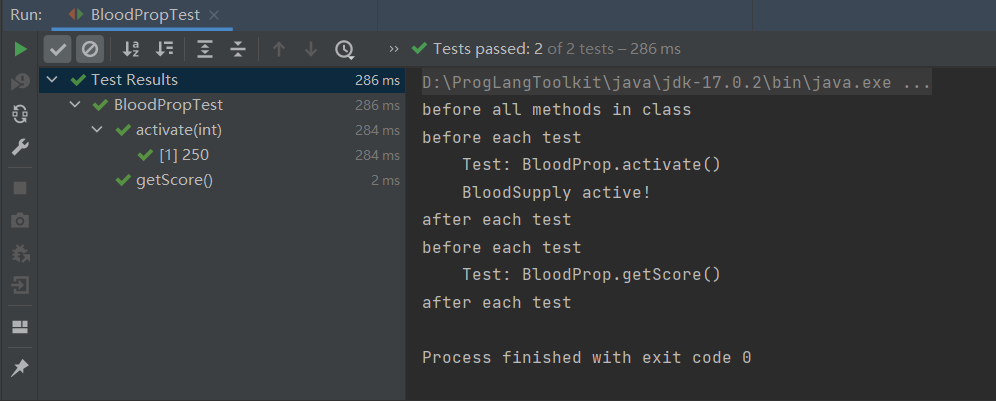
实验三报告

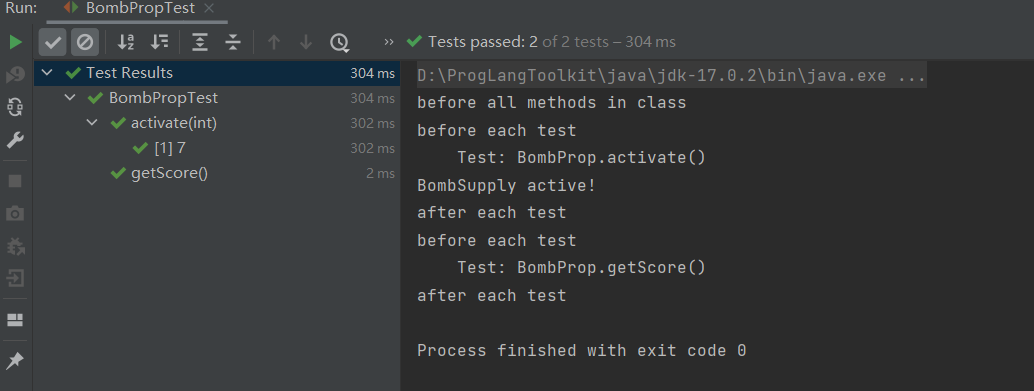
# 单元测试

1. 测试用例

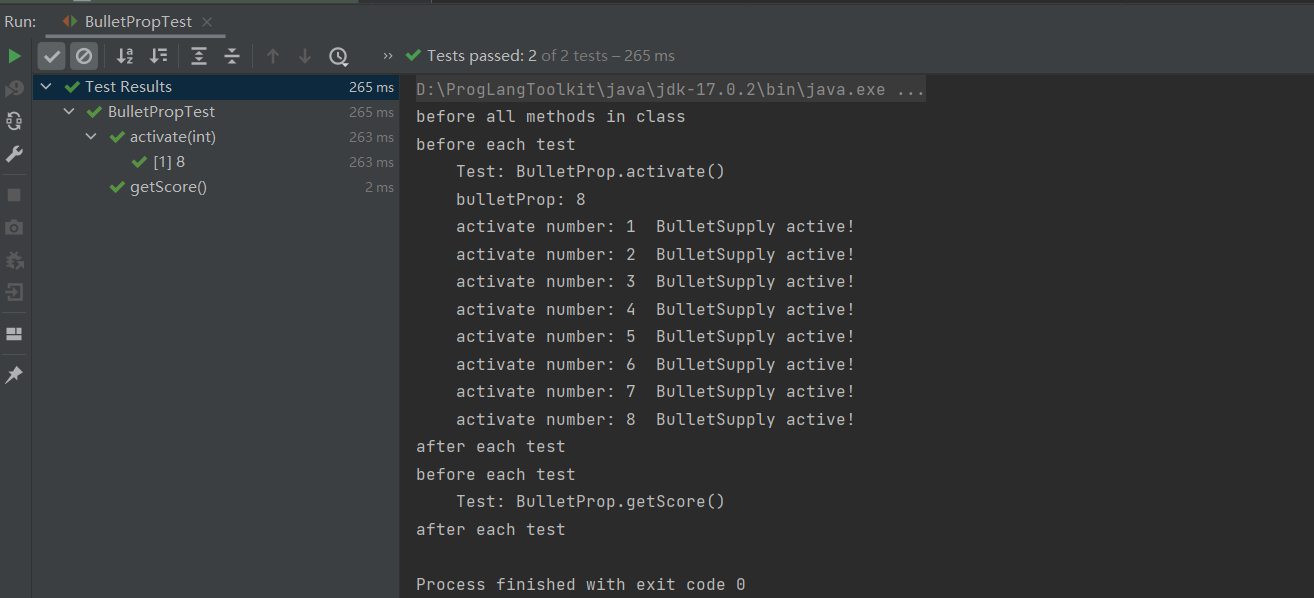
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | BloodPropTest1 | | | |
| 待测试类及方法 | BloodProp.activate() | | | |
| 测试类及方法 | BloodPropTest.activate() | | | |
| 前提条件（如有） | heroAircraft等初始化，生成bloodProp道具 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试  测试加血道具功能  参数为英雄机血量 | 1.初始化各对象  2.生成加血道具  3.根据参数设置英雄机血量  4.激活道具并检查结果 | 300 | 300 | 通过 |
| 用例编号 | BloodPropTest2 | | | |
| 待测试类及方法 | BloodProp.getScore() | | | |
