|  |  |
| --- | --- |
| **学 号：** | 0122001101314 |

****

**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** | **涂典** |
| **编 号** |  |
| **指导教师** | **唐祖锴** |

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

本次实践任务是以个人开发的方式对样例代码进行阅读、分析、标注和扩展，要求学生基于GitHub的分布式协同软件开发平台完成开发，同时参考互联网上的开源软件和开发知识库。

·实践任务

1. 阅读和理解样例代码

·fork样例工程，并clone到本地仓库；

·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；

·精读样例工程软件代码，描述代码结构及部件组成；

·以UML图描述样例工程的组成及结构图（类及类之间的关系）。

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（可参考Game类的标注样例）；

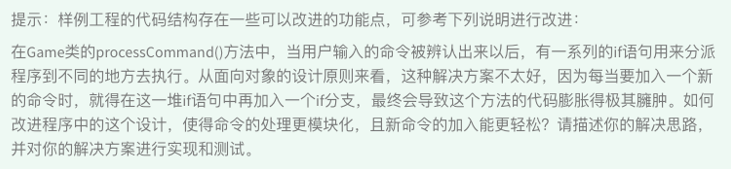
·注释后的代码提交到本地代码库后，同步推送到远程代码仓库；

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

3. 改进维护样例工程

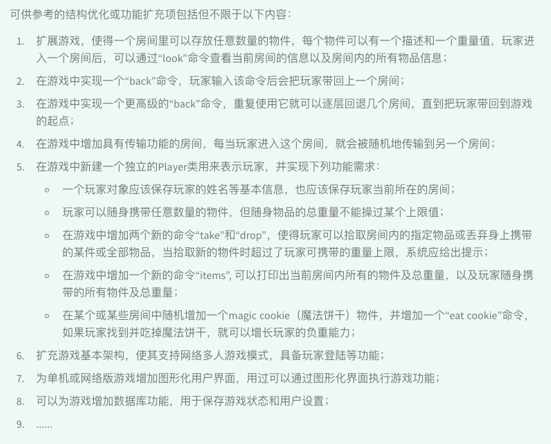
·对样例代码中的功能设计进行分析，找出若干设计缺陷和改进点，并进行修正或扩充，并集成到工程代码中；

·可借助代码质量分析工具或代码规范检查工具（如SonarQube、ESLint等)对代码质量进行分析，发现潜在问题。



4. 扩充样例功能

·样例工程“world-of-zuul”仅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项左右的功能扩展实现；



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。

任务输出相关要求：

·所有开发工作基于github平台完成，每位同学在自己项目根目录下创建一个名称为REPORT.md的文件，以markdown语法格式编写本实训过程完成的主要任务说明，主要包含样例工程的代码结构分析（可以用UML类图及文字进行说明），以及自己改进、扩展的功能实现说明，单元测试用例说明；

·所有源代码应包含规范化标注和扩充，并通过git提交到代码库；

·需另外提交word格式的课程实践报告。

·任务目的

·理解软件代码规范的重要性

·理解代码变化对软件质量带来的影响

·掌握基于Git的个人代码版本维护方法

·掌握MarkDown文件编写方法

# 2 任务分析

本次实践任务要求我们完成“巨洞探险”游戏完整的软件开发过程。经分析本次实践任务，可作如下归纳：

·工作重点

·clone样例工程并仔细阅读代码，描述代码结构及部件组成，并画出UML类图

·基于javadoc规范标注代码，并push到git仓库

·分析样例代码，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

·扩充样例功能

·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

·以markdown语法格式编写本实训过程完成的主要任务说明，主要包含样例工程的代码结构分析（可以用UML类图及文字进行说明），以及自己改进、扩展的功能实现说明，单元测试用例说明

