|  |  |
| --- | --- |
| **学 号：** | 0122004950122 |

****

**软件工程实践（一）**

|  |  |
| --- | --- |
| **学 院** | **计算机与人工智能学院** |
| **专 业** | **软件工程** |
| **班 级** | **软件2001** |
| **姓 名** | **闫瀚** |
| **编 号** | **0122004950122** |
| **指导教师** | **唐祖锴** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2023 | 年 | 1 | 月 | 3 | 日 |

目录

[1 任务概述 1](#_Toc8150)

[1.1任务目的 1](#_Toc14633)

[1.2任务内容 1](#_Toc14765)

[1.3任务要求 1](#_Toc31831)

[1.3.1阅读和描述样例工程 1](#_Toc10924)

[1.3.2标注样例工程中的代码 1](#_Toc13282)

[1.3.3扩充和维护样例工程 1](#_Toc13843)

[1.3.4功能扩充点 1](#_Toc1716)

[1.3.5编写测试用例 2](#_Toc15655)

[2 任务分析 2](#_Toc22264)

[3 开发计划 2](#_Toc28539)

[4 软件配置计划 3](#_Toc23486)

[5 测试计划 3](#_Toc32382)

[6 实施情况 3](#_Toc1238)

[6.1阅读和描述样例工程 3](#_Toc30324)

[6.2标注样例工程中的代码 5](#_Toc25181)

[6.3维护样例工程 5](#_Toc12358)

[6.4扩充样例工程 6](#_Toc21968)

[6.4.1房间物品和look命令 6](#_Toc10961)

[6.4.2高级的back命令 8](#_Toc16549)

[6.4.3传送房间 9](#_Toc20371)

[6.4.4独立的Player类 9](#_Toc17591)

[6.5 程序运行结果 15](#_Toc1521)

[6.6 测试结果分析 18](#_Toc3336)

[6.6.1测试方法 18](#_Toc20833)

[6.6.2测试Command类 21](#_Toc2474)

[6.6.3测试CommandWords类 21](#_Toc23107)

[6.6.4测试Player类 21](#_Toc8138)

[6.7 代码提交状况 22](#_Toc32306)

[7 实施过程问题记录与分析 22](#_Toc5936)

[8 任务总结 22](#_Toc11272)

[9 参考文献 22](#_Toc6326)

[《软件工程实践（一）》成绩评定表 23](#_Toc12536)

# 1 任务概述

1.1任务目的

1.理解软件代码规范的重要性

2.理解代码变化对软件质量带来的影响

3.掌握基于Git的个人代码版本维护方法

4.掌握MarkDown文件编写方法

1.2任务内容

1.阅读、理解和标注样例代码

2.分析和学习代码质量特征、设计方法和编程风格

3.运用所学方法，对开源代码进行标注

4.对样例工程进行简单功能扩充和维护

1.3任务要求

1.3.1阅读和描述样例工程

1.fork样例工程，并clone到本地仓库；

2.在本地开发环境上运行样例工程，理解样例工程的代码逻辑；

3.精读样例工程软件代码，描述代码结构及部件组成；

4.以UML图描述样例工程的组成及结构图（类及类之间的关系）

5.可结合markdown语法和mermaid插件绘制所需图形

1.3.2标注样例工程中的代码

1.基于javadoc规范标注代码，对包、类、方法、代码片段、参数和语句等代码层次进行注释（可参考Game类的标注样例）；

2.注释后的代码提交到本地代码库后，同步推送到远程代码仓库；

3.可参考ESLint、github/super-linter等开发插件了解关于代码规范的相关知识；

1.3.3扩充和维护样例工程

1.对样例代码中的功能设计进行分析，找出若干设计缺陷和改进点，并进行修正或扩充，并集成到工程代码中；

2.可借助代码质量分析工具或代码规范检查工具对代码质量进行分析，发现潜在问题。

提示：样例工程的代码结构存在一些可以改进的功能点，可参考下列说明进行改进：

在Game类的processCommand()方法中，当用户输入的命令被辨认出来以后，有一系列的if语句用来分派程序到不同的地方去执行。从面向对象的设计原则来看，这种解决方案不太好，因为每当要加入一个新的命令时，就得在这一堆if语句中再加入一个if分支，最终会导致这个方法的代码膨胀得极其臃肿。如何改进程序中的这个设计，使得命令的处理更模块化，且新命令的加入能更轻松？请描述你的解决思路，并对你的解决方案进行实现和测试。

1.3.4功能扩充点

1.样例工程“world-of-zuul”具备最基本的程序功能，该项目具有极大的扩展空间，各位同学可选择或自行设计系统结构优化或功能扩充需求，完成3项左右的功能扩充实现；

可供参考的结构优化或功能扩充项包括但不限于以下内容：

(1)扩展游戏，使得一个房间里可以存放任意数量的物件，每个物件可以有一个描述和一个重量值，玩家进入一个房间后，可以通过“look”命令查看当前房间的信息以及房间内的所有物品信息；

(2)在游戏中实现一个“back”命令，玩家输入该命令后会把玩家带回上一个房间；

(3)在游戏中实现一个更高级的“back”命令，重复使用它就可以逐层回退几个房间，直到把玩家带回到游戏的起点；

(4)在游戏中增加具有传输功能的房间，每当玩家进入这个房间，就会被随机地传输到另一个房间；

(5)在游戏中新建一个独立的Player类用来表示玩家，并实现下列功能需求：

\* 一个玩家对象应该保存玩家的姓名等基本信息，也应该保存玩家当前所在的房间；

\* 玩家可以随身携带任意数量的物件，但随身物品的总重量不能操过某个上限值；

\* 在游戏中增加两个新的命令“take”和“drop”，使得玩家可以拾取房间内的指定物品或丢弃身上携带的某件或全部物品，当拾取新的物件时超过了玩家可携带的重量上限，系统应给出提示；

\* 在游戏中增加一个新的命令“items”, 可以打印出当前房间内所有的物件及总重量，以及玩家随身携带的所有物件及总重量；

\* 在某个或某些房间中随机增加一个magic cookie（魔法饼干）物件，并增加一个“eat cookie”命令，如果玩家找到并吃掉魔法饼干，就可以增长玩家的负重能力；

(6)扩充游戏基本架构，使其支持网络多人游戏模式，具备玩家登陆等功能；

(7)为单机或网络版游戏增加图形化用户界面，用过可以通过图形化界面执行游戏功能；

(8)可以为游戏增加数据库功能，用于保存游戏状态和用户设置；

(9) ......

1.3.5编写测试用例

针对功能改进和扩充，在项目结构中编写单元测试用例，对代码执行单元测试。

# 2 任务分析

本次实践任务主要是基于样例工程进一步开发和完善“巨洞冒险”这一游戏，其工作重点应该在于对样例工程进行功能扩充。主要采用JAVA面向对象的方法，使用IDEA进行开发。在开发时要注意代码规范，例如命名规范和注释规范等，基于javadoc规范标注代码，使用驼峰命名法及markdown语法格式。在开发过程中使用Github进行代码管理。本次任务的重点在于开发者对程序的详细理解，以及立足于自身理解对程序进行改进，开发出更多的功能并避免产生bug，随着开发的进行解决出现的问题。

