|  |  |
| --- | --- |
| **学 号：** | 0122006290904 |

****

**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** | **张宸** |
| **编 号** | **0122006290904** |
| **指导教师** | **唐祖锴** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2023 | 年 | 1 | 月 | 2 | 日 |

**目录**

[1 任务概述 1](#_Toc91495513)

[2 任务分析 1](#_Toc91495514)

[3 开发计划 1](#_Toc91495515)

[4 软件配置计划 1](#_Toc91495516)

[5 测试计划 1](#_Toc91495517)

[6 实施情况 1](#_Toc91495518)

[7 实施过程问题记录与分析 2](#_Toc91495519)

[8 任务总结 2](#_Toc91495520)

[9 参考文献 2](#_Toc91495521)

# 1 任务概述

1.1实践的主要内容

通过中小型任务循序渐进开展实践，以个人为开发单位，采用迭代递进的实践过程逐步掌握课程内容。

1.2项目的主要内容

于程序设计方法，结合样例或自选题目，完成开发案例的设计、功能点扩展和功能实现。基于软件工程开发流程，针对设计任务采用适当的个人软件过程与项目管理工具进行项目流程管理。

1.3系统的主要内容

样例工程代码world-of-zuul是一个基于字符界面的探险游戏，这个游戏目前只具备一个基本的功能框架，需要你对其进行完善和扩充。

原版的world-of-zuul是由 Will Crowther在20世纪70年代开发、并经过 Don Woods扩充的一个探险游戏。它有时候也叫作巨洞探险（ Colossal Cave Adventure），在那个时代是一个精彩的、充满想像力的复杂游戏，包括要在一个复杂的洞穴系统中找到出路、寻找隐藏的财宝、使用暗语和其他一些神秘工具，最终的目的是获得高分。

1.4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

1.5具体实验步骤

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  
     
   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；  
     
   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。  
     
   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；
2. 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；  
     
   5. 编写测试用例  
    （1）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

2.1任务分析概述

本实验是基于world-of-zuul基础框架程序对软件展开迭代开发，首先应该对初始的游戏框架以及代码进行细致的阅读以及适当的代码注释以及功能分析，这有助于让我们对整个项目以及代码有初步的认识。再通过对框架以及代码的理解，使用UML建模将相应的用例图，类图绘制出来。进一步的熟悉整个项目。接下来再进行最重要的迭代开发，通过分析新的需求与功能，同时利用IntelliJ IDEA编写JAVA代码来完善整个项目以及功能扩充点。最后再设计相应的测试用例，结合《软件工程基础》课程学习的软件测试的知识，设计测试用例，并完成相应的测试，根据测试的结果进行补充和完善。

2.2工作重点

（1）UML建模设计类图

（2）迭代开发实现功能扩展点

（3）设计测试用例并测试系统

2.3工具使用方案

UML建模：Enterprise Architect

迭代开发：IntelliJ IDEA

软件测试：Junit

2.3重难点分析

1. 项目规模不小，所以在UML建模过程中，类与类之间的复杂关系难以完整描述，需要长时间的分析与长时间的建模。
2. 功能扩展点逻辑复杂，首先得对已有部分进行详细的认识，避免冗余代码过多，其次要分析新添功能的逻辑，再进行代码的编写与分析测试，根据基本测试的结果与预期结果进行对比，来看哪些地方需要更改。
3. 软件测试的难点是如何针对系统设计测试用例，并利用Junit创建测试类进行测试。

# 3 开发计划

（1）阅读代码并代码中实现的方法添加相应的注释，并利用Enterprise Architect画出类图建模。

（2）创建markdown文件REPORT，利用markdown的语法来记录软件开发过程。

（3）使用IntelliJ IDEA Community Edition 2022.1.1来进行软件开发，实现功能拓展点的代码实现。

（4）在对代码更改、添加之后上传至Github来进行版本管理与记录。

（5）开发完成后，设计相对应的测试用例，利用Junit进行代码测试。

# 

# 4 软件配置计划

1. 版本编码规范：利用Version.\*.\*来对版本进行编码。

（2）命名规范：利用骆驼式命名法（Camel-Case），这样让方法的名字看起来通俗易懂，方便别人进行理解。

（3）分支管理规范：每次迭代上传至Github，迭代都在sept1-2583759442-master目录下更新。

（4）提交规范：每次更新后的代码以及更新的简介

# 

# 5 测试计划

在实验过程中使用Junit对代码进行单元测试。主要针对Command类、CommandWords类以及Player类进行测试，测试的内容是除了一些基本的get\*函数以及show\*函数。对每一个类建立其测试类：\*Test类，然后通过Junit测试其功能。

# 

# 6 实施情况

**6.1 阅读和理解样例代码**

**6.1.1 fork样例工程，并clone到本地仓库**

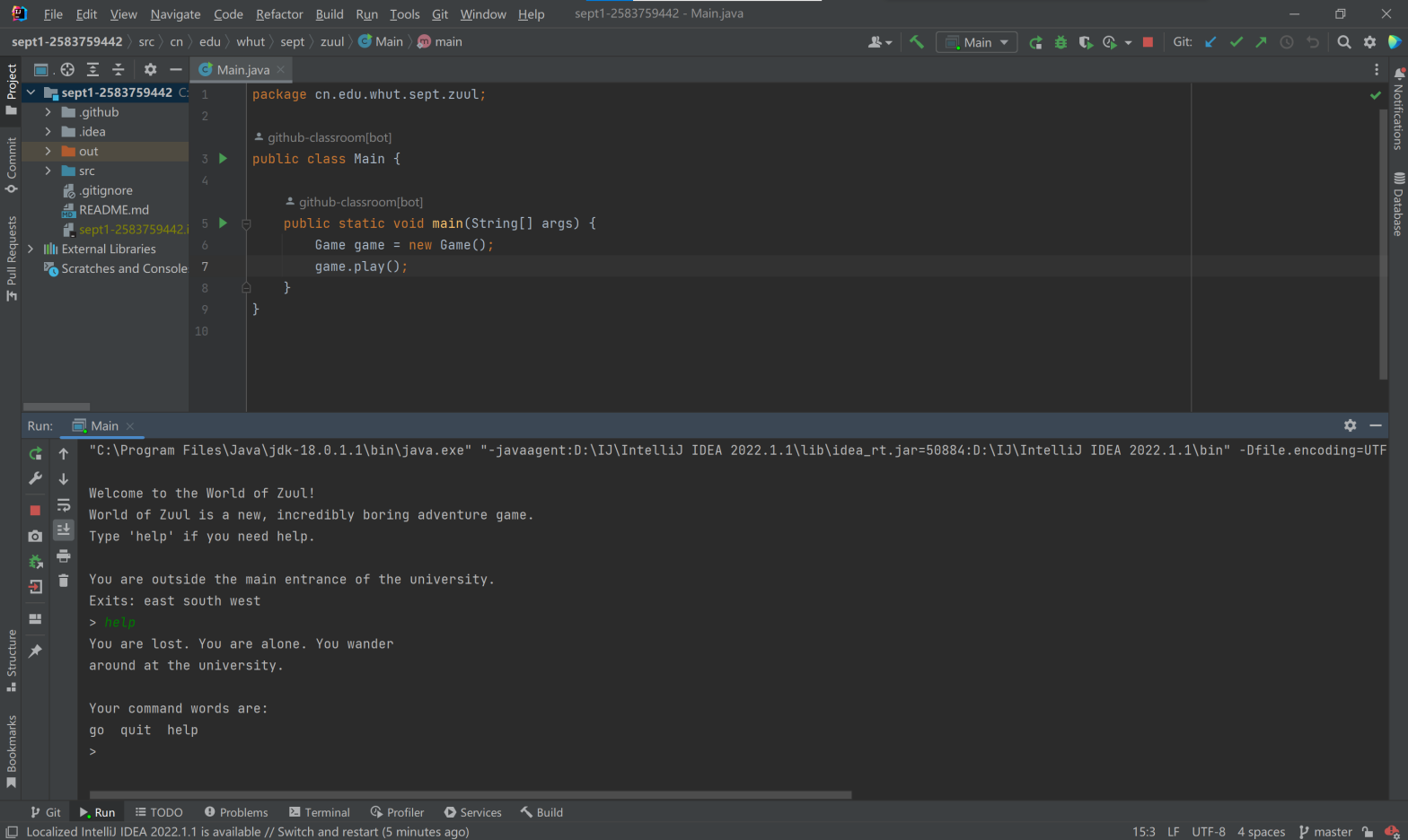
先将对GitHub上的项目进行fork操作将他创建一个相同的项目在我的账号中，然后再Windows PowerShell中根据其HTTPS进行git clone操作，将其下载至指定路径下。