·工具使用方案

本次实践任务主要采用的工具有：

·IntelliJ IDEA，进行主要的开发过程

·Enterprise Architect，进行UML建模

·

·重难点分析

·扩充样例功能十分考验编写代码的实践能力

·单元测试在之前做的并不多，不太熟悉

·需学习markdown语法格式来进行本实训过程完成的主要任务说明的编写

# 3 开发计划

·clone样例工程并仔细阅读代码，基于javadoc规范标注代码，并利用Enterprise Architect画出UML类图

·找出样例代码中的设计缺陷和改进点，并进行修正或扩充

·扩充样例功能

·每完成一项工作后都使用Github进行版本管理

·在完成整个开发过程后使用Junit进行代码测试

·创建markdown文件REPORT，利用markdown语法记录软件开发过程

# 4 软件配置计划

·版本的命名及提交规范

采用“vx”+“版本说明”的格式作为版本的命名及提交规范，例如，提交说明 “v1 增加某某功能”。

·源代码文件命名规范

采用驼峰命名法进行文件的命名，尽量做到一目了然。

·分支管理规范

整个开发过程始终合并在master分支下。

# 5 测试计划

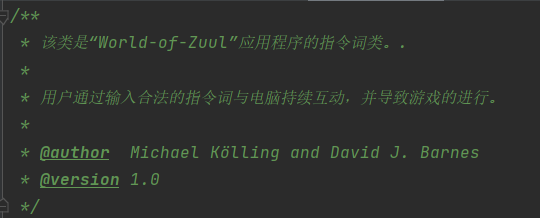
通过上网查阅相关资料[2]，本次实践任务我采取Junit对代码进行单元测试，主要的测试对象为Operation类和Room类，针对这两大类中的实现的方法。

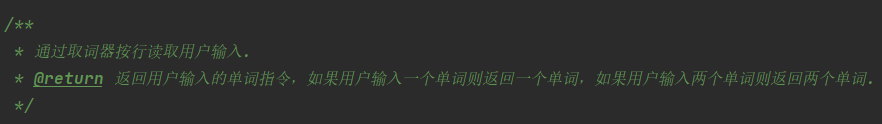
采取的测试手段是使用IDEA快速创建单元测试，提高开发和测试效率。

# 6 实施情况

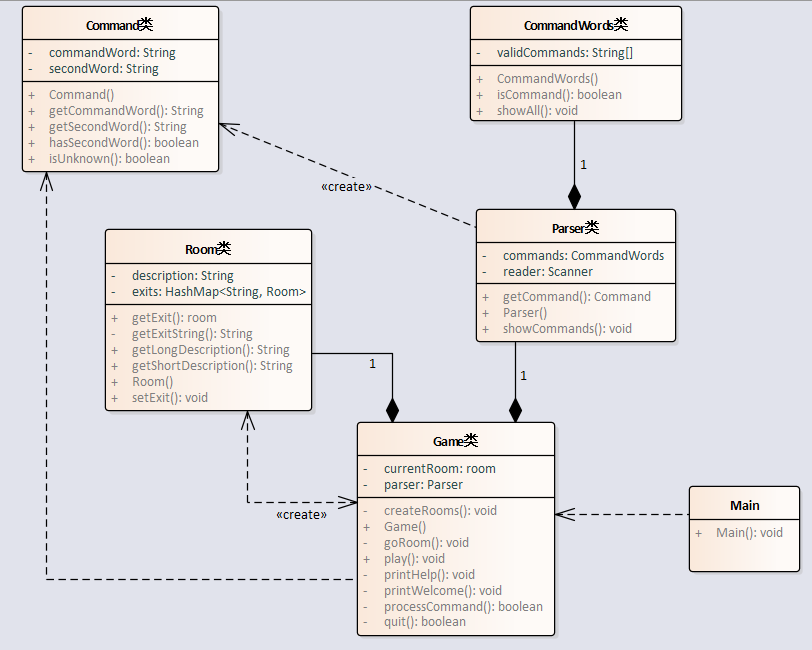
·通过上网查阅相关资料[3]，了解javadoc注释规范，我先对样例工程中的原始代码进行了相应注释，并提交到github。部分注释截图如下所示：