# 3 开发计划

1.阅读代码，通过Javaboc对代码中实现的方法添加注释，并用EA画出类图建模。

2.创建markdown文件REPORT，利用markdown语法记录软件开发过程 。

3.使用IntelliJ IDEA Community Edition 2022.1.2来进行软件开发。

4.在对代码更改、添加之后使用Github进行版本管理。

5.开发完成后利用Junit进行代码测试。

# 4 软件配置计划

版本的编码规范**：**使用Ver.\*.\*进行版本编码，比如Ver.1.2意为第一个版本的第二次更新。

命名规范：采用驼峰命名法，功能的实现采用上下文模式。

分支管理规范：开发始终在master分支下进行合并。

提交规范：通过git提交到代码库。

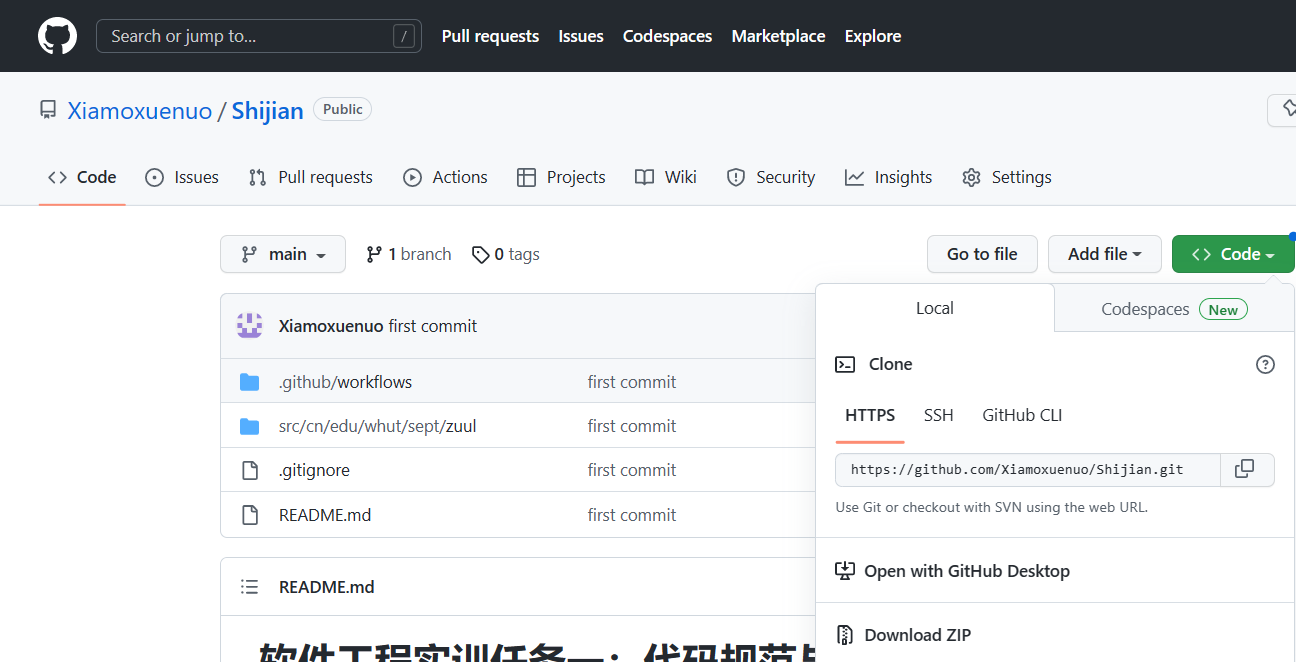
# 5 测试计划

对Command类、CommandWords类和Player类用Junit进行测试，主要测试除了get和show以外的方法。

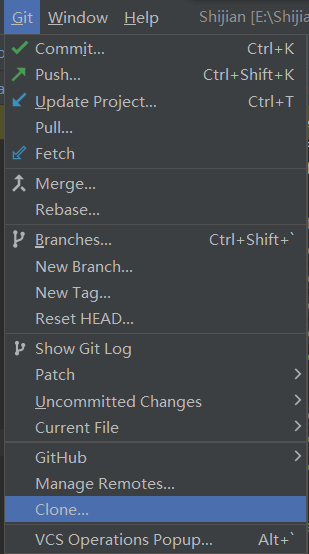
# 6 实施情况

6.1阅读和描述样例工程

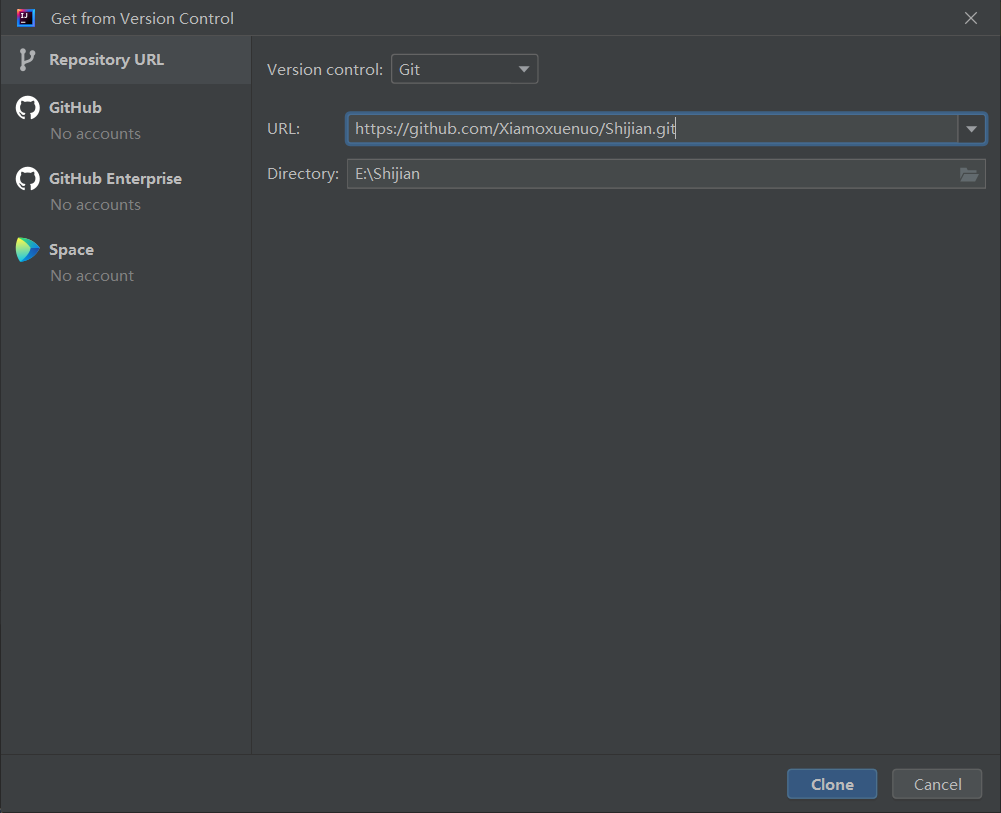
首先在github上fork样例工程，并复制web URL。



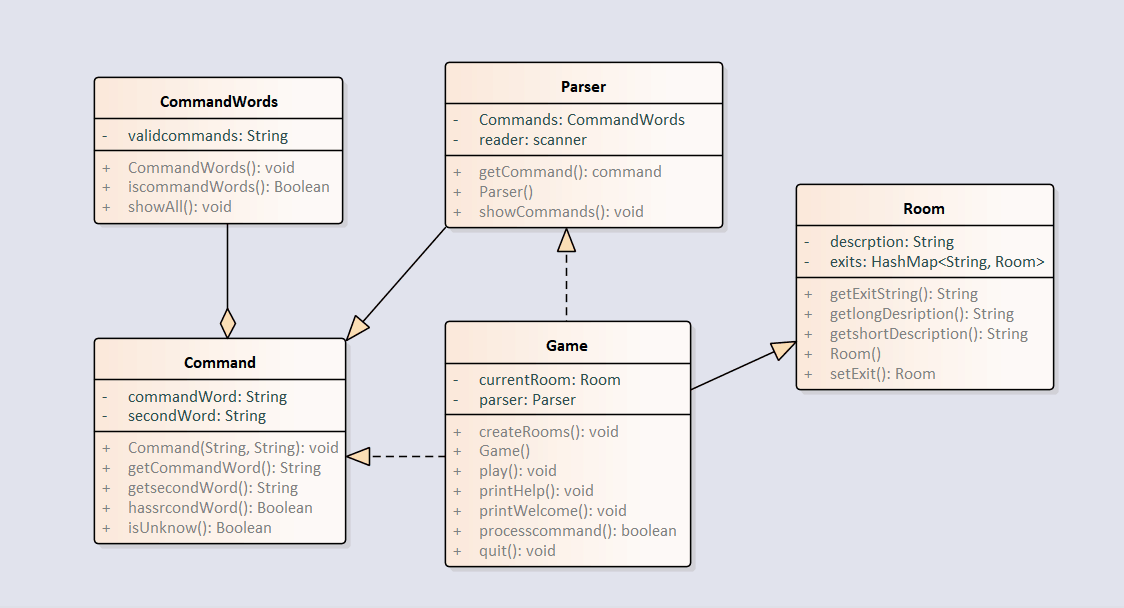
在IDEA中用选择Git-Clone克隆工程。



在弹出的窗口中用URL完成克隆。



精读样例工程软件代码后画出类图。



6.2标注样例工程中的代码

基于javadoc规范标注代码，对包、类、方法、代码片段、参数和语句等代码层次进行注释，结果已提交git。

6.3维护样例工程

在样例代码Game类的processCommand()方法中，当用户输入的命令被辨认出来以后，原本是用一系列的if语句用来分派程序到不同的地方去执行。从面向对象的设计原则来看，这种解决方案不太好，因为每当要加入一个新的命令时，就得在这一堆if语句中再加入一个if分支，最终会导致这个方法的代码膨胀得极其臃肿。我通过一下方法解决了该问题。