利用PowerShell对GitHub的项目进行clone操作

**6.1.2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**

根据使用help功能，可以看到这个初始框架程序可以使用三个功能：go、help、quit。并且在使用最主要的功能go时可以发现，游戏基本的地图与框架以及基本完成，真正的任务是接下来对功能的完善与添加。



运行后调用help功能

**6.1.3精读样例工程软件代码，描述代码结构及部件组成**

首先根据项目的目录可以看到，基础的系统框架中一共有六个类，其中一个是Main。其主要功能作用的是：Command类、CommandWords类、Game类、Parser类以及Room类。除了一些基础的展示以及获取功能，这个框架已经将游戏的地图给写好了，只需要添加新功能即可完成游戏系统的编写。

这个框架实现了这个系统最基础的三个功能：go、help、quit功能。

go功能：玩家可以输入go \*（\*代表可走的方向）来进入新的房间。

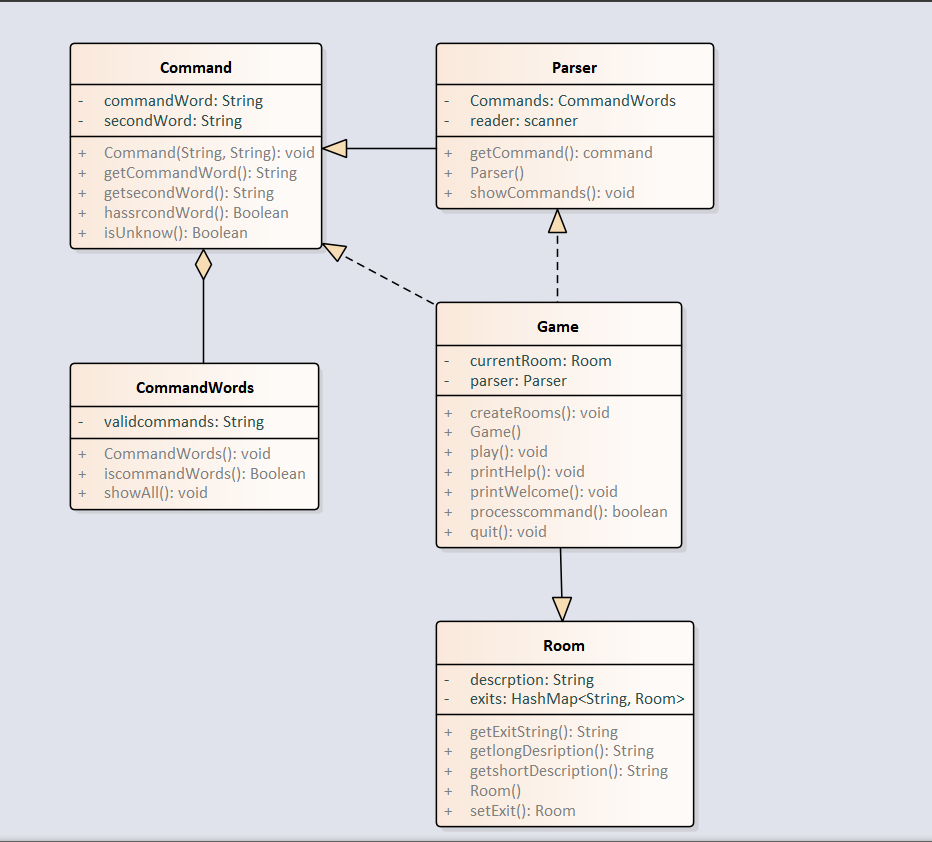
help功能：系统会显示现在这个游戏的基本背景以及，现在已有的命令集。

quit功能：输入这个功能可以退出运行，结束游戏进程。

**6.1.4以UML图描述样例工程的组成及结构图（类及类之间的关系）**

通过UML建模软件以及我们对代码阅读后我们可以得到这个基础框架，以及这个系统目前已有的五个类他们自己的属性以及方法，同时也可以看出这几个类之间的关系。

由此我们可以得到系统的类图，如下：



系统的类图

**6.2标注样例工程中的代码**

**6.2.1基于javadoc规范标注代码**

我们先通过Game类里的Javadoc规范来进行学习如何注释代码，我截截取了Game类里的方法processCommand()的代码注释来为例子。



processCommand()的代码注释

**6.2.2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**

通过GitHub上的Add file —> Upload file讲写好注释后的代码文件上传至GitHub。



上传后GitHub上的代码

**6.3改进维护样例工程**

**6.3.1对功能设计进行分析，找出设计缺陷和改进点，并进行修正或扩充**

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。

通过以下方法来解决上述问题。

基于哈希表的 Map 接口，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。

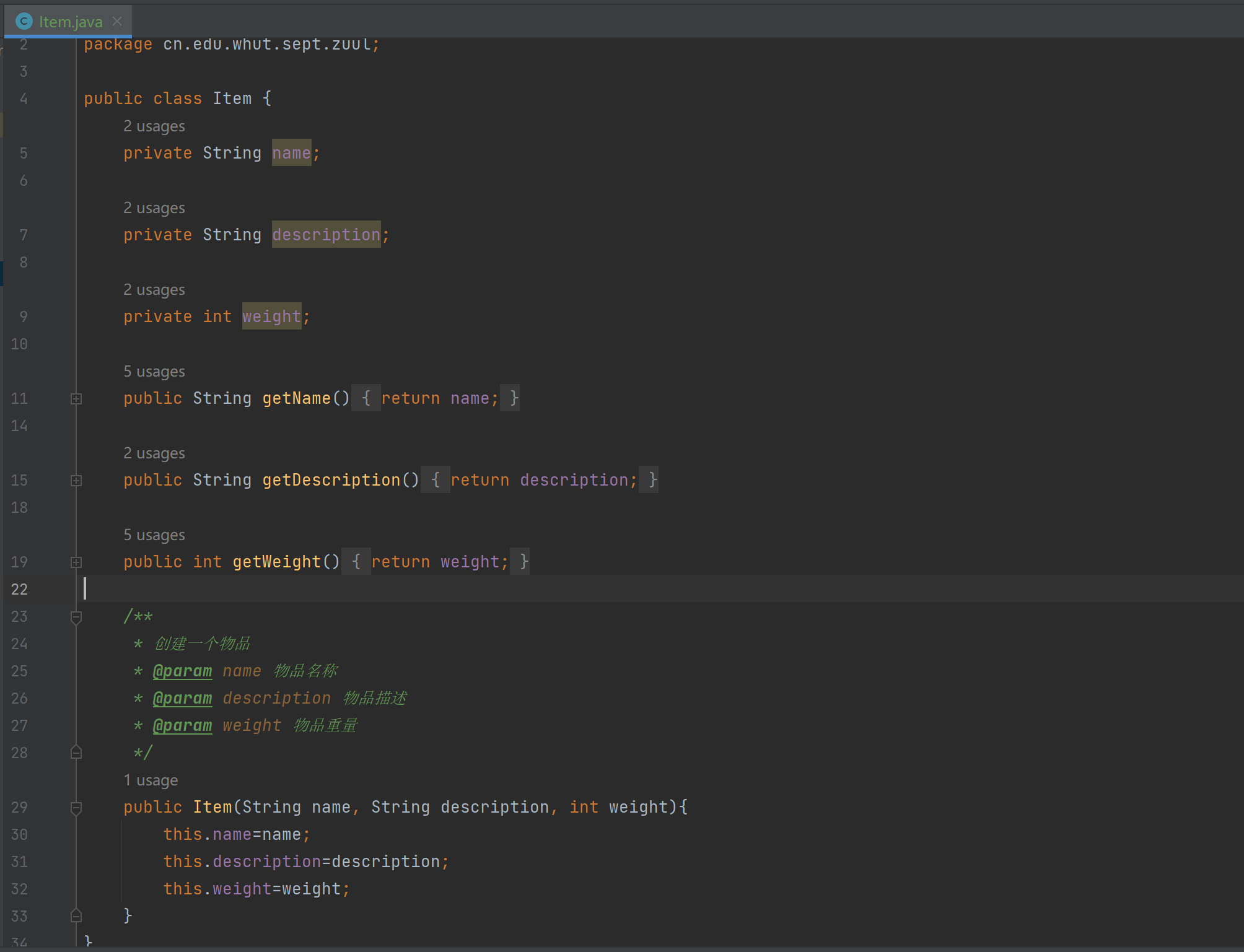


经过这样修改，添加新的命令后执行时不需要if-else语句去匹配，只需要设置一条新的命令，在setCommandList()方法中把命令和对应的方法匹配即可。这样就减少了if-else带来的代码冗余和臃肿，这样也让代码的结构显得更加清晰。