·parser类





·精读样例工程软件代码后，我理解了样例工程的代码逻辑，并使用Enterprise Architect画出UML了类图：



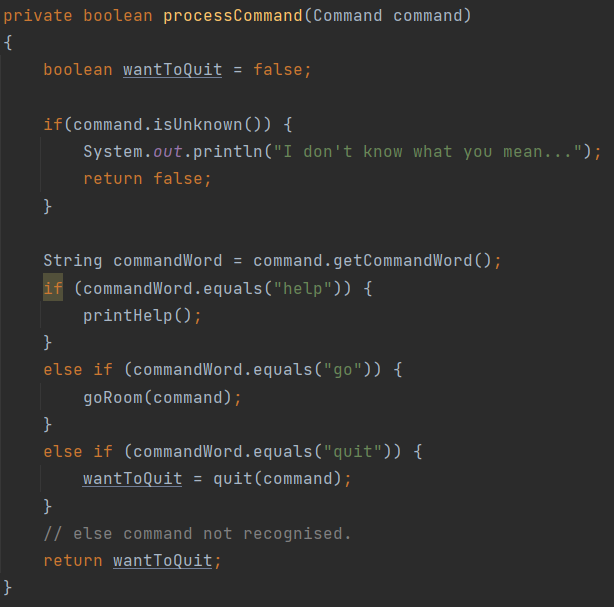
·找出样例代码中的设计缺陷和改进点，并进行修正或扩充

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。

通过上网查阅相关资料[1]，我找到了优化if else的设计模式的方案。通过阅读博客[1]中的相关内容，最终我采取了一种之前没有听说过的方案“策略模式+工厂方法消除if else”。

在实际操作的过程中，发现由于每个操作实现起来的区别，不太方便结合“工厂”方法，于是乎改用抽象类代替“工厂”功能。

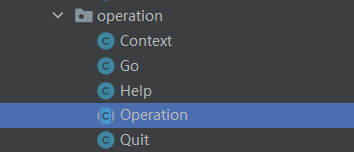
已知需求为，根据不同的指令类型，进行相对应的操作，优化前有以下代码：

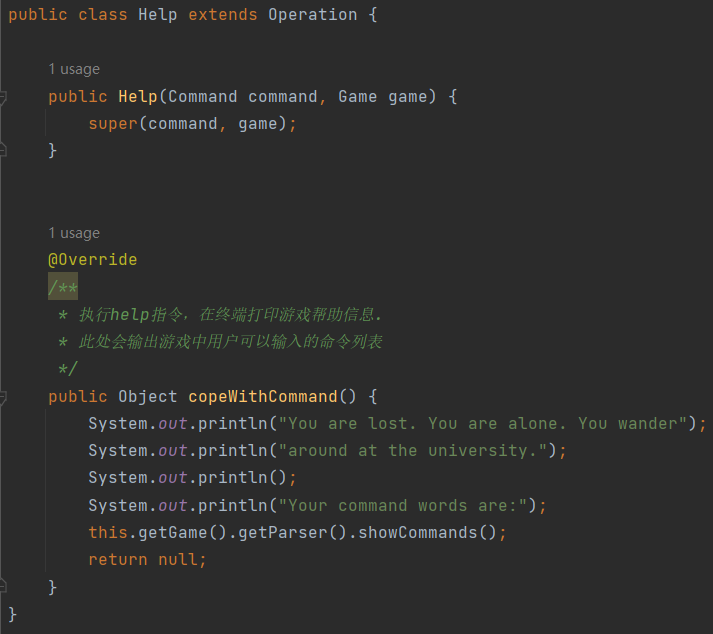


首先，我把每个条件逻辑代码块，抽象成一个公共的抽象类Operation，可以得出以下代码：

接着我根据每个逻辑条件，定义相对应的操作策略实现类“go”，“help”，“quit”，可得以下代码：





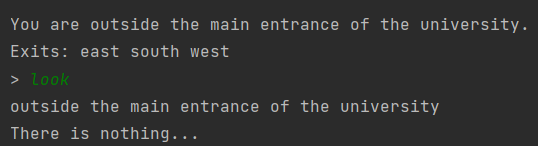
最终，在每个Operation的子类中的copeWithCommand()方法一一对应原本在Game类中实现的方法。由此可见，这样就避免了每当要加入一个新的命令时，就得在这一堆if语句中再加入一个if分支而导致的代码膨胀臃肿问题。这种代码实现方法也更符合面向对象设计原则，使得命令的处理更模块化，且新命令的加入能更轻松。

·扩充样例功能

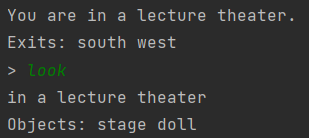
·增加look功能

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，如下所示：

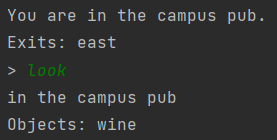
·outside the main entrance of the university