首先定义一张哈希图，用来存储命令列表。



初始化时定义哈希表类型的commandList。



通过setCommandList()方法为commandList赋初值。



在处理用户输入的指令时只要在哈希图中匹配相应命令对应的方法执行即可。

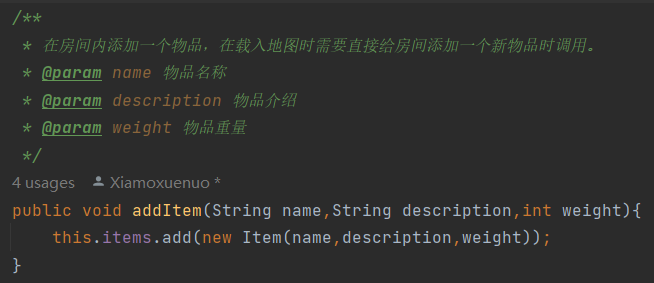


经过这样修改，添加新的命令后执行时不需要if-else语句去匹配，只需要设置一条新的命令，在setCommandList()方法中把命令和方法匹配即可。

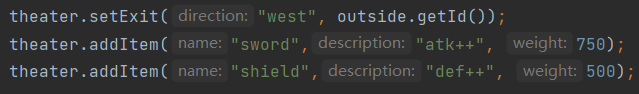
6.4扩充样例工程

6.4.1房间物品和look命令

在Room类中新定义一个方法addItem()用来在载入地图时直接给房间添加物品，添加时需要给出物品名称、物品介绍和物品重量。



在Game类中初始化房间时调用该方法，theater是一个房间。

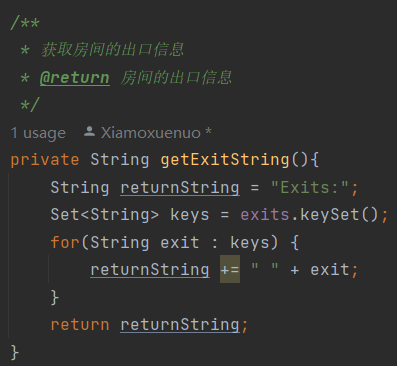


在Game类中定义look()方法，对应look命令，以实现查看查看当前房间的信息以及房间内的所有物品信息的功能。

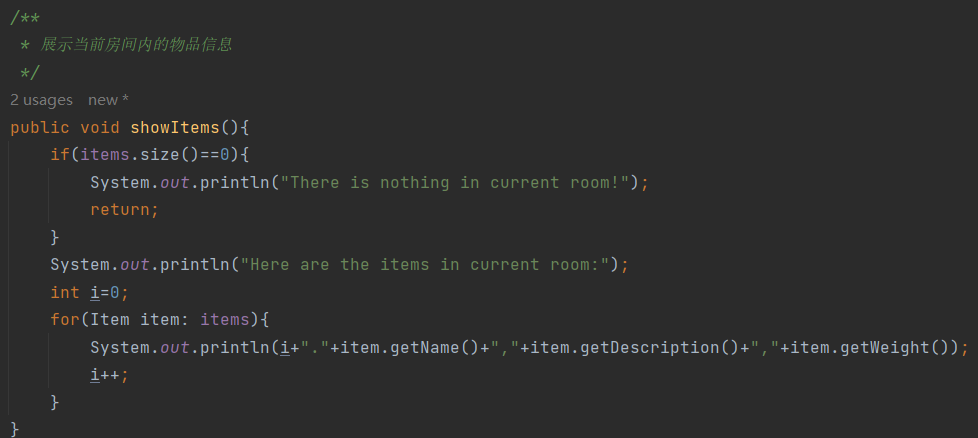


其中getLongDescription()是调用了Room类中的getLongDescription()方法，用来展示房间的完整信息。该方法中调用了Room类的getExitString()方法用来获取出口信息。





look()中的showItems()方法是定义在Room类中用来展示当前房间内物品信息的方法。首先判断该房间中是否有物品，有物品就通过循环打印物品信息，包括物品序号、物品名称、物品介绍和物品重量。获取物品的名称、介绍和重量时进行了封装，再Item类中定义了这三个方法，这里不再展示。



6.4.2高级的back命令

在Game类中定义了back方法，以实现back命令。该back命令是高级的back命令，也就是可以通过重复使用该命令回退多个房间，直到把玩家带回起点。同时还考虑了遇到传送房间的情况，如果上一个房间是传送房间，则会回到传送房间的上一个房间，这样设计是比较合理的。

首先在Game类中定义了双端队列path用以记录经过的房间。



实现重复回退，主要是对已经经过的房间计数，如果经过的房间数是一就说明在起点了，不能继续回退。每回退一个房间，通过removeLast()把最后一个房间移除，并通过getLast()获取当前的最后一个房间并将其设置为当前所在房间。