| 测试类及方法 | BloodPropTest.getScore() | | | |
| 前提条件（如有） | 生成bloodProp道具 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 测试  测试加血道具分数 | 1.初始化道具工厂  2.生成加血道具  3.获取加血道具分数并核对正确数值 | 30 | 30 | 通过 |



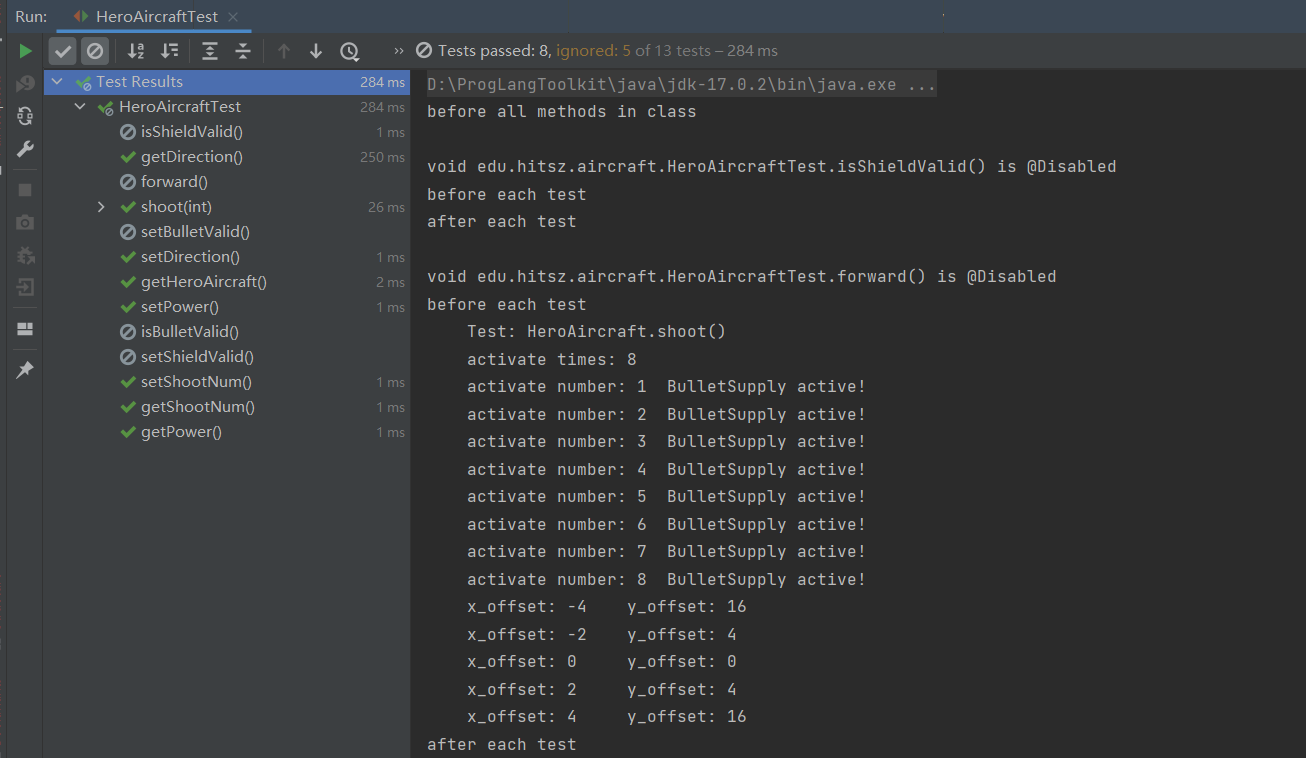
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | BombPropTest1 | | | |
| 待测试类及方法 | BombProp.activate() | | | |
| 测试类及方法 | BombPropTest.activate() | | | |
| 前提条件（如有） | enemyAircrafts等初始化，生成bombProp道具 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试  测试炸弹道具功能  参数为每种敌机的数量 | 1.初始化各对象  2.生成炸弹道具  3.由工厂制造各敌机  4.激活炸弹道具并检查结果（只能炸毁非boss机） | 7 | 7 | 通过 |