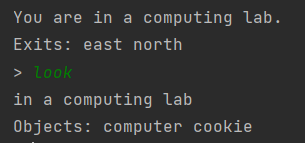
·lecture theater



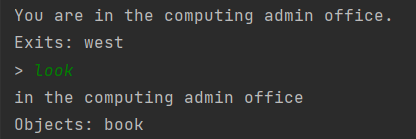
·campus pub



·computing lab

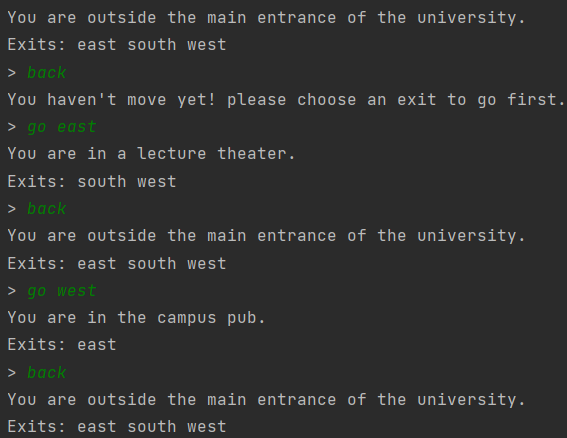


·computing admin office



·增加 back 功能

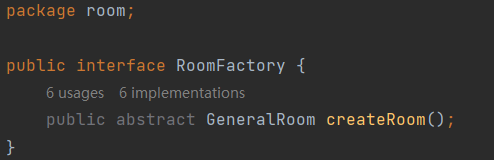
在游戏中实现一个“back”命令，玩家输入该命令后会把玩家带回上一个房间，具体操作如下所示：



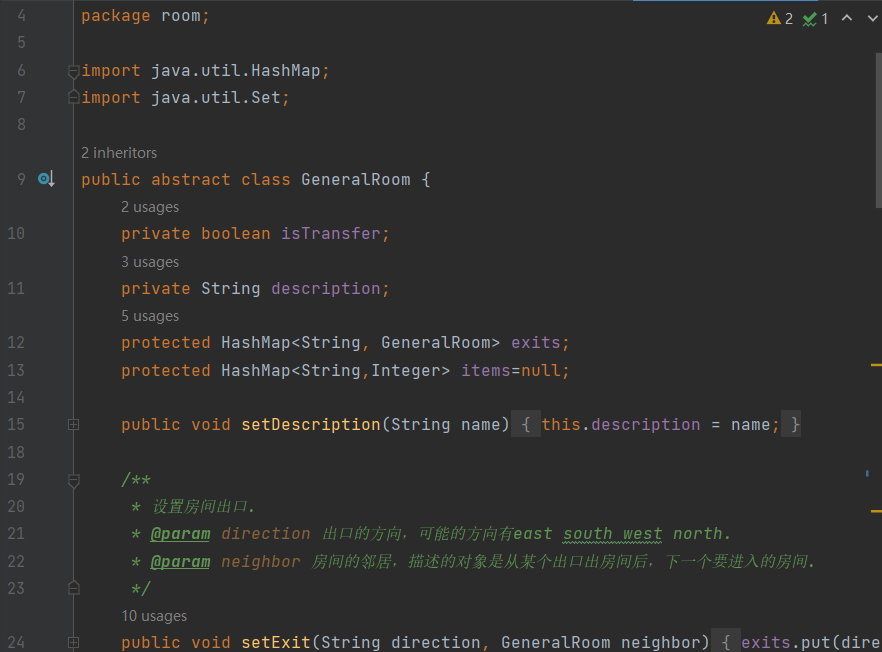
·新增room module，实现模块化设计。将Room类修改为抽象的工厂类，用来管理这些Room类的实现，并在各个房间中添加一些物品。再增加一个随机房间。

通过将Room类抽象化为GeneralRoom，同时抽象出一个公共的接口RoomFactory用于生成新的房间，这样设计很好的避免的代码冗余问题，符合“高内聚，低耦合”设计原则。