**6.4扩充样例功能**

**6.4.1 Item类以及look()方法**

编写可一个Item类用于存储房间内物品的信息，Item()用于存储房间内或是背包内物品的信息。



Item类

在Room类中定义了AddItem()方法，用于在当前房间内添加一个物品，在载入地图时需要直接给当前房间添加一个新物品时调用。



AddItem()方法

在Game类中定义了look()方法，此方法可以根据用户命令查看当前房间的基本信息以及存在于房间内的物品信息。分别执行了getLongDescription()方法和showItems()方法。

getLongDescription()：介绍进入的房间，每个房间有一段介绍，输出对应的房间介绍。

showItems()：输出房间内有的物品信息，若没有物品则提示没有物品，若有则输出房间内有的物品信息。



look()方法

**6.4.2 Player类**

在系统中新建了Player类来表示玩家每个玩家有这些属性：userName、nowRoomId、bag、capacity和weight。因为新增加的Player，应该适配添加许多新增功能 （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；（2）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；（3）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；（4）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；（5）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。

Player类里定义了下列方法：

public Player(String userName,int nowRoomId,int capacity)

public boolean carryItem(Item item)

public Item dropItem(int itemId)

public void showBag()

public boolean eatCookie()

public String getUserName()

public int getNowRoomId()

public int getCapacity()

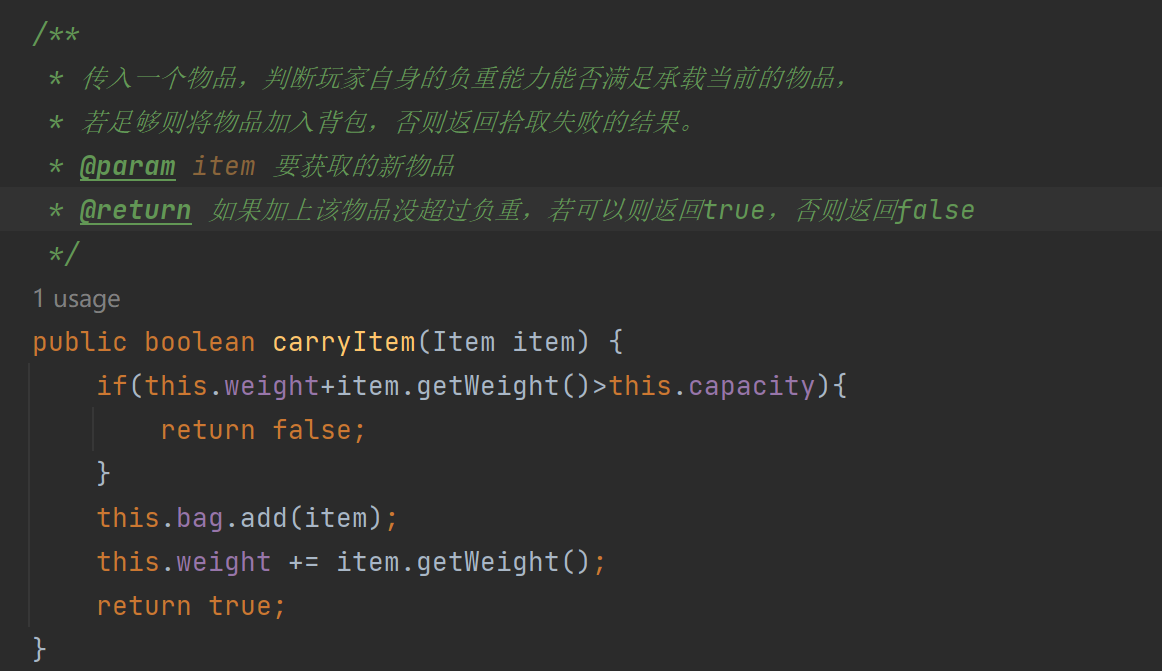
上述方法的具体功能在代码中有详细且符合规范的注释。

1. 因为有了Player，所以可以设计一个login()方法来根据不同玩家进行不同的登录，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。所以在Game类中编写了login()方法。