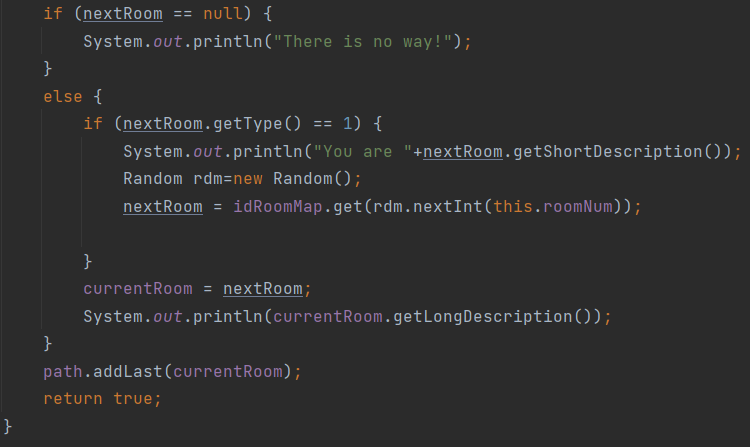


6.4.3传送房间

在初始化房间创建传送房间时，给予其特殊的房间类型，将其type值设为1(普通的房间是0)。



在原有goRoom方法的基础上进行修改



在移动到下一个房间时检查房间类型，如果是1说明进入的是传送房间，那么调用随机函数生成一个随机数。



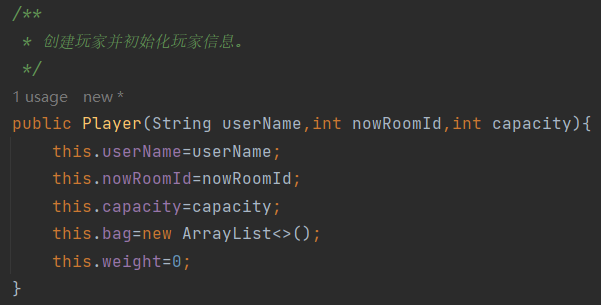
idRoomMap是一个哈希图，通过随机数产生对应的随机房间。

6.4.4独立的Player类

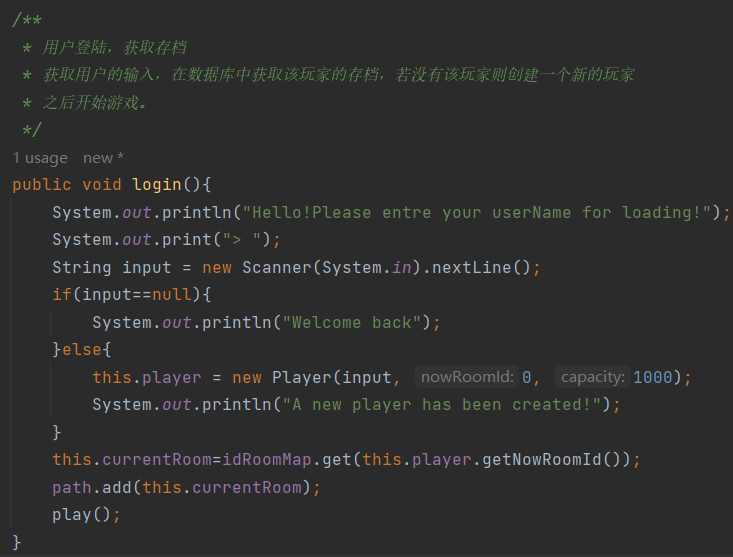
在游戏中新建了一个独立的Player类用来表示玩家，并实现了下列功能需求：

1. 一个玩家对象应该保存玩家的姓名等基本信息，也应该保存玩家当前所在的房间。

Player类中定义的Player对象。



Game类中游戏开始时创建player对象，提示玩家输入玩家昵称并读入。初始负重能力设置为1000。



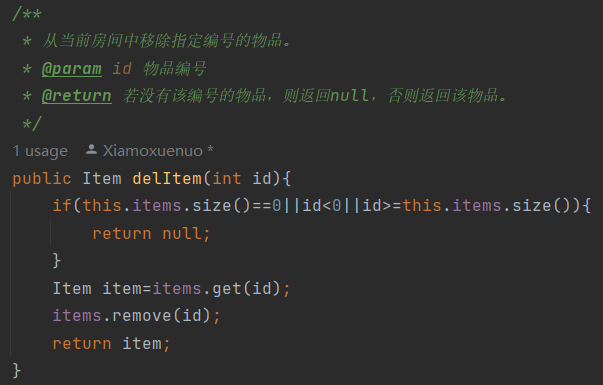
1. 玩家可以随身携带任意数量的物件，但随身物品的总重量不能超过某个上限值。

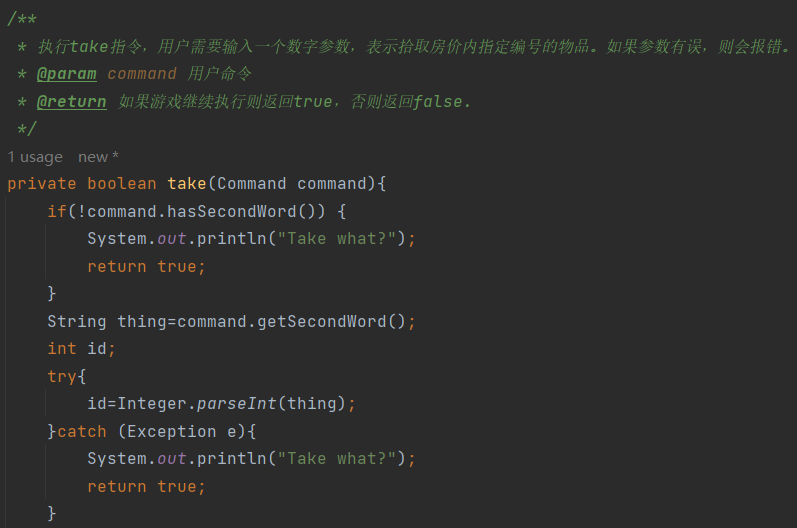


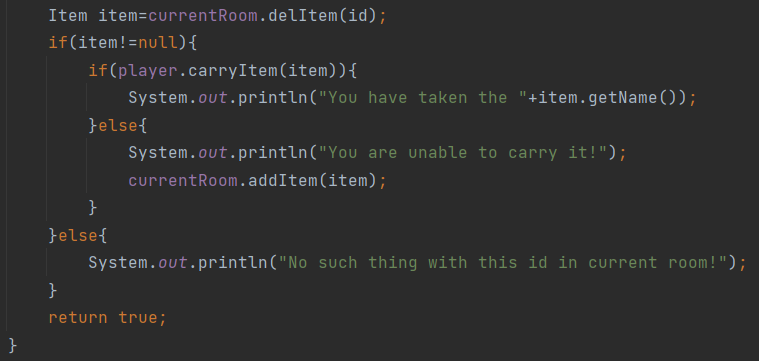
该方法carryItem()用来检查玩家携带的物品总重量是否超过负重能力，如果超过了就返回false，否则将物品放入背包并加上其重量。

1. 在游戏中增加两个新的命令“take”和“drop”，使得玩家可以拾取房间内的指定物品或丢弃身上携带的某件或全部物品，当拾取新的物件时超过了玩家可携带的重量上限，系统应给出提示。

在Game类中定义take()方法，以实现take命令。每个房间中对应的物品会有相应编号，take()通过获取输入的编号确定要拿的物品，然后调用Room类中的delItem()方法将该物品从该房间中删除，然后调用刚才的carryItem()方法检查玩家负重是否超过上限。