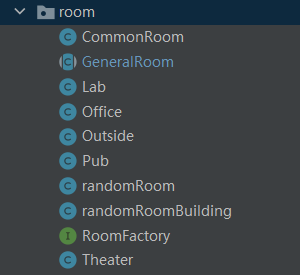
·RoomFactory



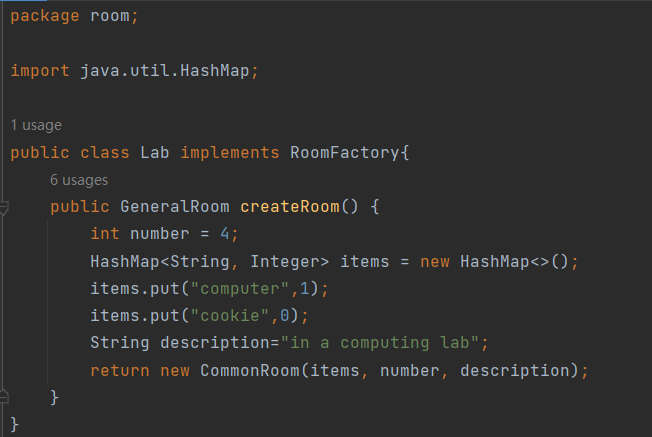
·GeneralRoom



经过上述重构，我把每个房间都单独设为一个类：



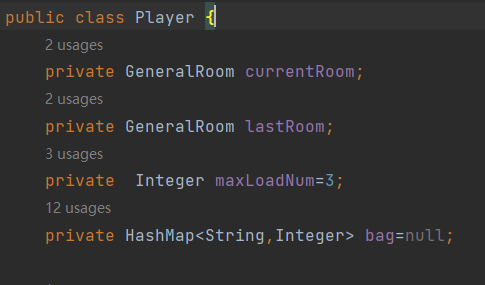
例如，Lab房间类如下所示：



随后将其余各个房间的基本信息也都定义在其自身的类中，这样设计的好处是，当需要修改某个房间的基本信息时，只需要对该房间对应的类进行修改即可，减少了修改代码时出错的可能性。

·增加Player module

在游戏中新建一个独立的Player类用来表示玩家，该类具有的属性如下：



currentRoom表示玩家当前所在房间；

lastRoom表示玩家进入当前所在房间之前的所在的上一个房间，用于实现“back”操作；

bag表示玩家的背包，背包中的物品具有物品名、重量两个属性；

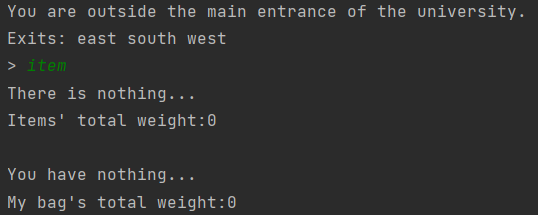
maxLoadNum表示最大负重量，玩家拾起房间内物品的重量不能超过该数值，同时玩家可以通过吃饼干操作来增加最大负重量。

·增加item功能

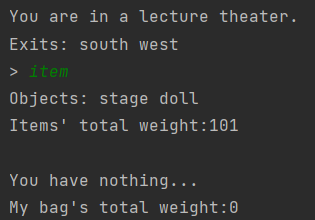
在游戏中增加一个新的命令"items",可以打印出当前房间内所有的物件及总重量，以及玩家随身携带的所有物件及总重量。

玩家进入各个房间时的打印信息如下：

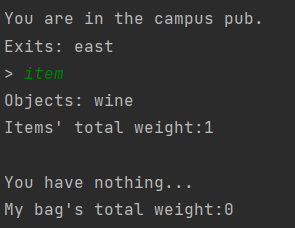
·outside the main entrance of the university