login()方法

1. carryItem()方法的作用是为了让玩家可以背负一定数量的物品，每个玩家都有自己可承受的重量，每次获取物品时都该判断是否合法没超过玩家的负重能力。



carryItem()方法

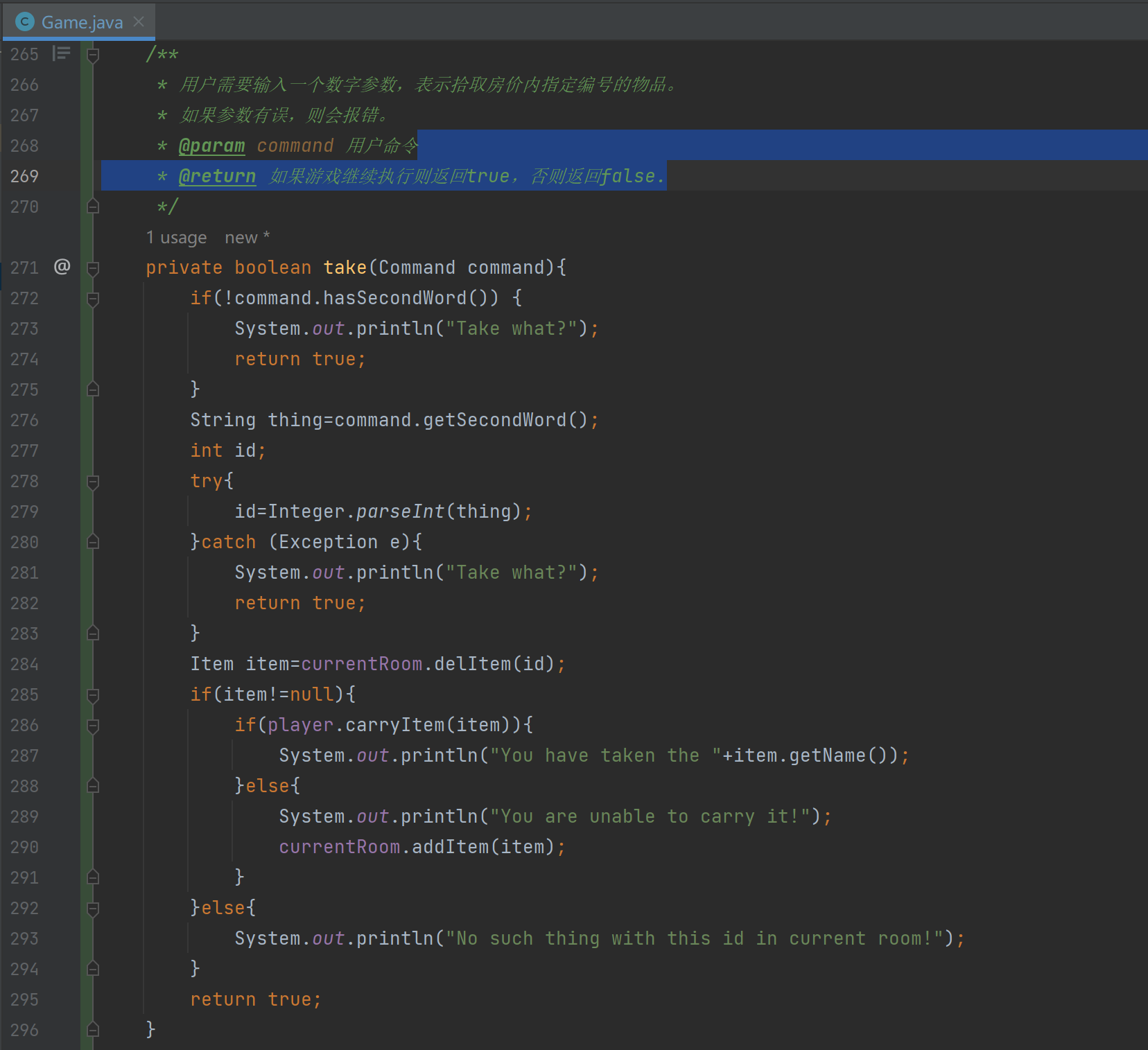
（3）某些房间可能随机产生一个magic cookie物件，并增加一个“eat-cookie”命令，玩家想要提高自身的负重能力，就要找到并吃掉魔法饼干。定义一个eatCookie()方法来实现eat-cookie命令。玩家输入该命令时检查玩家背包中是否有magic cookie，有就将其从玩家背包中删除并增加玩家的负重能力。



eatCookie()方法

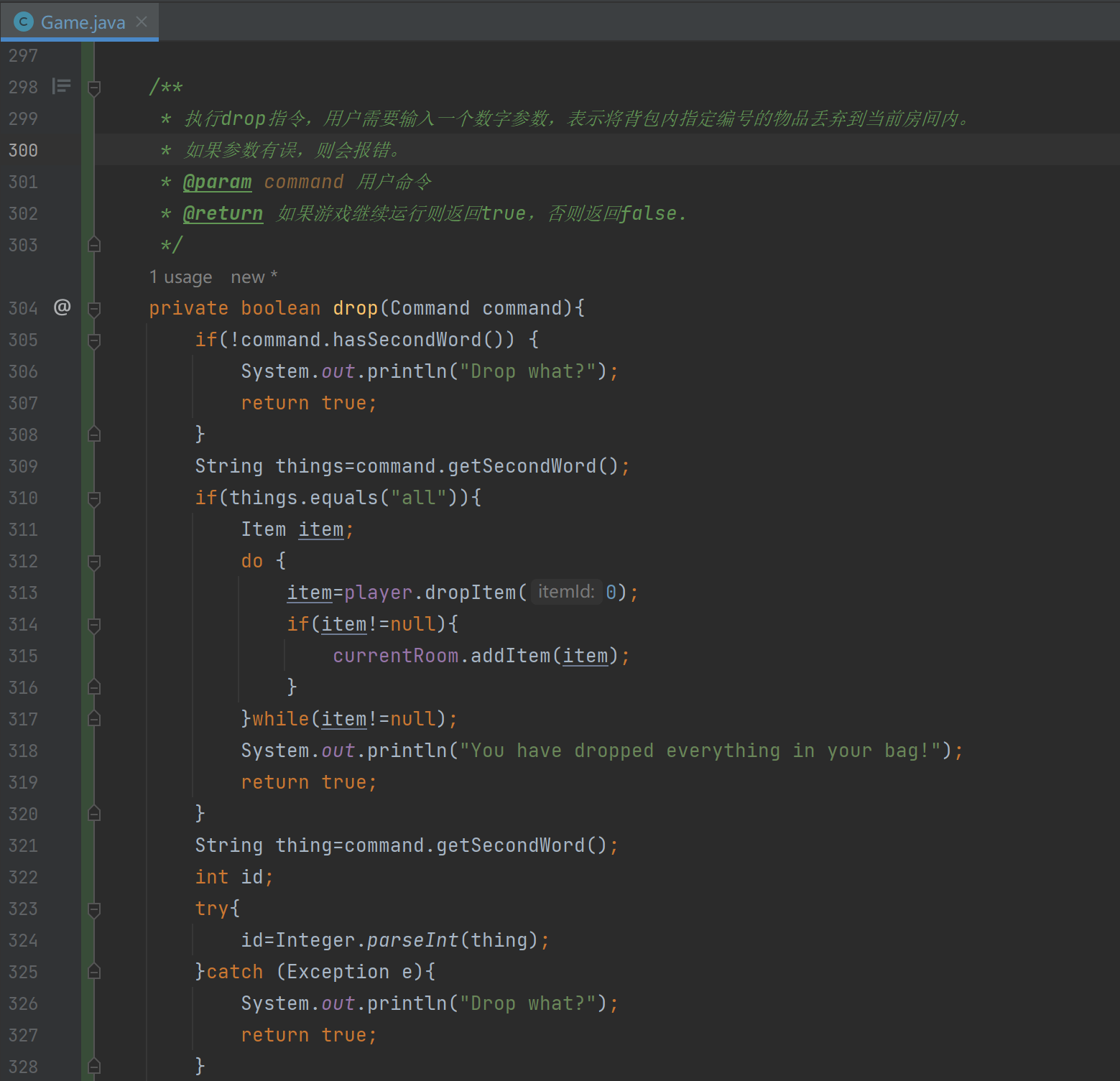
（4）在游戏中增加两个新的命令"take"和"drop"，使得玩家可以拾取房间内的指定物品或丢弃身上携带的某件或全部物品，当拾取新的物件时超过了玩家可携带的重量上限，系统应给出提示。