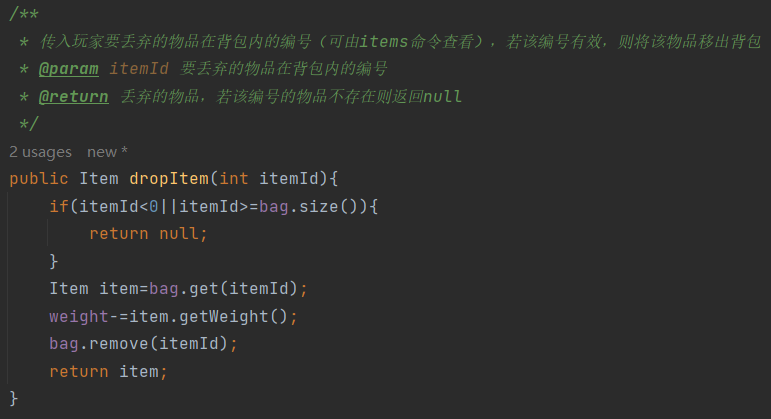


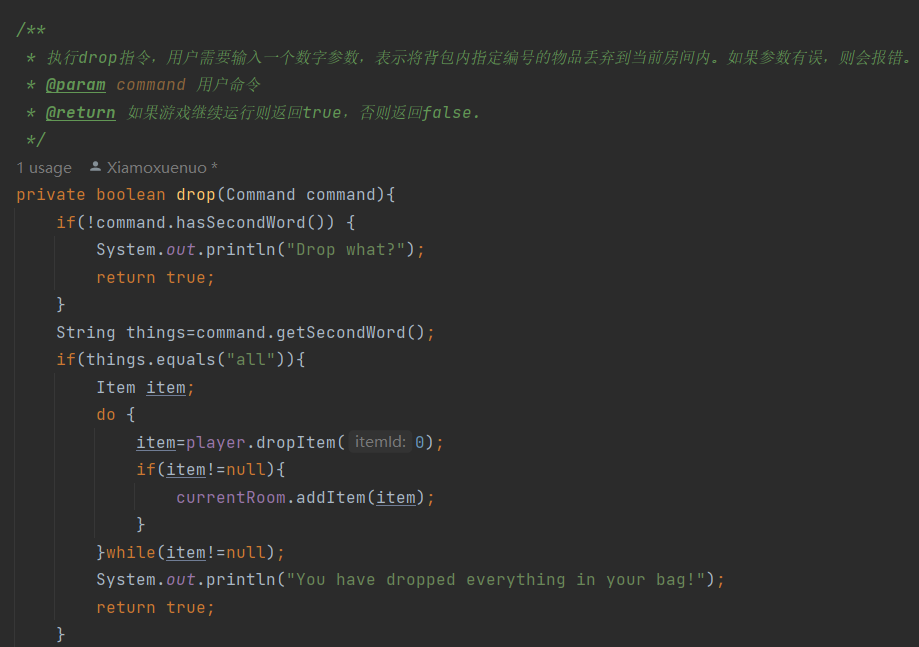


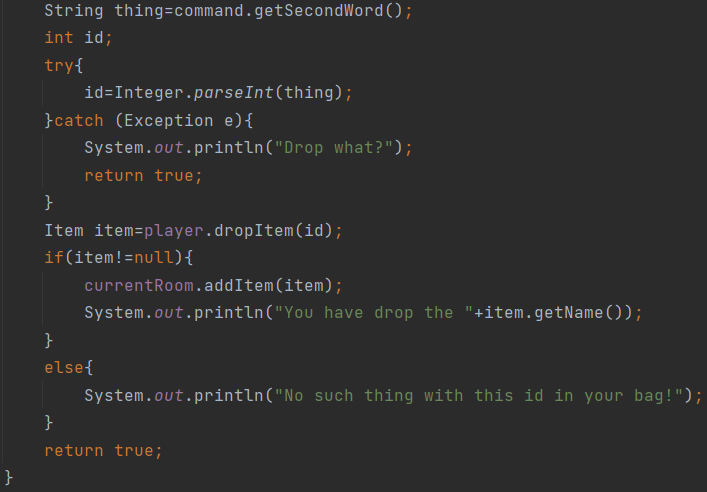


检查的结果有两个，如果过没超过负重上限就提示成功拿走物品，否则提示带不走该物品，并调用上文所述Room类中的addItem()方法将物品放回房间。

在Game类中定义了drop()方法，以实现drop命令。通过读入输入的数字，对应背包内相应物品的编号，调用Room类中的addItem()方法将物品加入该房间中并调用Player类中的dropItem()方法将物品从背包中移除。如果丢弃了所有物品会提示，输入错误的编号也会提示。玩家可以通过输入“all”来一次性丢弃所有物品。

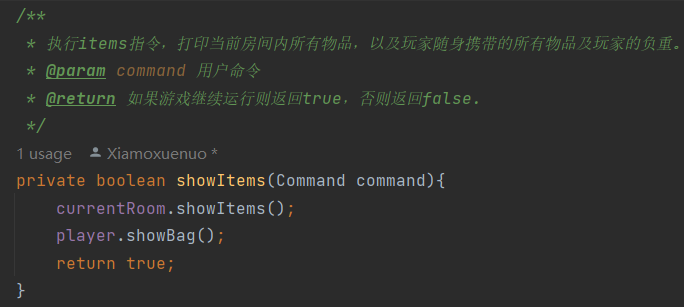




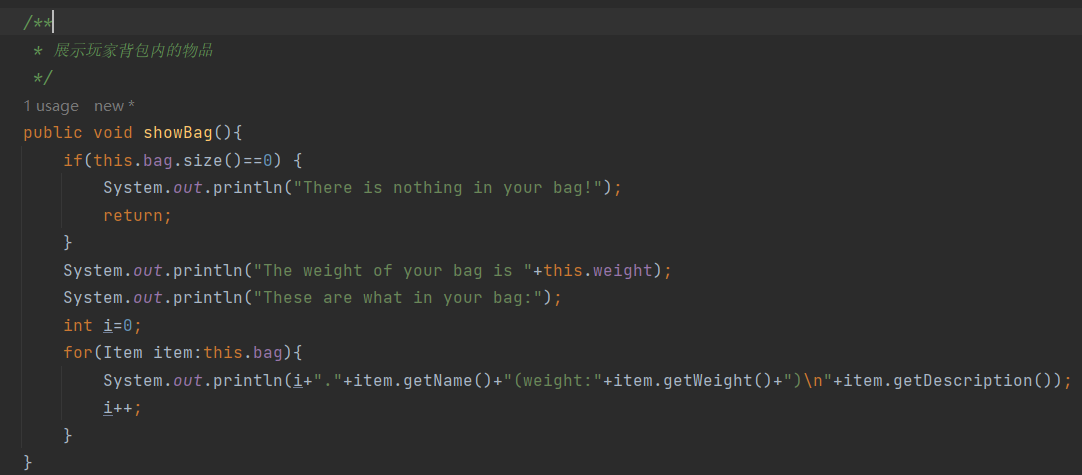


1. 在游戏中增加一个新的命令“items”, 可以打印出当前房间内所有的物件及总重量，以及玩家随身携带的所有物件及总重量。

在Game类中定义了showItems()方法，以实现items命令。



该方法分别调用了Room类的showItems()方法和Player类的showBag()方法来展示当前房间的物品信息和玩家背包中的物品信息，前者前文已经展示过，这里展示后者，其实现逻辑与前者基本一致。



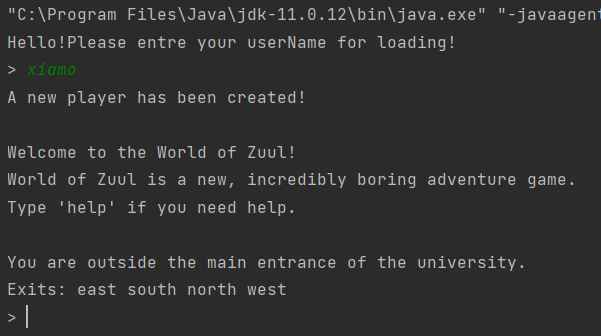
1. 在某个或某些房间中随机增加一个magic cookie（魔法饼干）物件，并增加一个“eat cookie”命令，如果玩家找到并吃掉魔法饼干，就可以增长玩家的负重能力。

在Player类中定义eatCookie()方法以实现eat-cookie命令。玩家输入该命令时检查玩家背包中是否有“magic cookie”，有就将其从玩家背包中删除并增加玩家的负重能力。

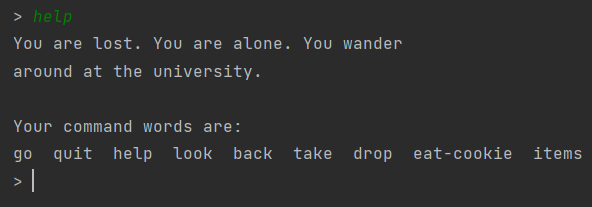


6.5 程序运行结果

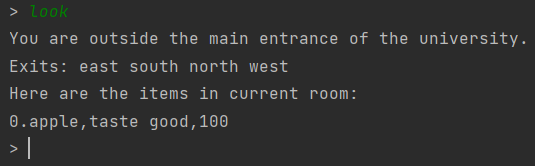
调用Main类以运行程序，首先提示用户输入用户名，创建新玩家后即可开始游戏。



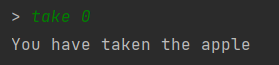
输入help指令可以查看可以使用的命令，包括go,quit,help,look,back,take,drop,eat-cookie,items。



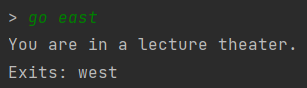
输入look查看当前房间信息和当前房间内物品信息，房间内有一个苹果。



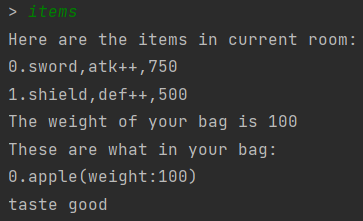
输入take 0拿走苹果。



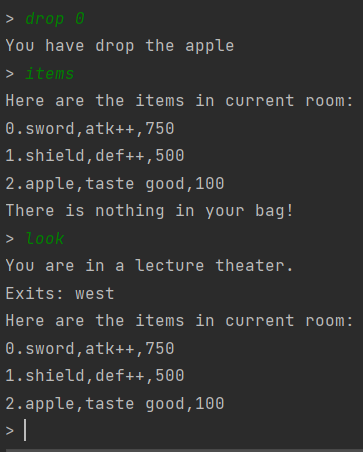
输入go east进入下一个房间。



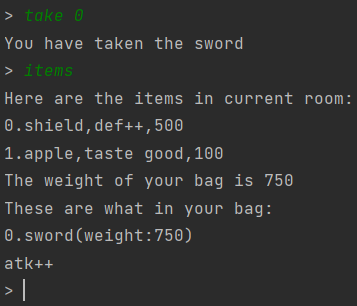
输入items查看当前房间内物品和背包内物品，房间内有一把剑和一个盾牌，玩家背包内有一个苹果。