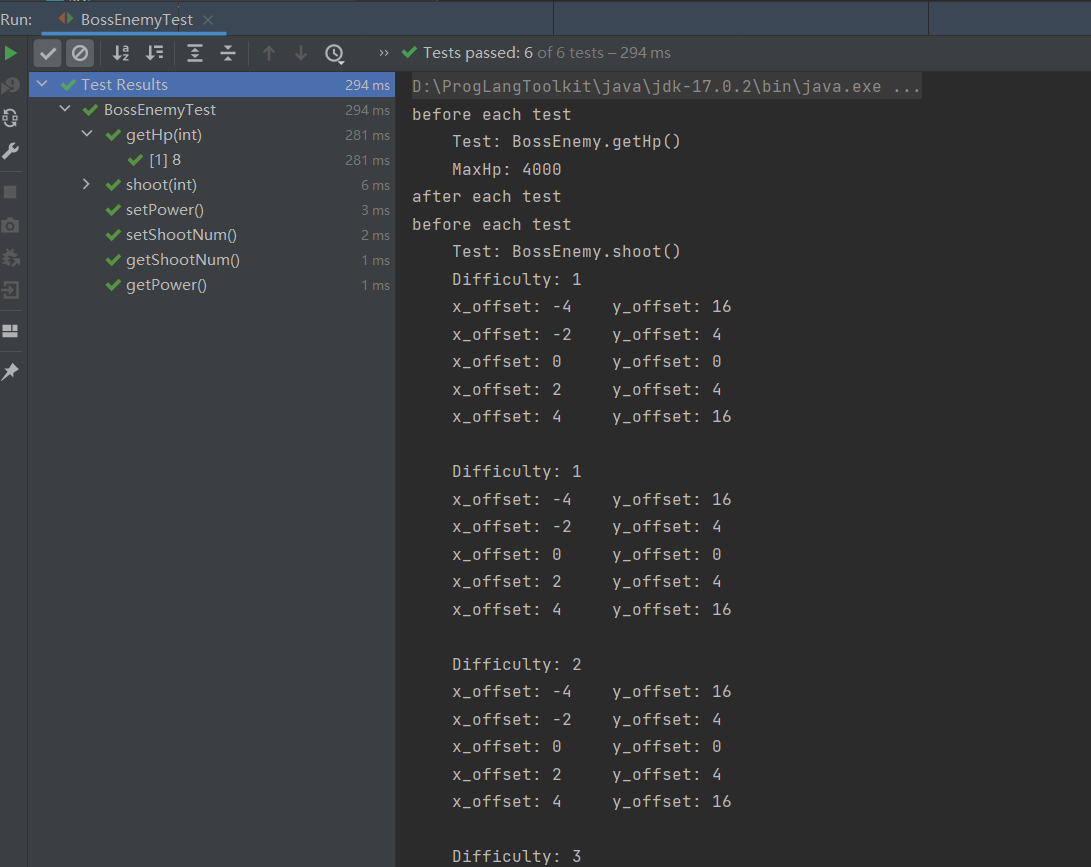
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | BulletPropTest1 | | | |
| 待测试类及方法 | BulletProp.activate() | | | |
| 测试类及方法 | BulletPropTest.activate() | | | |
| 前提条件（如有） | heroAircraft初始化，生成bulletProp道具 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试  测试火力道具功能  参数为激活火力道具的次数，改变子弹个数，该测试只测试子弹数 | 1.初始化各对象  2.生成火力道具  3.根据参数激活特定次数的火力道具  4.根据可装配子弹数目上限检查激活后的结果 | 5 | 5 | 通过 |