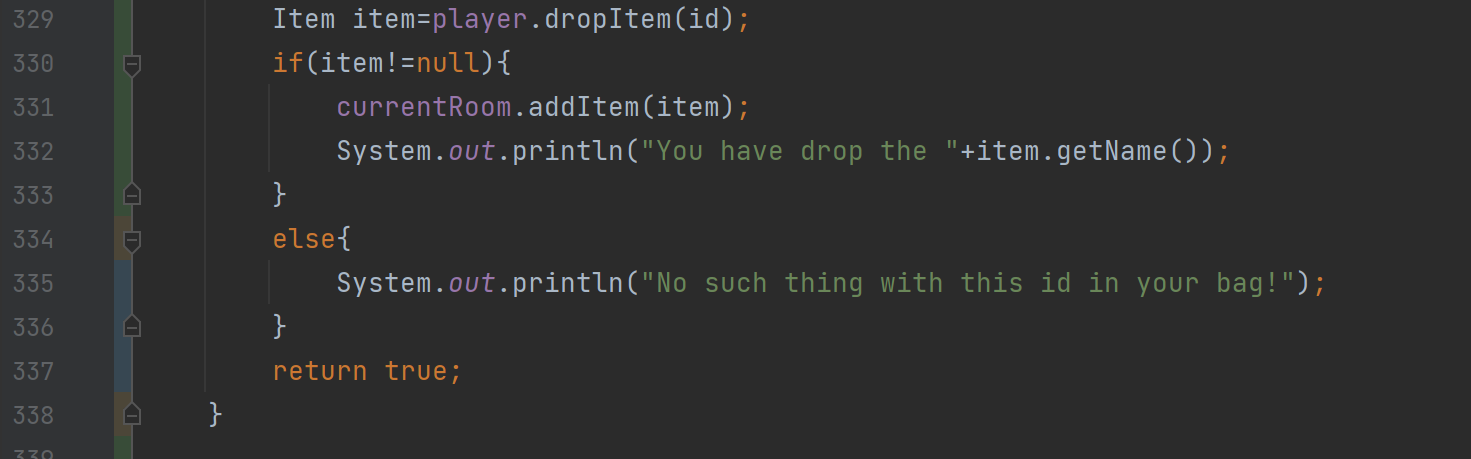
take()方法是实现用户用于输入房间内有的物品的方法。



take()方法

drop()方法是用于将玩家背包中的物品丢掉的方法。





drop()方法

**6.4.3 back命令**

在Game类中定义了back方法，这里的back命令是实验安排中提到的高级的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。这个back指令同时还考虑了遇到传送房间的情况，如果来到此房间是因为上个房间传送的原因，则会回到传送前所在的房间，这样设计是较为合理且逻辑清晰的。

首先在Game类中定义了deque双端队列path用来记录经过的房间。如果新进入一个房间就将房间加入队列，如果倒退一个房间就将此房间出队。

实现所谓的高级back也就是重复回退，主要是在游戏过程中加一个累加器，每次进入一个房间累加器就加一，这样就知道以及进过几次房间了，相反倒退一次累加器中的数字就减一，如果经过的房间数是1就说明在起点了，不能继续回退。每倒退一个房间，通过removeLast()把最后一个房间移除，并通过getLast()获取当前双端队列path中的的最后一个房间并将其设置为当前玩家所在房间。



back()方法

**6.4.4 传送功能goRoom()方法**

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添加传送的部分代码。

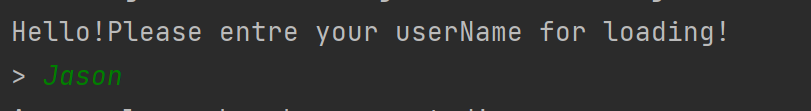


goRoom()方法传送部分代码

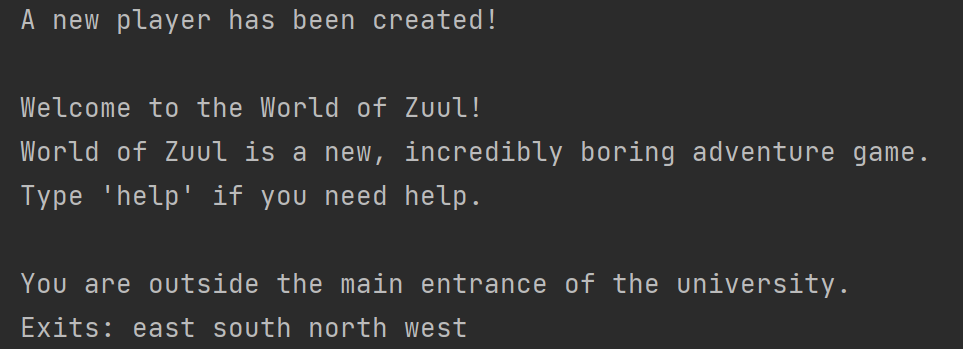
在移动到下一个房间时检查房间类型，如果是1说明进入的是传送房间，那么调用随机函数生成一个随机数。产生的随机数对应的房间就是传送去的房间，并且该房间不能与原房间相同。

**6.5运行程序及其结果**

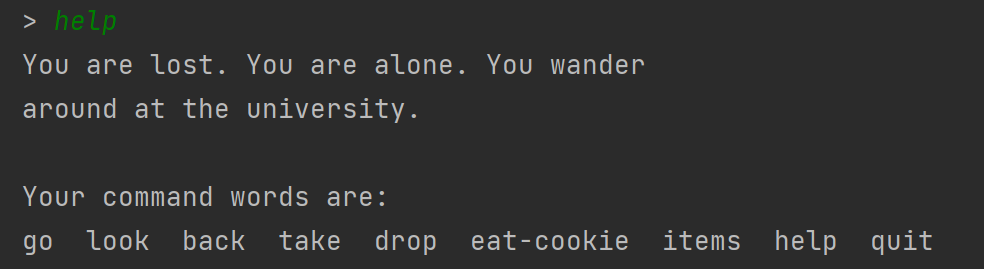
首先运行Main函数，系统出现用户登录界面。



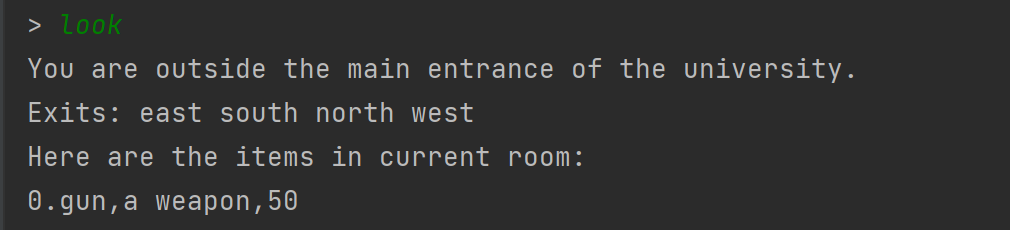
然后进入游戏有欢迎语以及给你提示关键词help介绍其作用。



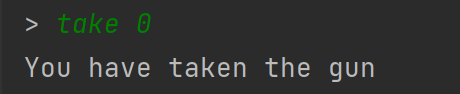
接下来我们输入help来看看有哪些功能。Help功能能介绍游戏简单的背景以及介绍可使用的功能。



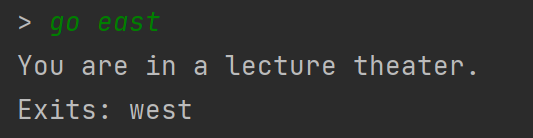
然后我们输入look功能来看看此房间有没有物品。



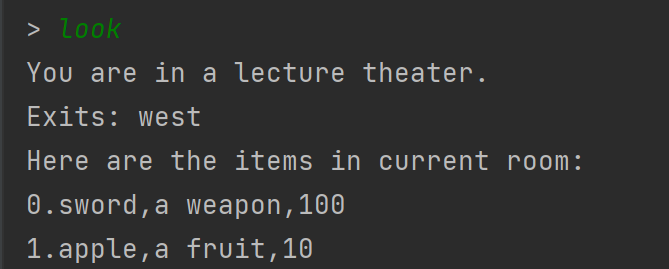
我们使用take功能将这个物品捡起。



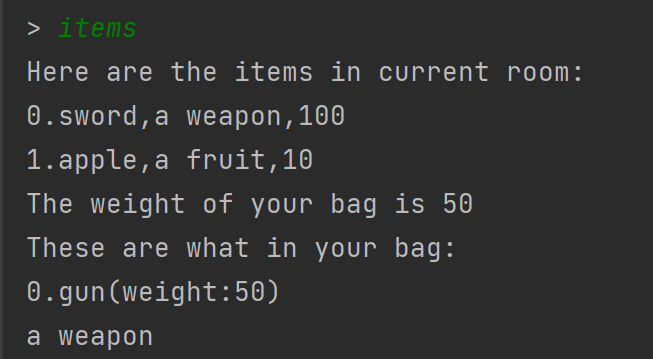
然后我们使用go功能进入下一个房间，我们输入go east进入lecture theater。