输入drop扔掉苹果，输入items和look查看，房间内有三个物品，玩家背包内没有物品。



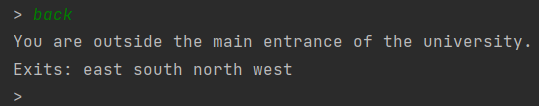
输入take 0拿走sword，输入items查看，房间内有盾牌和苹果，玩家背包中有剑。由于物品减少，物品会自动重新编号。



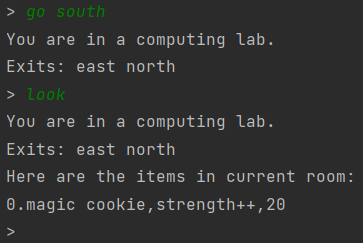
输入take 0拿走shield，提示超过负重能力。



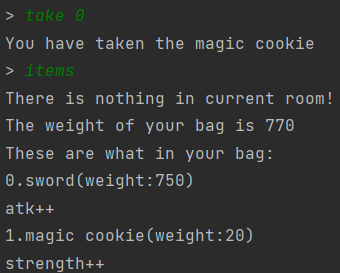
输入back，回到上个房间。



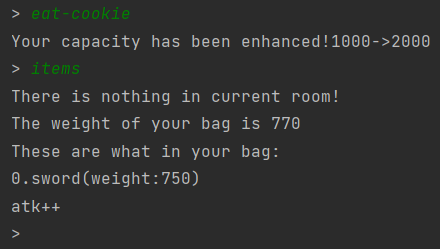
输入go south进入下一个房间，输入look查看。



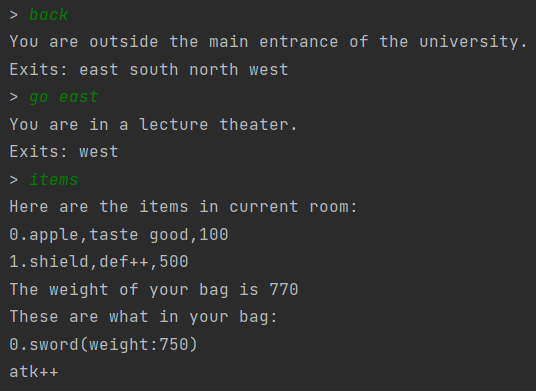
输入take 0拿走魔法饼干，输入items查看背包物品，有魔法饼干。



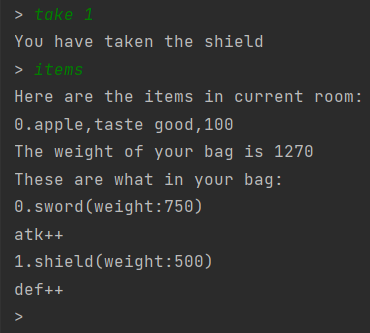
输入eat-cookie吃掉饼干，提示负重能力增加。输入items查看物品，魔法饼干已经消失了。



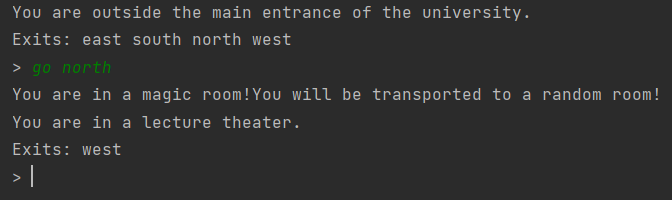
回到lecture房间。



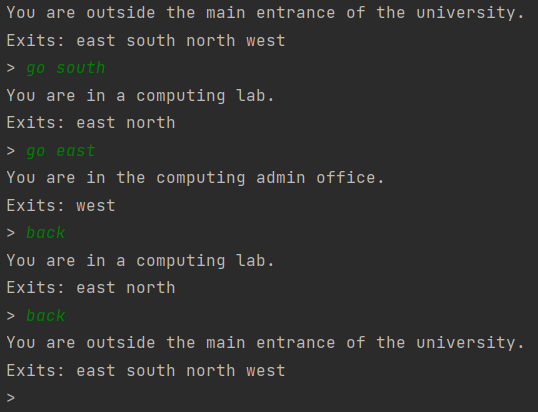
输入take 1拿走盾牌，此时不会再提示超过负重能力，成功拿走，此时玩家背包既有剑也有盾牌。



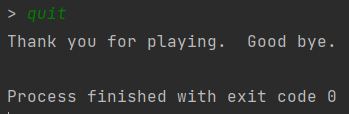
回到初始房间后输入go north，进入了传送房间，被随机传送到一个房间。



输入go west回到起始房间，然后输入go south，再输入go east，再输入两次back，成功回到出发点，实现了逐层退回。



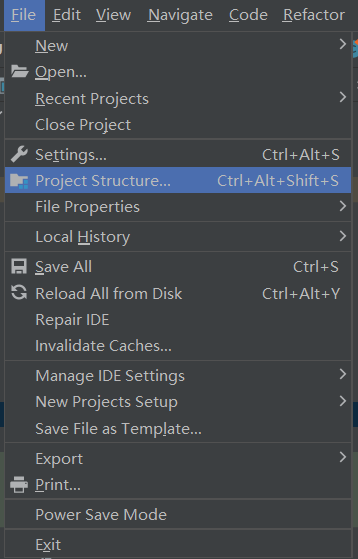
输入quit退出。



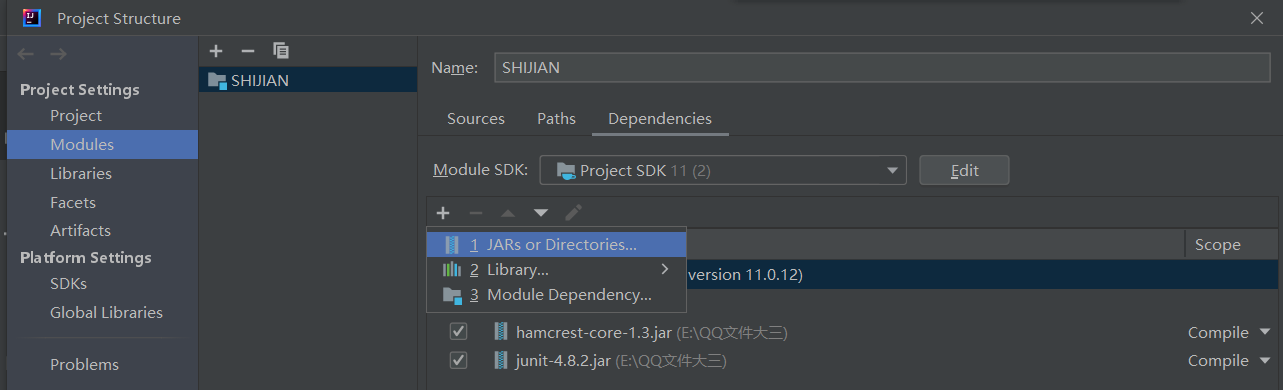
6.6 测试结果分析

6.6.1测试方法

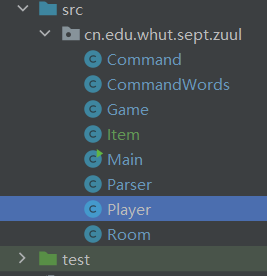
首先在IDEA中导入jar包。选择File中的Project Structure选项。