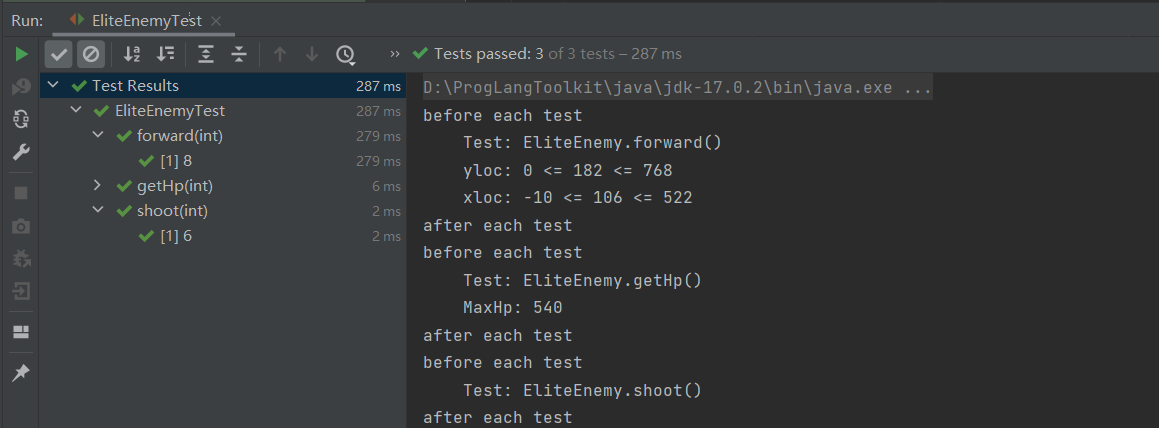
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | HeroAircraftTest1 | | | |
| 待测试类及方法 | HeroAircraft.shoot() | | | |
| 测试类及方法 | HeroAircraftTest.shoot() | | | |
| 前提条件（如有） | heroAircraft,bulletProp等对象初始化 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试  参数为火力道具激活次数  测试火力道具对shoot的加成  测试shoot弹道 | 1.初始化各对象  2.使用火力道具  3.分别测试shoot方法发射的子弹xy绝对坐标或均值（打印出的为便于观察的相对坐标）  注：因本实验要求只能测试一个测试数据，原参数化测试数据被注释 | 699  256 | 699  256 | 通过 |