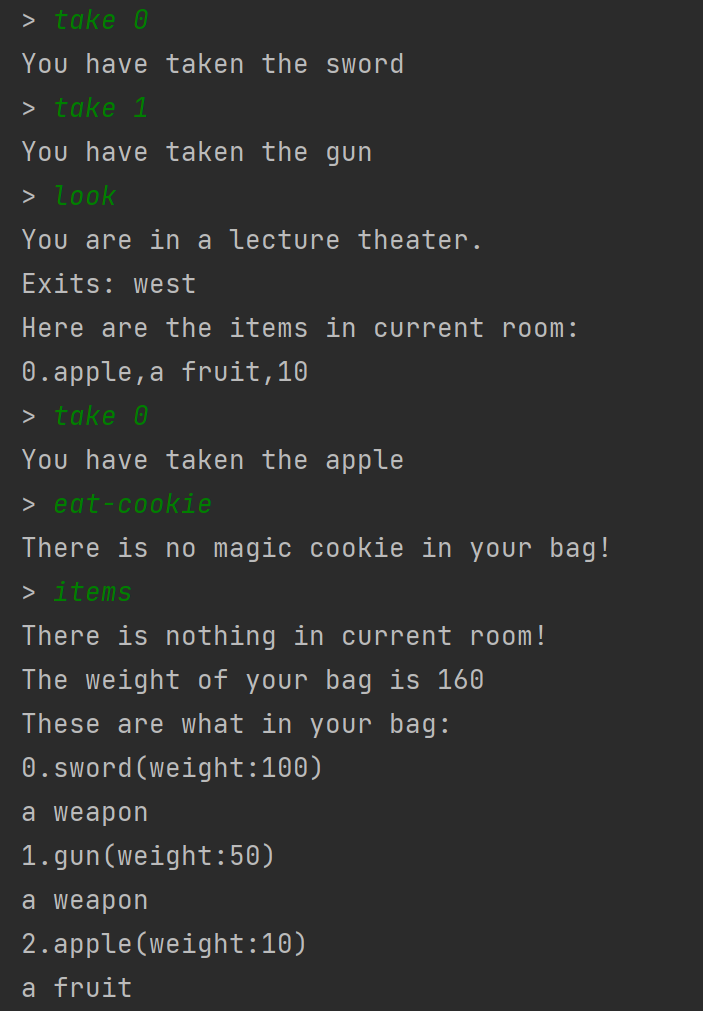
然后我们使用look功能再看这个房间中有什么散落的物品。



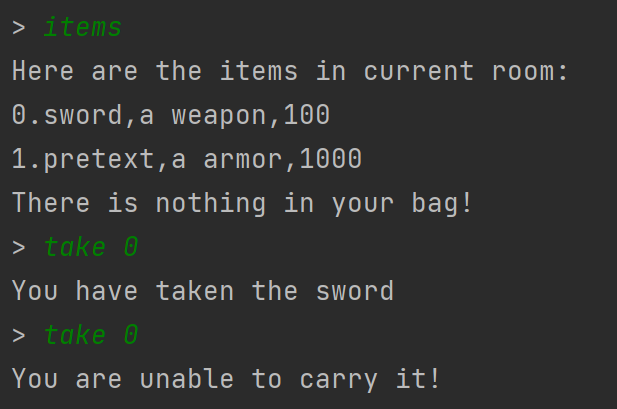
我们在调用items功能看看这个房间散落的物品以及玩家背包里的物品。



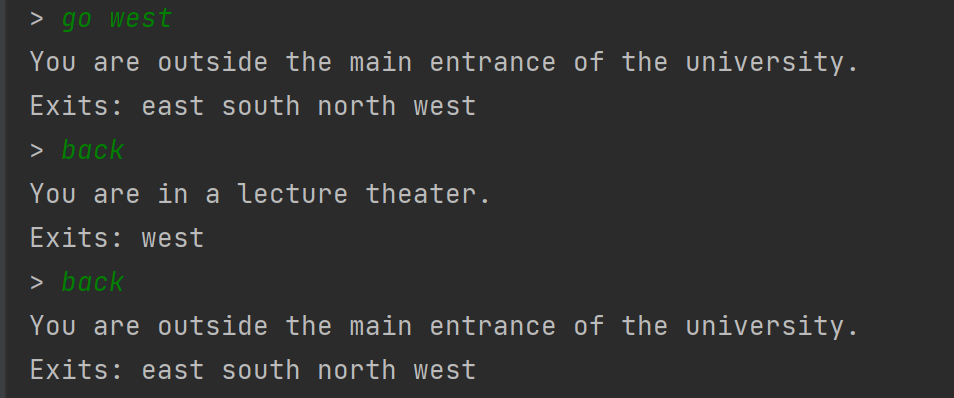
我们在使用take和drop功能对物品进行拾取和丢弃。



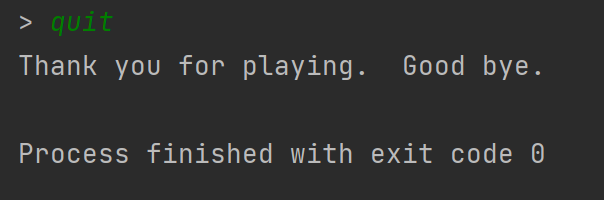
如果物品超重则提示无法拾取。



最后我们再使用高级的back功能，来回到之前的房间。



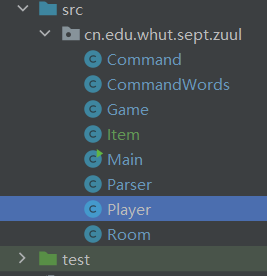
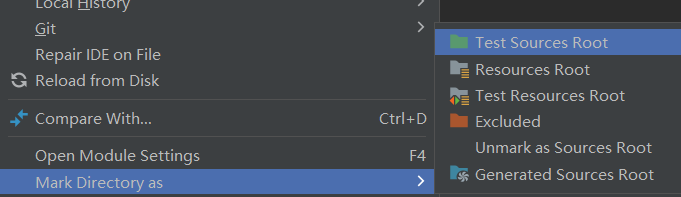
最后我们再使用quit功能来退出游戏。



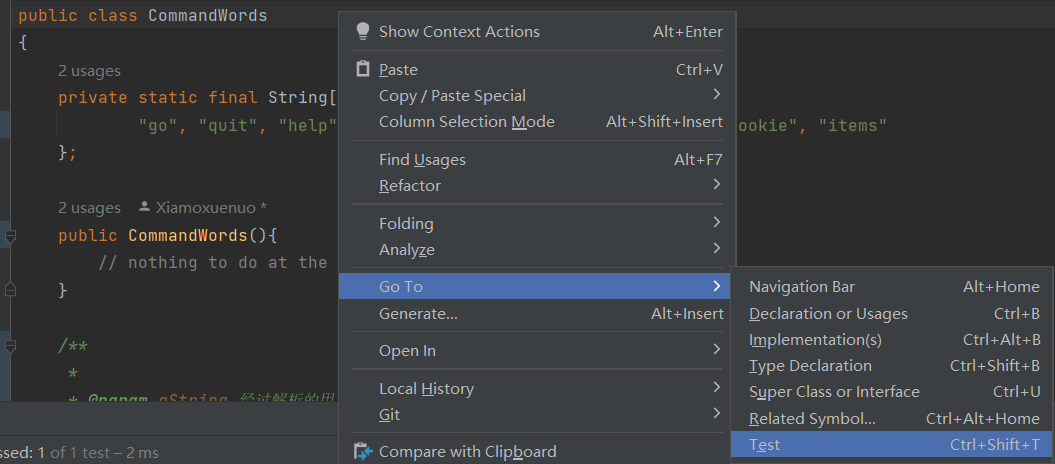
**6.6软件测试**

**6.6.1测试方法**

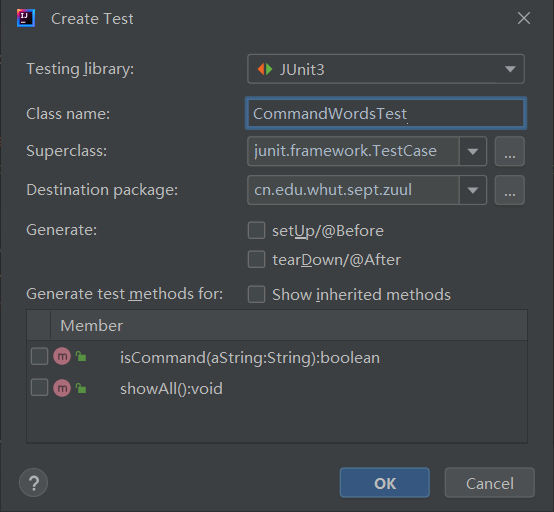
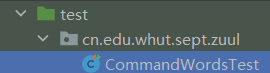
新建一个与src同级的文件夹保存测试代码，右键该文件夹，将其设置为测试文件根目录。