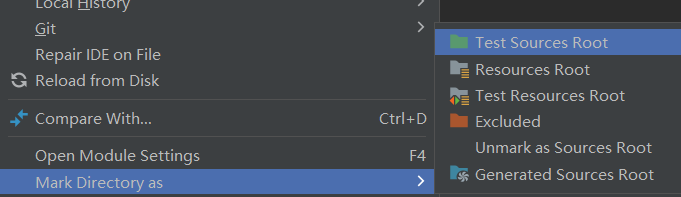


选择Modules-Dependencies，选择+号导入JARs or Directions导入本地的jar包。

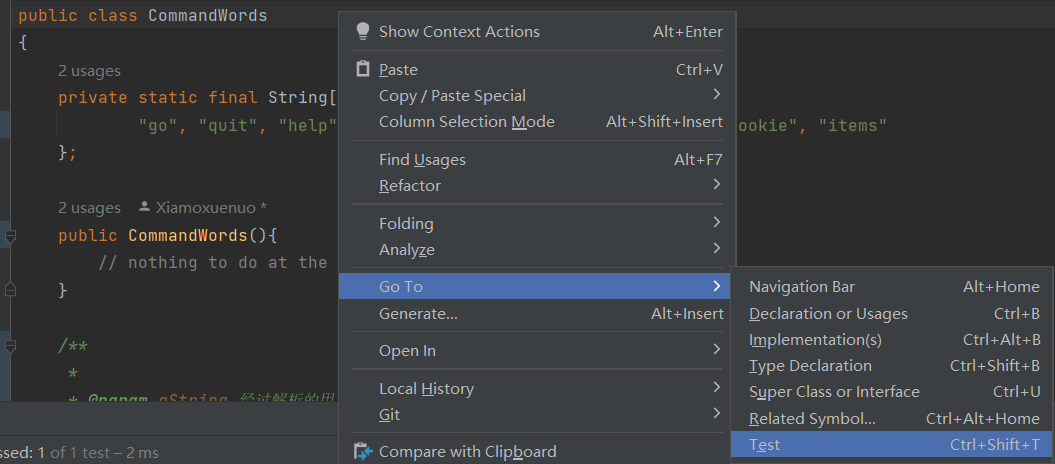


新建一个与src同级的文件夹保存测试代码，右键该文件夹，将其设置为测试文件根目录。

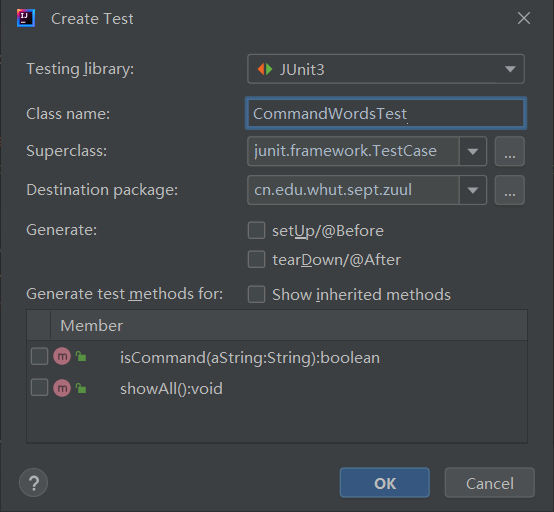


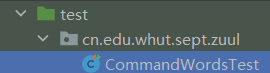


右键需要测试的类，选择Go To-Test



勾选需要测试的方法，得到测试类。

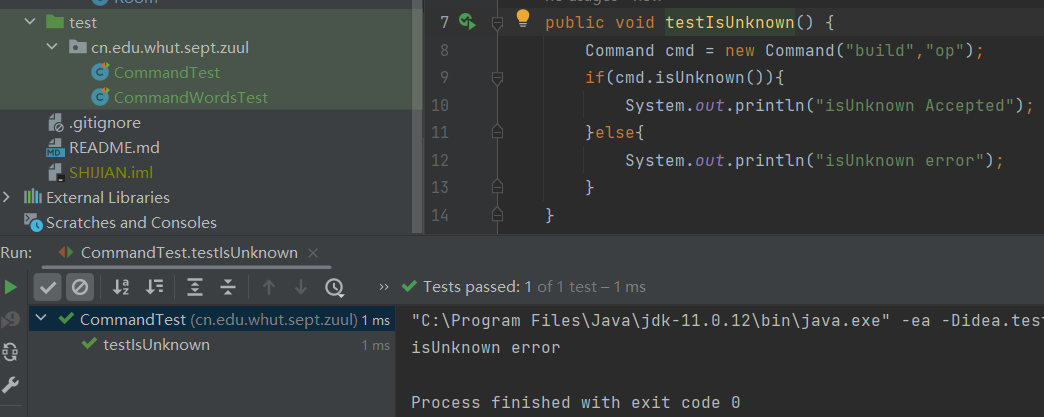




在测试类中编写测试用例，之后右键要测试的方法即可运行测试。

6.6.2测试Command类

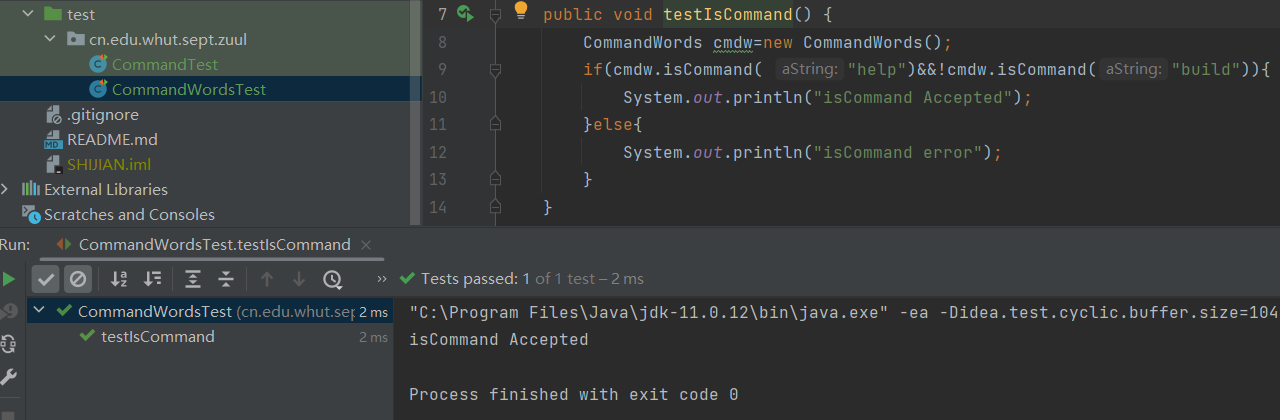
1.在CommandTest类中对isUnknown()方法进行测试。



build不是有效命令，故测试结果正确。

6.6.3测试CommandWords类

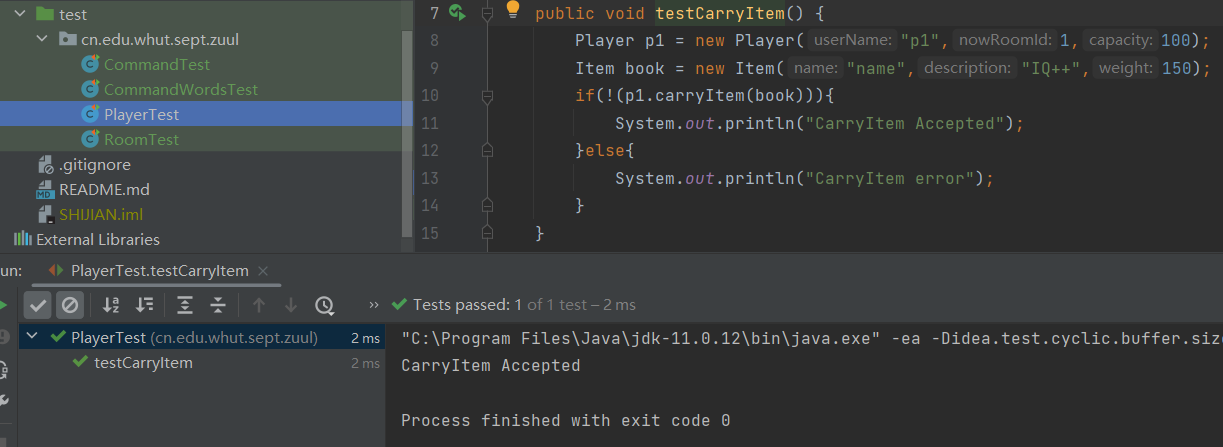
1.在CommandWordsTest类中对isCommand()方法进行测试。