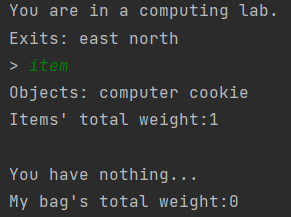
·lecture theater



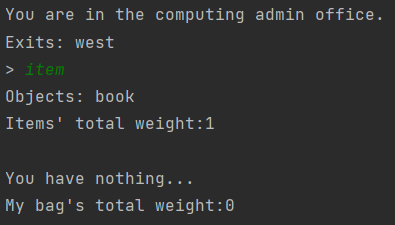
·campus pub



·computing lab



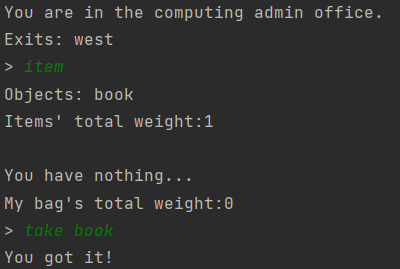
·computing admin office



·增加take功能和drop功能

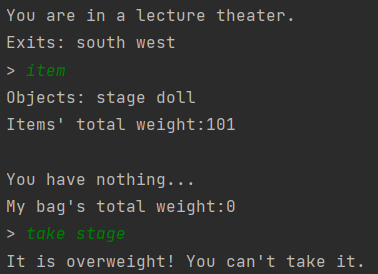
在游戏中增加两个新的命令“take"和"“drop”，使得玩家可以拾取房间内的指定物品或丢弃身上携带的某件或全部物品，当拾取新的物件时超过了玩家可携带的重量上限，系统应给出提示。