右键需要测试的类，选择Go To —> Test



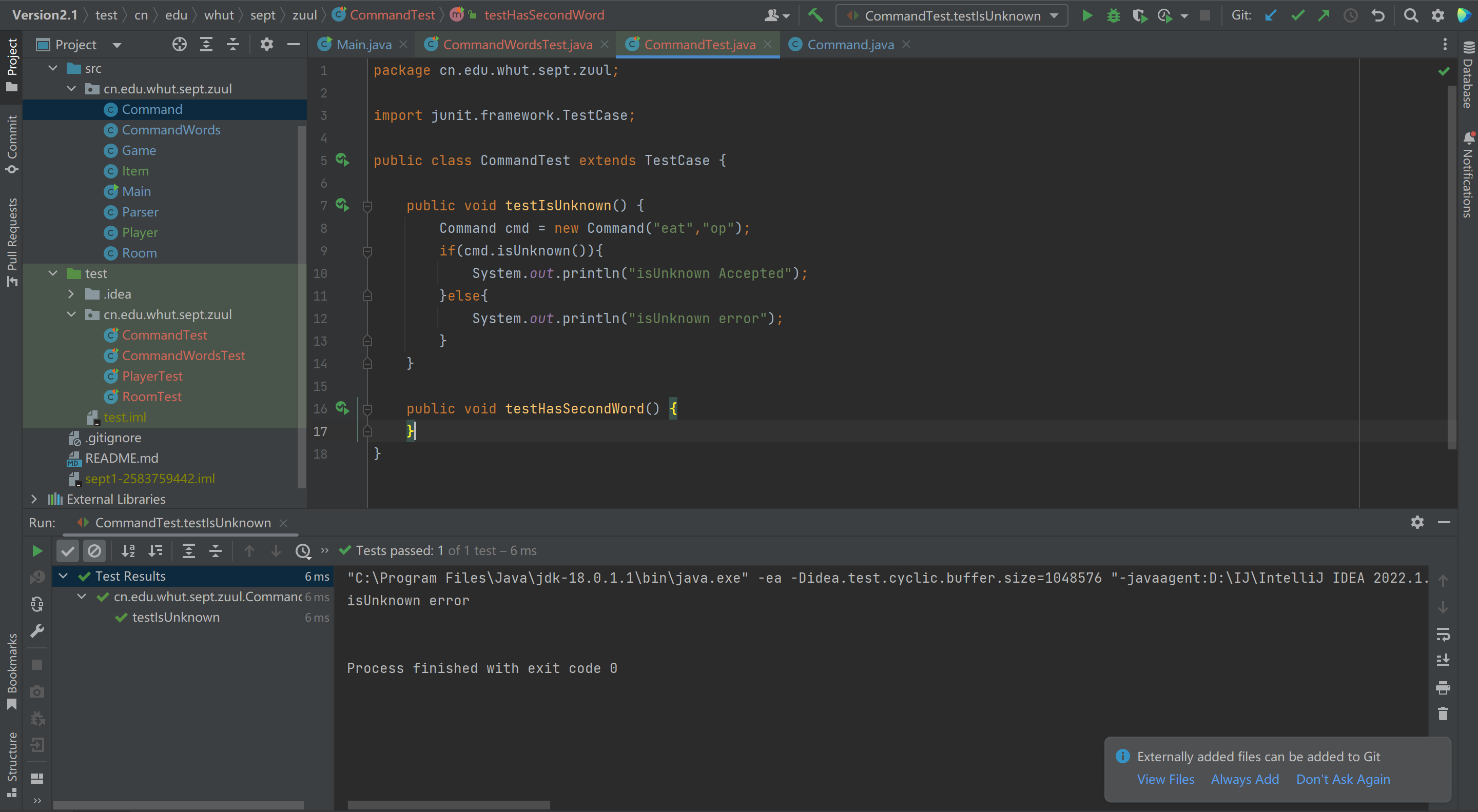
勾选需要测试的方法，得到测试类。

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**

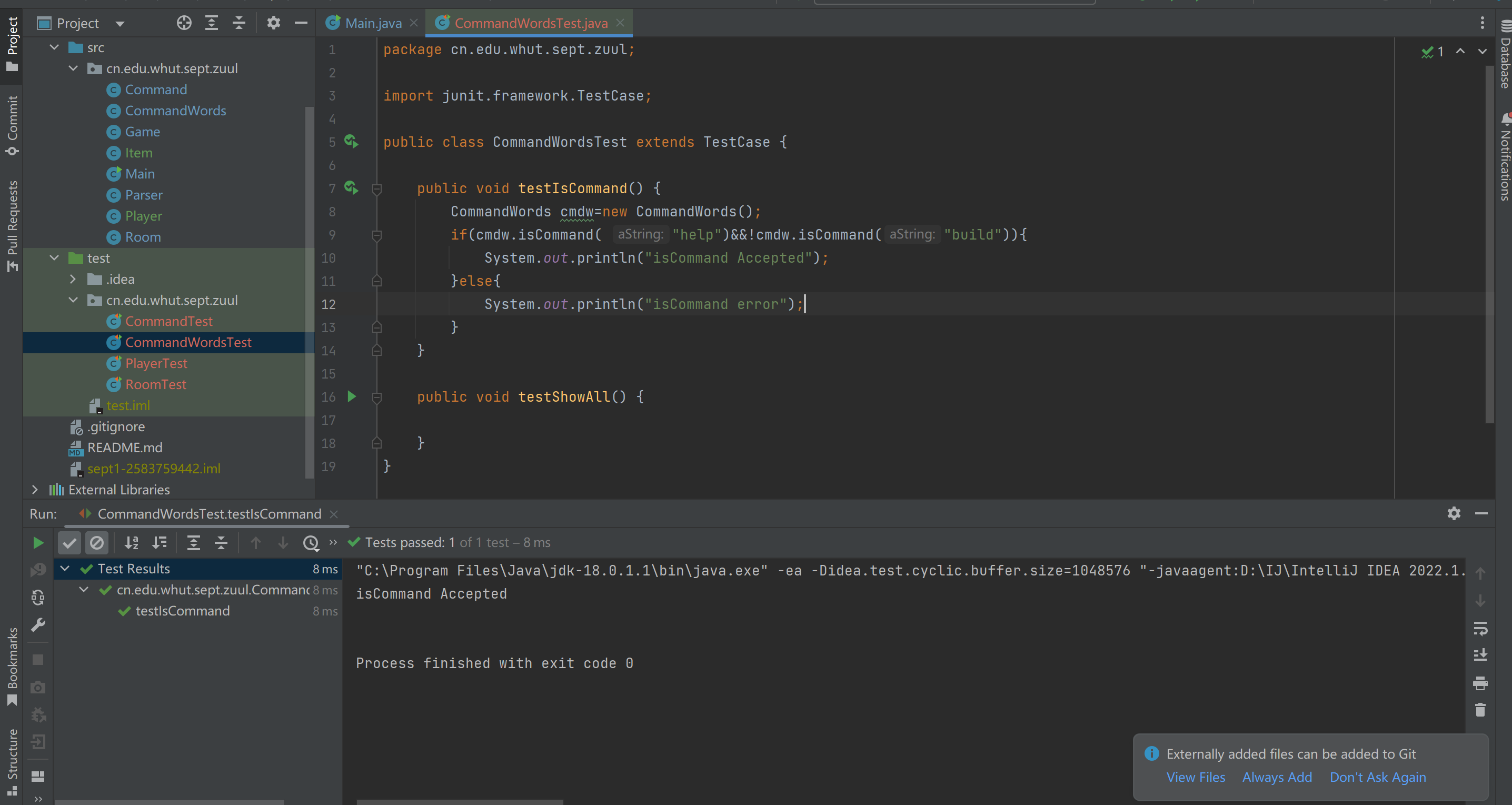
在CommandTest类中运行testIsUnknown()对Command类中的isUnknown()方法进行测试。