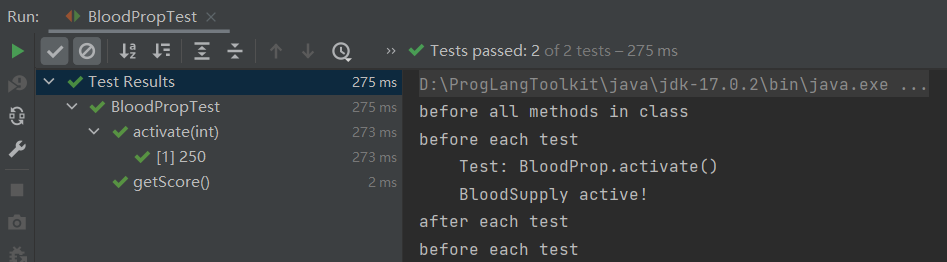
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | BossEnemyTest1 | | | |
| 待测试类及方法 | BossEnemy.getHp() | | | |
| 测试类及方法 | BossEnemyTest.getHp() | | | |
| 前提条件（如有） | bossFactory初始化，生成BossEnemy类的实例 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试  参数为难度  根据设定的难度生成boss敌机，获取boss敌机血量值 | 1.bossFactory初始化  2.根据参数构造对应难度的boss敌机  3.获取boss敌机血量  注：当前代码中，难度只调控了boss机血量 | 4000 | 4000 | 通过 |
| 用例编号 | BossEnemyTest2 | | | |
| 待测试类及方法 | BossEnemy.shoot() | | | |
| 测试类及方法 | BossEnemyTest.shoot() | | | |
| 前提条件（如有） | bossFactory初始化，生成BossEnemy类的实例 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试  参数为从最低难度变化到的最高难度  根据设定的难度生成boss敌机，检查boss在不同难度下的弹道 | 1.bossFactory初始化  2.根据参数构造对应难度的boss敌机  3.遍历难度最小从1开始，检查boss敌机子弹xy坐标绝对值或均值（只检查了抛物线，散射未检查）  注：当前代码中，难度只调控了boss机血量 | 51  63  67  63  51  213  43  55  59  55  43  206  … | 51  63  67  63  51  213  43  55  59  55  43  206  … | 通过  注：由于boss敌机生成坐标有随机性，以及表格空间有限，此处仅列举示例的部分 |

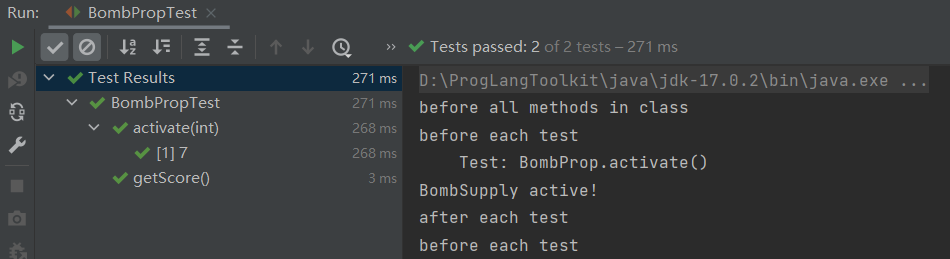


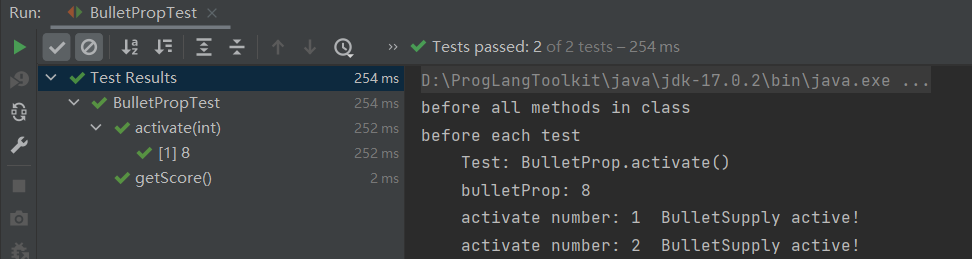
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | EliteEnemyTest1 | | | |
| 待测试类及方法 | EliteEnemy.forward() | | | |
| 测试类及方法 | EliteEnemyTest.forward() | | | |
| 前提条件（如有） | eliteFactory初始化，生成EliteEnemy类的实例 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试  检测forward方法的范围限制  精英机的forward方法同时需要检测左右反弹和上下出界，故仅对其进行测试即可 | 1.初始化eliteFactory等对象  2.工厂模式生成elite实例  3.根据elite是否超出最下方边界对其状态进行检测 | true  true  true  true | true  true  true  true | 通过 |
| 用例编号 | EliteEnemyTest2 | | | |
| 待测试类及方法 | EliteEnemy.getHp() | | | |
| 测试类及方法 | EliteEnemyTest.getHp() | | | |
| 前提条件（如有） | eliteFactory初始化，生成EliteEnemy类的实例 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试，参数为难度  生成elite机，检查getHp()方法 | 1.初始化eliteFactory等对象  2.工厂模式生成elite实例  3.测试elite血量值 | 540 | 540 | 通过 |
| 用例编号 | EliteEnemyTest3 | | | |
| 待测试类及方法 | EliteEnemy.shoot() | | | |
| 测试类及方法 | EliteEnemyTest.shoot() | | | |
| 前提条件（如有） | eliteFactory初始化，生成EliteEnemy类的实例 | | | |
| 用例描述 | 测试步骤 | 期望结果 | 实际输出 | 测试结果 |
| 参数化测试，参数为难度  检查elite机子弹绝对位置 | 1.初始化eliteFactory等对象  2.工厂模式生成elite实例  3.测试elite子弹绝对坐标 | 121 10  407 42  57 60  227 145  377 106  141 124 | 121 10  407 42  57 60  227 145  377 106  141 124 | 通过  注：生成有随机性，仅举作示例 |