例如，进入computing admin office并拾起book时：



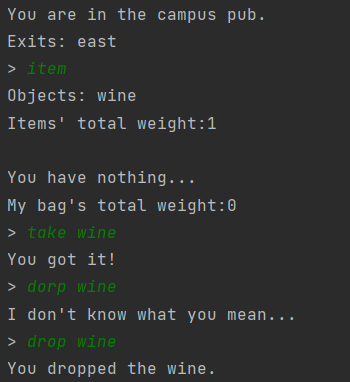
成功取走物品。

进入lecture theater并拾起stage时：



取走物品失败，系统提示超重。

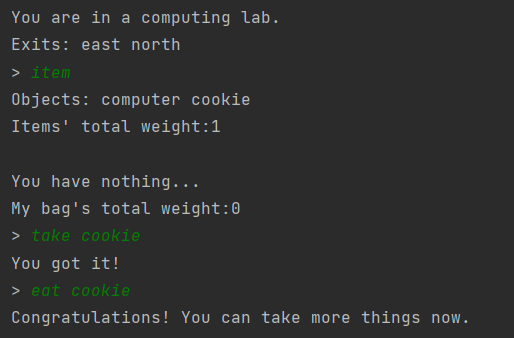
进入pub拾起wine后，将其丢弃：



·增加eat cookie功能

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。

例如，computing lab中有一块饼干，拾起并吃掉后可以增加玩家的负重能力：

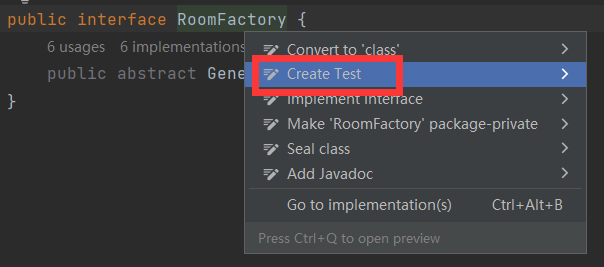


·单元测试

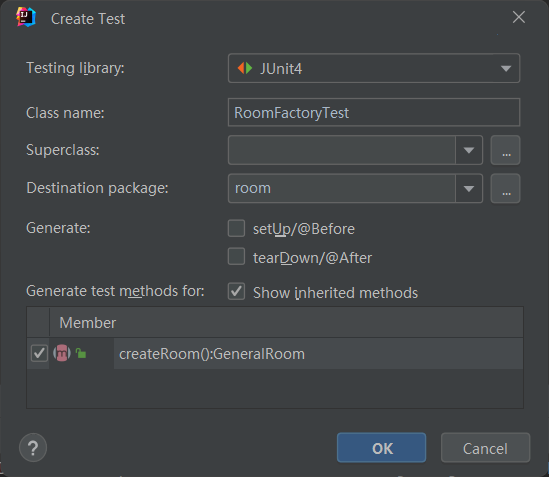
针对功能改进和扩充，在项目结构中编写单元测试用例，对代码执行单元测试。

·对RoomFactory接口进行测试：

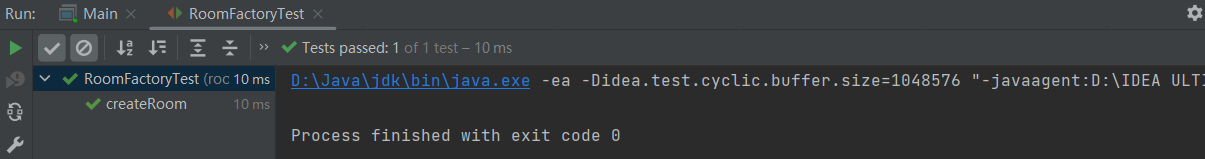
先生成测试代码：



勾选测试方法：



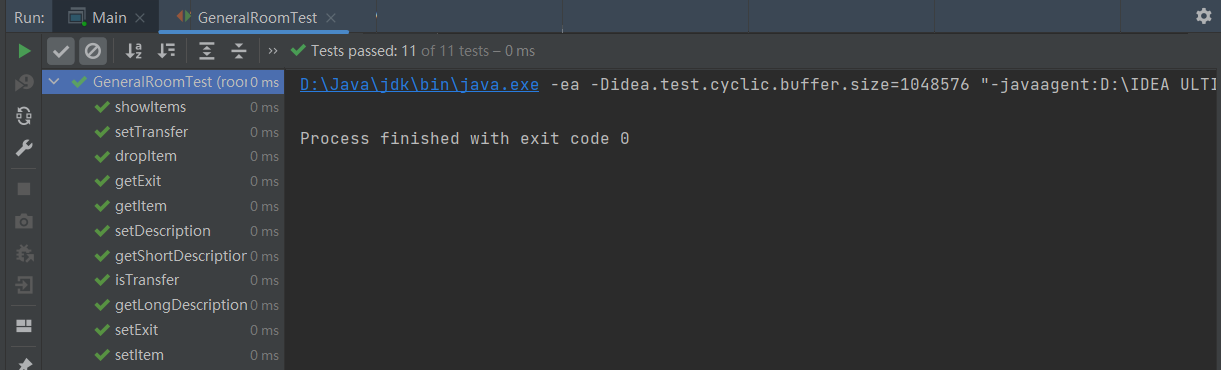
运行单元测试：



测试通过。

·采取上述相同方式对对GeneralRoom类进行测试：

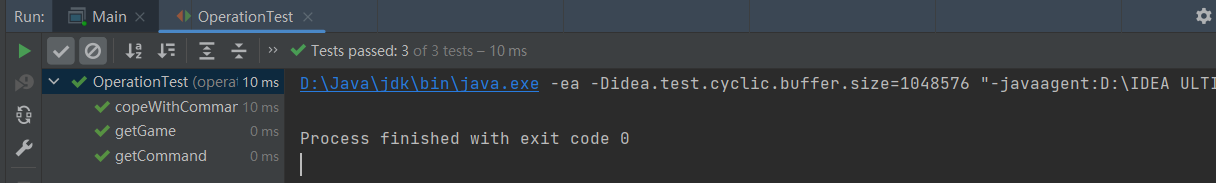
运行单元测试：



测试通过。

·采取上述相同方式对对operation类进行测试：

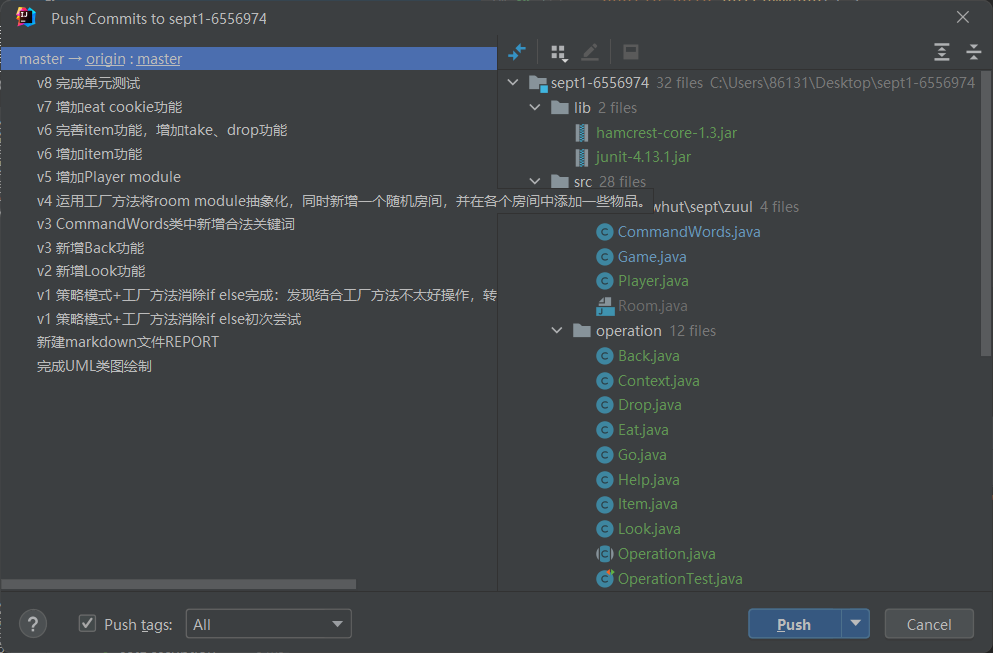
运行单元测试：



测试通过。

·代码提交情况

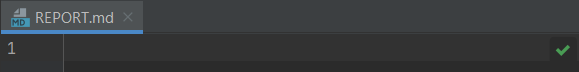
由于网络原因，与远程仓库的连接时常不稳定。故我在每完成一部分内容后，都先将相关修改后的代码提交到本地仓库：



最后再统一push到git远程仓库。

·以markdown语法格式编写本实训过程完成的主要任务说明

新建markdown文件：



参考网上相关文档[4]以及README.md文件的书写内容进行编写。

# 7 实施过程问题记录与分析

相关问题记录与分析：

·由于网络环境问题，进行版本更新时，经常导致push到远程仓库这一操作超时而无法完成。

·对于javadoc的书写没有任何经验，花费了一定时间进行学习。

·重构room module的过程显得较为艰难，但通过网上学习以及同学之间的讨论，最终实现了这一步骤。

·对于Markdown的基本语法不甚了解，参考了网上相关文档[4]。

# 8 任务总结

本次软件工程实训任务以java这门面向对象程序语言为基础，综合考察了我们在编程实践过程中的绝大部分要素。从最基本的编程能力，到规范化的代码规范与注释，再到项目开发中涉及到的版本管理等等。总得来说，这次实训与我而言，对我个人作为一名未来的“程序员”的综合素质提升是巨大的。

