | 正常运行，无抛出异常，打印—Test Success--信息 | 正常运行，无抛出异常，打印—Test Success--信息 | 通过 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | B1 | | | |
| 待测试类及方法 | BloodProps.crash | | | |
| 测试类及方法 | BloodPropsTest.crash | | | |
| 前提条件（如有） | 在指定位置生成一个血包物品 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 测试血包与英雄机的碰撞反馈是否正常 | 先生成一个英雄机在血包的位置，然后调用血包的crash函数，参数为英雄机，得到是否碰撞的boolean，用assumeTrue判断结果，再使得英雄机坐标在距离血包足够远的地方，重复判断assumeFalse,成功运行后打印—Test Crash Finished-- | 正常运行，无抛出异常，打印—Test Crash Finished--信息 | 正常运行，无抛出异常，打印—Test Crash Finished--信息 | 通过 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | B2 | | | |
| 待测试类及方法 | BloodProps.effectCrash | | | |
| 测试类及方法 | BloodPropsTest.effectCrash | | | |
| 前提条件（如有） | 在指定位置生成一个血包物品 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 检验血包的被碰撞后的生效函数 | 直接调用该血包的effectCrash函数，目前函数内容只有打印回血量 | 正常运行，成功打印回血量，后输出—Test effect Crash Finished--信息 | 正常运行，成功打印回血量，后输出—Test effect Crash Finished--信息 | 通过 |



*用例编号：唯一标识测试用例的序号，一般是数字或模块名首字母大写+数字序号。*

*待测试类及方法：该用例所测试的类名和方法名*

*测试类及方法：相应的测试代码的类名和方法名*

*前提条件（如有）：执行该测试用例的前提条件，比如碰撞检测，需已创建英雄机和敌机（或道具）。*

*用例描述：用一句话简单总结该测试用例的用意和目的。*

*测试步骤：详细完整的操作过程描述。*

*期望结果：正常情况下的响应结果。*

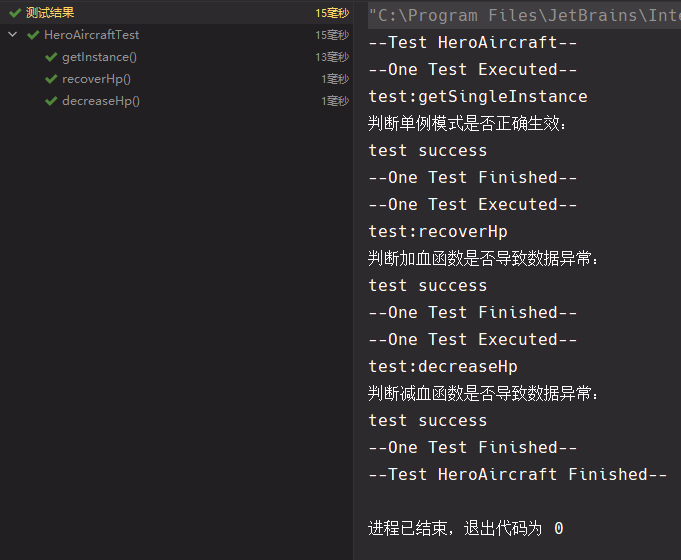
*实际结果：程序通过测试步骤后实际的响应结果。*

*测试结果：通过或失败*

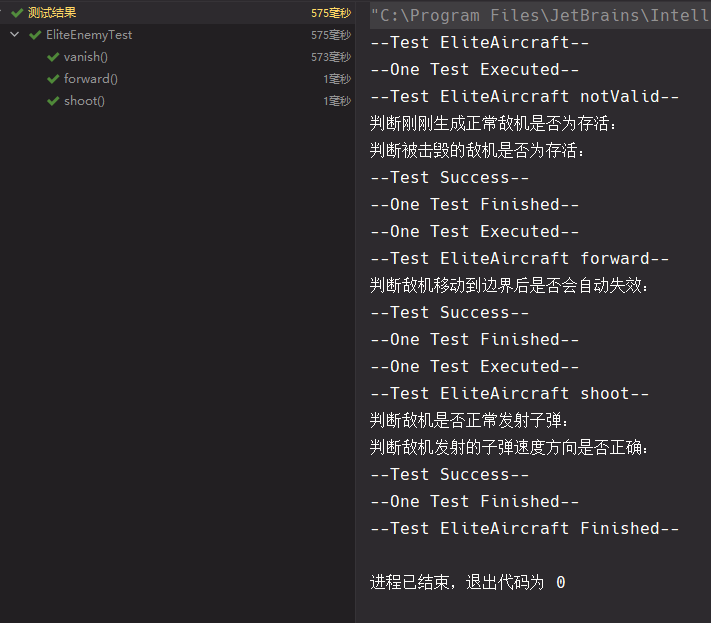
1. JUnit单元测试结果

*请截图JUnit每个测试类（包含多个方法）的运行结果。*

*1.* HeroAircraftTest



*2.* EliteEnemyTest



*3.* BloodPropsTest



# 编码规约

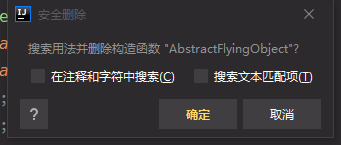
*列举2个实验中使用到的阿里编码规约插件的例子，截图或文字描述插件扫描出的代码问题以及你是如何解决的。*

1.修改AbstractFlyingObject类

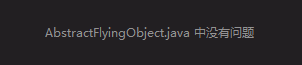
只有一个问题：类中有一个从未被使用的空构造函数



这里直接右键问题中的错误跳出快速修复选项：安全删除

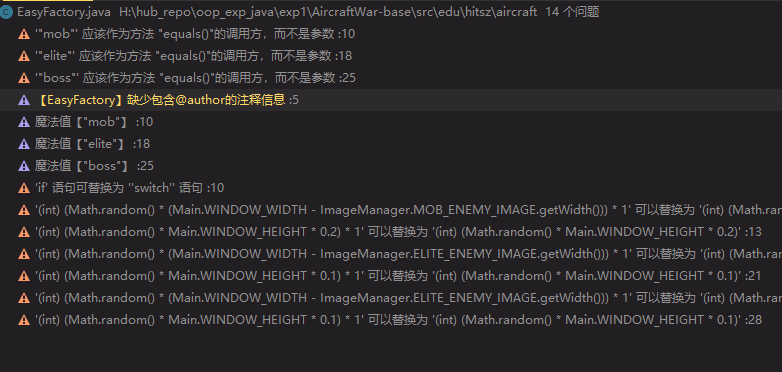


利用idea的功能安全删除



2.修改EasyFactory

该类为我自己构造的简单难度下创造敌机的工厂实例



右键错误使用快速修复，idea自动操作

把if结构改为switch结构，添加default

并且将构造函数参数简化



修复完成