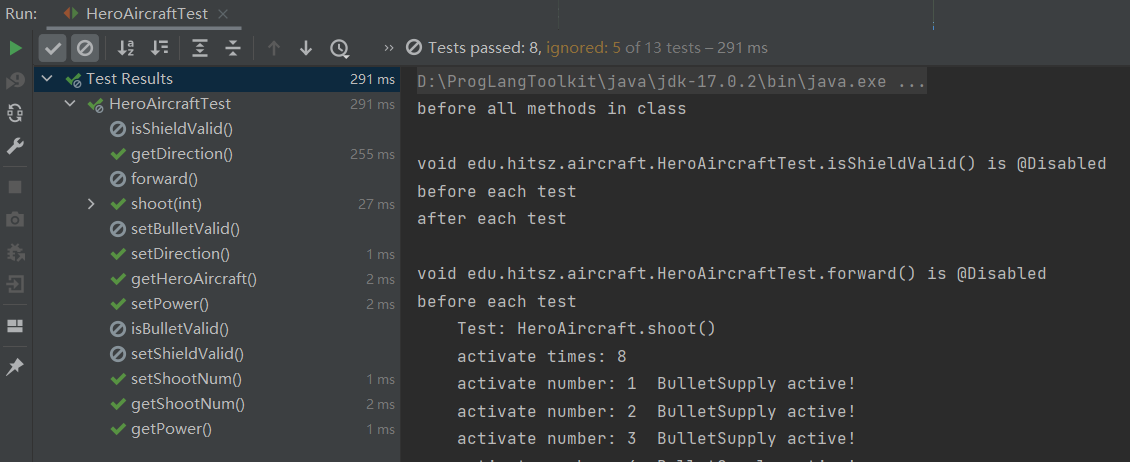


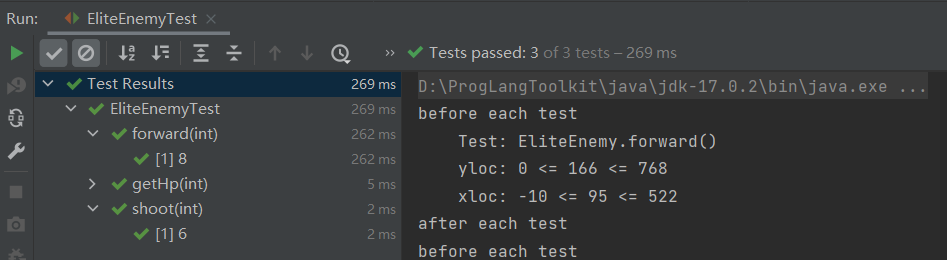
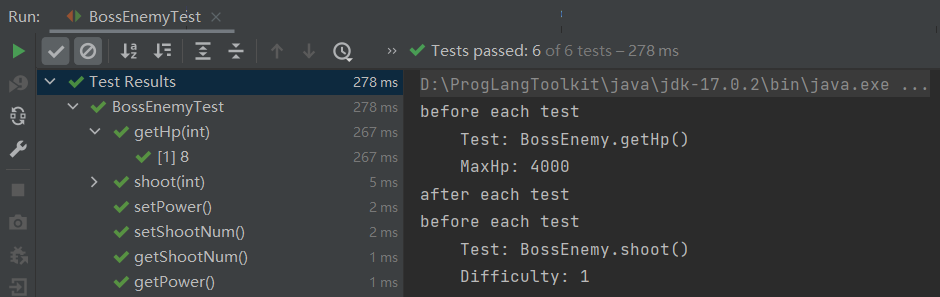
1. JUnit单元测试结果