首先，在本次实训开始前，我对于github的使用并不熟练。我对github的了解仅仅停留在软件工程基础这门课的课堂教学中，虽然后面参加学校里老师的开发项目组完成一些工作时也有用到github进行版本管理，但仍旧只是一知半解。本次实训任务算是提供了一个非常好的机会给我去完完全全的了解并掌握github版本管理技术。

其次，我学习了JavaDoc文档的书写。在日后的软件开发生涯中，书写JavaDoc文档是非常基本、重要的一项技能，可以说是每个程序员必备。本次实训中，我通过完成一整个项目工程的JavaDoc文档的书写，可以说是对这项技能有了一个初步的掌握，相信对日后会有很大帮助。

再者，我学习到了软件测试相关的技术。之前对于代码的编写，基本都是停留在运行、查bug、完成这一系列步骤中，可以说是很少用到软件测试技术。但实际上，软件测试对于软件工程项目，尤其是大型软件工程而言是必不可少的。本次学习经验也加深了我对软件测试的了解。

接着，我通过完成REPORT.md文档的书写，简单了解了markdown语法。markdown让整个开发过程更有条理，同时也构建起了开发者与交付方之间沟通的桥梁。

最后，必不可少的一项当然就是提高了我编写代码的能力，整个实训任务走下来，从阅读理解样例工程到改进、扩展功能，我查阅了网上不少资料，也和几位同学进行过交流沟通，最终较为顺利的完成了整个任务，这不仅是对我编写能力的一大考验，也考查了我的创新、灵活变通能力。

总而言之，本次实训令我受益匪浅。

# 9 参考文献

[1] <https://blog.csdn.net/weixin_36755535/article/details/123377660>

[2] <https://blog.csdn.net/weixin_38556197/article/details/123835435>

[3] <https://blog.csdn.net/GarfieldEr007/article/details/54959597/>

[4] <https://www.jianshu.com/p/191d1e21f7ed>

[5] <https://m.tw.cljtscd.com/baike-%E5%B7%A8%E6%B4%9E%E5%86%92%E9%9A%AA>

**《软件工程实践（一）》成绩评定表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓 名** | | **涂典** | **学 号** | **0122001101314** | |
| **专业、班级** | | **软件工程 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**