eat不是有效命令，故测试结果正确。

**6.6.3测试CommandWords类**

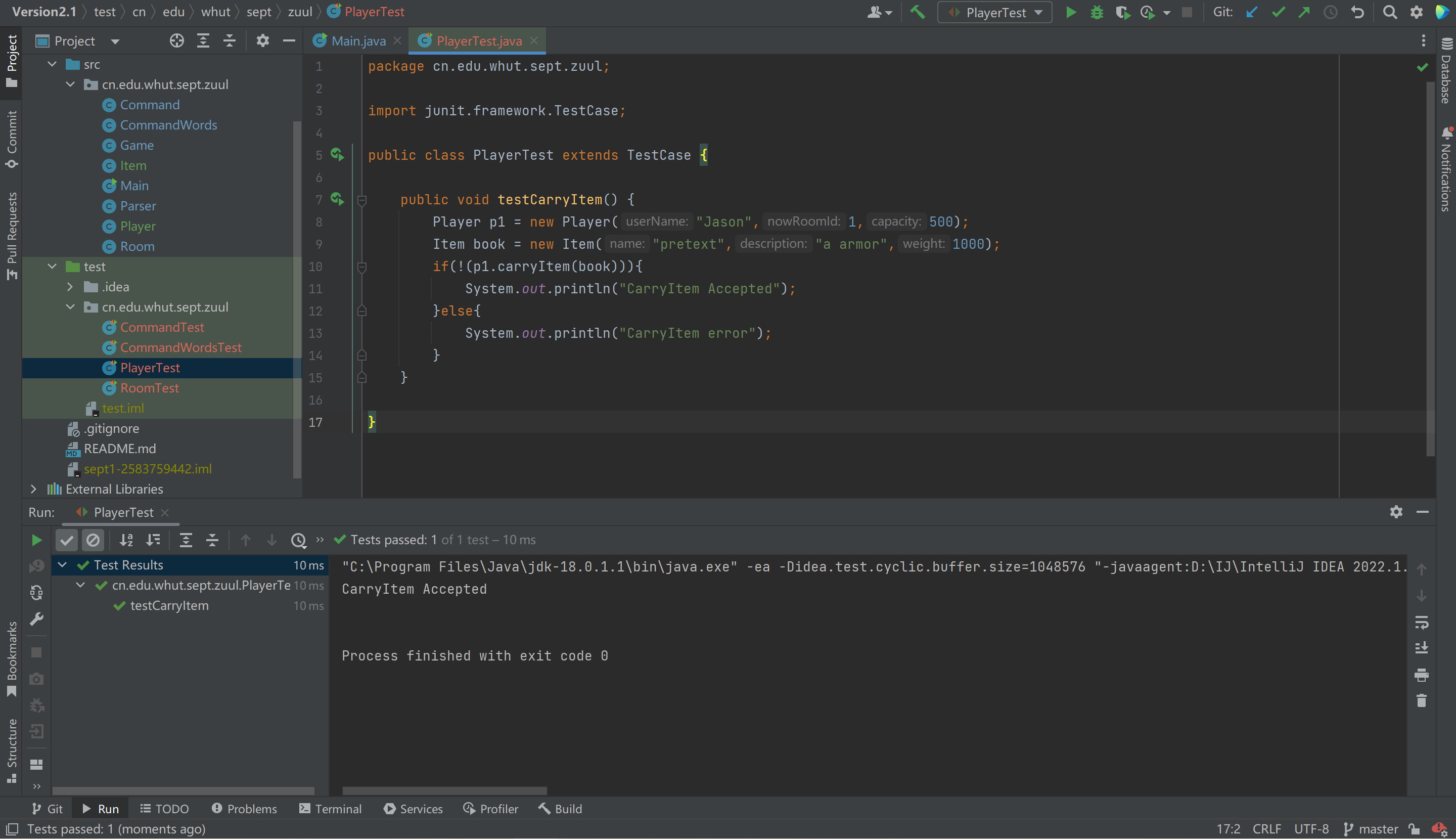
在CommandWordsTest类中运行testIsCommand()对CommandWords类中的isCommand()方法进行测试。



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**

在PlayerTest类中对CarryItem()方法进行测试。



Pretext的重量为1000，超过了Jason的承重能力500，故函数返回值应为false，取反则为true，测试结果正确。

粗略的对几个类中的部分方法进行测试，测试均通过。

# 7 实施过程问题记录与分析

本次实验主要集中在功能扩展点以及系统测试部分，还有一开始使用IntelliJ IDEA时，我的编译环境没有没配置好，主文件也选的不正确，在问了很多同学以及在网上长时间的查阅资料后，经过漫长的调配后，我终于配置好我的环境了。

在功能扩展点的编写上，我也遇到了很多困难，很多逻辑以及工程方面的问题，让我很是头疼，需要十分仔细的去分析，同时还要很熟悉前面自己或是框架已经完成的方法，这样可以减少很多冗余代码的出现。同时在编写完代码需要即使的写代码注释，这样方便之后寻找相应的功能。

在系统测试阶段，我因为使用Junit不是很频繁，所以在使用和编写测试用例的时候不是很熟练。设计测试用例有一定的困难。

另外还有一个点比较重要，就是我在使用IntelliJ IDEA进行push操作将工程上传至GitHub时，经常无法将IntelliJ IDEA与GitHub连接，导致无法直接上传，后来通过大量的查阅，发现可以直接在GitHub中Add file —> Upload file来直接上传文件，十分方便。

# 8 任务总结

通过本次实践任务，我理解、掌握和应用软件工程的思路、方法、技术和工具，在实践中培养了自己个人的软件工程能力。通过这种与小游戏相关的中小型任务来循序渐进的开展实践，以我自己为开发单位，采用迭代递进的实践过程逐步来完成整个实践工程。

我基本掌握了基于GitHub开源平台的个人代码版本维护方法和MarkDown文件编写方法，生成md文件。复习到了《软件工程基础》的知识，复习了很多git指令以及操作，我对使用GitHub进行迭代开发变得更加熟练了。通过基于javadoc规范对代码进行标注，我在网上搜索代码注释规范，最后学习了codestyle，这样的学习，让我编写代码时加深自己对代码的理解，同时更加规范。

通过设计功能扩展点，我的编程能力得到了很大提高。虽然在代码测试不是很熟练，我还是学习了Junit的用法，基本对每一个类都进行测试了。

这次以个人开发方式对样例代码进行阅读、分析、标注和扩展，再通过基于GitHub的分布式协同软件开发平台，最后基于互联网的开源软件和开发知识库，完成了这次实验。

这次实验我系统地了解了在编写一个项目时有哪些需要注意的地方，也使我意识到单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** | | **张宸** | **学 号** | **0122006290904** | |
| **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**