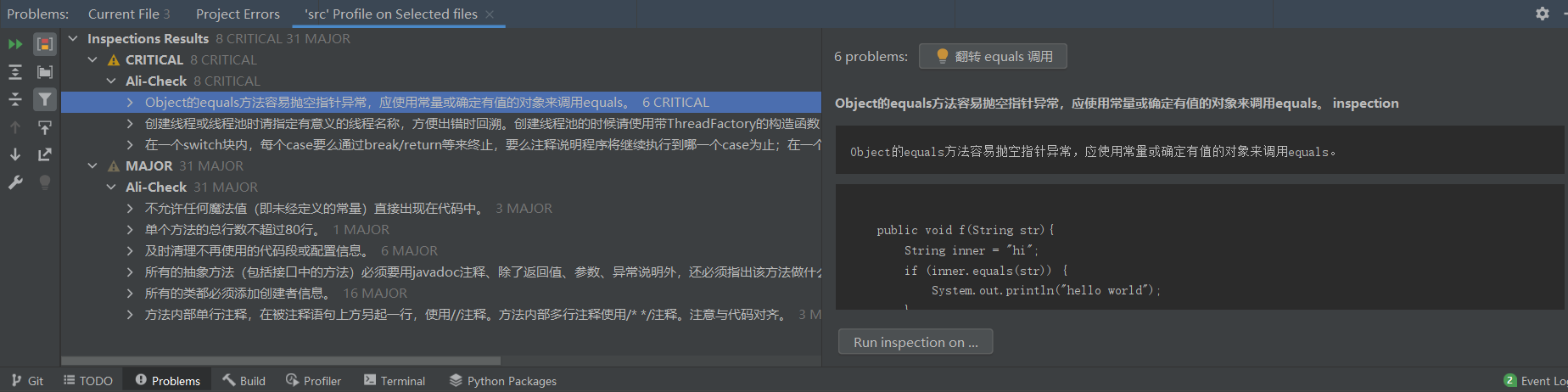






# 编码规约

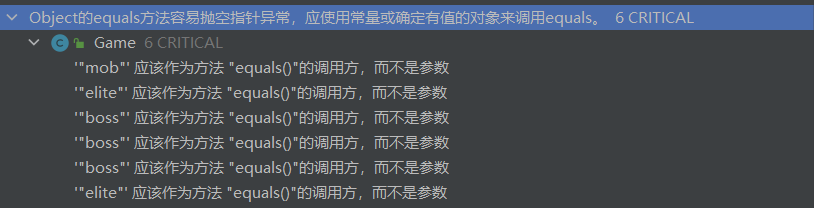
使用阿里编码规约插件能方便的找出代码中存在的问题，如下图所示：



对于我写下的未特别加以注意的代码，插件扫描出了8 CRITICAL, 31 MAJOR的问题。

下面对其中2类比较典型的问题进行修改：

1. **Object的quals方法调用的问题，应当使用确定有值的对象来调用**



很容易能通过插件将原来代码中调用方进行调换，修复了这6个CRITICAL，更正后的示例代码如下：



1. **使用switch时应当有default语句，即使什么也不做**

****

最后，处理了剩余的绝大多数MAJOR问题，在代码中添加了完整的javadoc注释，最终的插件扫描结果如下：