help是有效命令而build不是有效命令，故测试结果正确。

6.6.4测试Player类

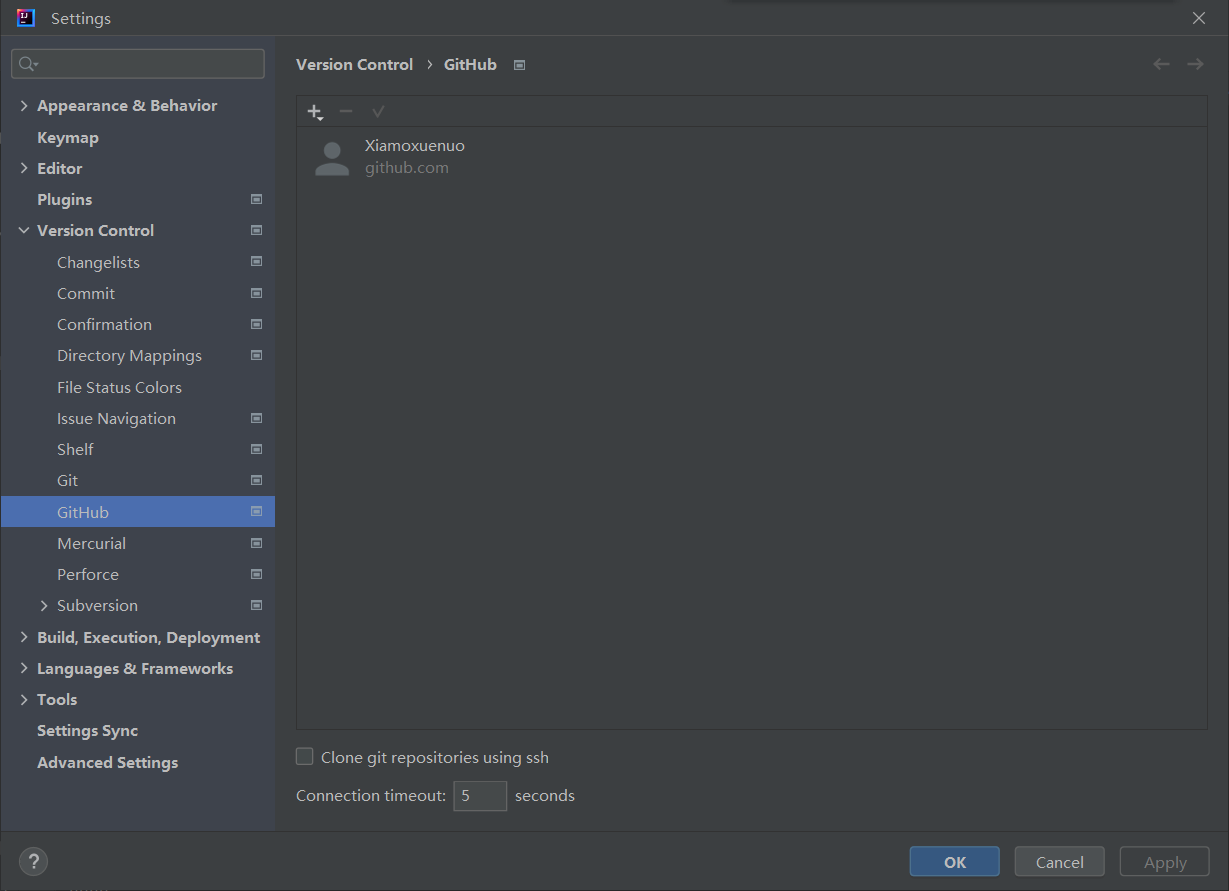
1.在PlayerTest类中对CarryItem()方法进行测试。



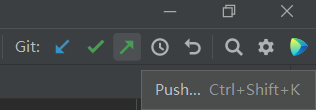
book的重量为150，超过了p1的承重能力100，故函数返回值应为false，取反则为true，测试结果正确。

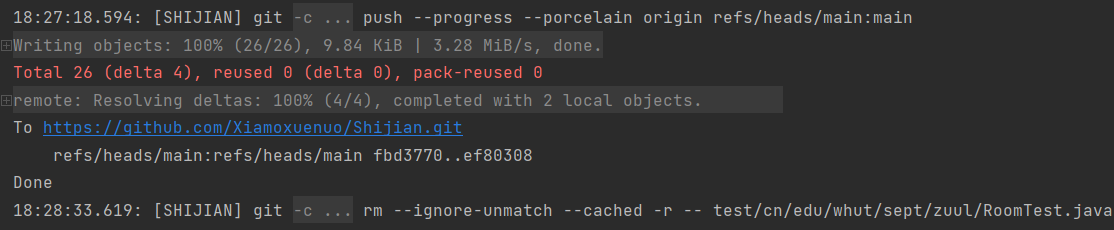
6.7 代码提交状况

在GitHub中新建token，在IDEA中用token登陆GitHub账号，登陆后可以在Settings-Github中看到自己的账号。



点击右上角的push，成功在IDEA中向GitHub提交代码。





# 7 实施过程问题记录与分析

一开始使用Eclipse进行项目开发，在开发时遇到了自编译速度慢、无法识别Java文件、无法创建包文件等等很多问题，浪费了大量的时间。最后JDK又出现了问题，决定更换采用IDEA重新开发，使用IDEA配置JDK没有任何问题，而且经过一段时间的使用，IDEA使用起来要比Eclipse方便许多。

# 8 任务总结

通过本次实践任务，我理解了软件代码规范的重要性和代码变化对软件质量带来的影响，掌握了掌握基于Git的个人代码版本维护方法和MarkDown文件编写方法。通过对GitHub的代码进行fork和clone，以及后续的提交，我对使用GitHub更加熟练了。通过基于javadoc规范对代码进行标注，我第一次学习了规范的注释方法，这对我以后的编程工作都有很大的帮助。通过对样例代码进行扩充，我的Java编程能力得到了很大提高。通过编写测试用例对程序进行单元测试，结合了软件工程所学知识，我对如何科学地完成一个工程有了更深刻的理解。这次实践任务全面地提高了我的编程能力，我系统地了解了编写一个项目有哪些需要注意的地方，也使我意识到要做好编程工作单有编程能力是不够的，还要注意许多规范和编程以外的工作，在以后的时间中我还要继续努力学习，提高自己。很感谢这次实践任务，我受益匪浅。

# 9 参考文献

**[1]** [邵维忠](https://book.douban.com/search/%E9%82%B5%E7%BB%B4%E5%BF%A0),[麻志毅](https://book.douban.com/search/%E9%BA%BB%E5%BF%97%E6%AF%85),[马浩海](https://book.douban.com/search/%E9%A9%AC%E6%B5%A9%E6%B5%B7).UML用户指南[M].人民邮电出版社.2013-1.

**《软件工程实践（一）》成绩评定表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓 名** | | **闫瀚** | **学 号** | **0122004950122** | |
| **专业、班级** | | **软件2001** | | | |
| **成绩评定：** | | | | | |
| 评价内容 | | | 满分 | 实得分 | |
| 得分 | 小计 |
| 实践任务  完成情况 | 软件项目设计、改进与扩充 | | 20 |  |  |
| 个人软件过程与项目管理 | | 15 |  |
| 代码版本管理 | | 25 |  |
| 代码注释与编码规范 | | 25 |  |
| 单元测试 | | 15 |  |
| 实践报告  总评情况 | 学习态度与考勤 | | 10 |  |  |
| 报告格式的规范性 | | 10 |  |
| 报告的逻辑结构与语言表达 | | 15 |  |
| 实践内容的正确性与合理性 | | 60 |  |
| 文献引用及标注 | | 5 |  |
| 总分 | | | 100 |  |  |
| **最终评定成绩（以优、良、中、及格、不及格评定）** | | | | | |

**指导教师签字：**

**年 月 日**