|  |  |
| --- | --- |
| **学 号：** | 0122010880228 |

****

**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** | **雷霆杰** |
| **编 号** | **0122010880228** |
| **指导教师** | **唐祖锴** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2022 | 年 | 12 | 月 | 29 | 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

**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

**任务内容：**

**阅读和描述样例工程：**

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；

以UML图描述样例工程的组成及结构图（类及类之间的关系）

可结合markdown语法和mermaid插件绘制所需图形

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

**扩充和维护样例工程：**

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对代码质量进行分析，发现潜在问题；

**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

本次软件工程实践主要内容为开发基于字符界面的探险游戏world-of-zuul，项目的开发对学生的软件代码编写能力、软件框架设计能力、软件管理、代码规范、运行维护都做出了相应的要求，考察的不仅是编程者的代码编写能力，更是一次对编程者项目管理综合编程能力的全面考察。

本人通过敏捷编程的方法，使用intellij idea开发工具采用面向对象的设计方法进行设计，首先对题目中所给的样例程序进行UML架构分析，并在分析好的基础上，按照软件工程化的要求对该项目进行进一步的维护、功能的拓展和测试，

# 3 开发计划

**项目开发计划（说明各阶段计划采用的主要软件工程工具）：**

**一、软件项目设计：**

基于程序设计方法，完成开发案例的设计、功能点扩展和功能实现

本项目使用intellij idea开发工具采用面向对象的设计方法进行设计，在基础功能的充分理解上，改进维护基础功能以及优化项目结构，并对系统功能进行进一步的拓展优化。

**基础功能：**

1. 玩家可以通过go指令前往与该房间相连的房间；
2. 玩家可以通过help指令获取当前可执行的命令；
3. 玩家可以通过quit指令退出游戏；

**拓展功能：**

1. 新增look指令，玩家可以通过“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. 新增 “back”命令，玩家输入该命令后会把玩家带回上一个房间；

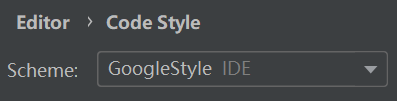
**二、代码版本管理：**

通过代码每次不断的更新，以及扩充代码功能。本项目通过中小型任务循序渐进开展实践，基于代码版本管理的概念与方法，对开发过程中代码仓库、工作空间、版本基线、代码提交与合并过程进行控制，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，项目及其开发分支存储在Github中。

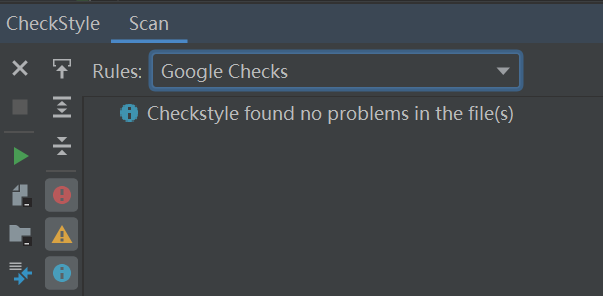
**三、软件编码规范：**

遵循开发语言编码规范，保持软件代码的可读性与可维护性

本项目的代码规范为google-style利用CodeStyle和checkStyle工具对项目代码和注释的编写进行规范性约束和检查



使用谷歌模式对项目进行checkCode检查，检查通过。



**四、软件单元测试：**

按照单元测试理论，在项目开发过程中通过测试用例实施单元测试

基于Junit框架通过编写测试类的方法实施单元测试和测试驱动开发

# 4 软件配置计划

**源代码文件命名规范：**

命名规范参照**阿里巴巴Java开发手册**，做出规范：

1. 【强制】代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

2. 【强制】所有编程相关的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。

3. 【强制】类名使用 UpperCamelCase 风格，但以下情形例外：DO / BO / DTO / VO / AO / PO / UID 等。

4. 【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格。

5. 【强制】常量命名全部大写，单词间用下划线隔开。

**版本命名规范：**

软件版本号由四部分组成，第一部分为主版本号，第二部分为次版本号，第三部分为修订版本号，第四部分为日期版本号加希腊字母版本号，希腊字母版本号共有五种，分别为base、alpha、beta 、RC 、 release，

* Base：通常包括所有的功能和页面布局，但是页面中的功能都没有做完整的实现，只是做为整体网站的一个基础架构。
* Alpha ：软件的初级版本，以实现软件功能为主，存在较多Bug，需要继续修改，是测试版本。测试人员提交Bug经开发人员修改确认之后，发布到测试网址让测试人员测试，此时可将软件版本标注为alpha版。
* Beta ：相对于Alpha 版消除了严重错误，但还需要经过多次测试来进一步消除，此版本主要的修改对象是软件的UI。修改的的Bug 经测试人员测试确认后可发布到外网上，此时可将软件版本标注为 beta版。
* RC ：该版本已经相当成熟了，基本上不存在导致错误的Bug，与即将发行的正式版本相差无几。
* Release：最终版本，在前面版本的一系列测试版之后，终归会有一个正式的版本，是最终交付用户使用的一个版本。该版本有时也称标准版。

**分支管理规范：**

本系统分支规范包括 main、develop、release、hotfix、feature分支。版本库的两条主要分支：Master和Develop。前者用于正式发布，后者用于日常开发。除了常设分支以外，还有临时性分支，用于应对一些特定目的的版本开发。临时性分支主要有三种：功能（feature）分支、预发布（release）分支、漏洞修补（hotfix）分支。这三种分支都属于临时性需要，使用完以后，应该删除，使得代码库的常设分支始终只有Master和Develop。

**各分支使用办法说明如下：**

1. ***主分支main：***

main分支上存放的应该是随时可供在生产环境中部署的代码（Production Ready state）。当开发活动告一段落，产生了一份新的可供部署的代码时，main分支上的代码会被更新。同时，每一次更新，都有对应的版本号标签（TAG）。

1. ***开发分支develop：***

develop分支是保存当前最新开发成果的分支。通常这个分支上的代码也是日常开发发布的代码（Nightly build）。因此这个分支有时也可以被称作“integration branch”。

当develop分支上的代码已实现了软件需求说明书中所有的功能，通过了所有的测试后，并且代码已经足够稳定时，就可以将所有的开发成果合并回main分支了。对于main分支上的新提交的代码建议都打上一个新的版本号标签（TAG），供后续代码跟踪使用。

1. ***功能分支featrue：***

为了开发某种特定功能，从Develop分支上面分出来的。开发完成后，要再并入Develop。

功能分支的名字，可以采用feature-\*的形式命名。

1. ***预发布分支release：***

发布正式版本之前（即合并到main分支之前），我们可能需要有一个预发布的版本进行测试。预发布分支是从Develop分支上面分出来的，预发布结束以后，必须合并进Develop和main分支。

它的命名，可以采用release-\*的形式。

1. ***漏洞修补分支hotfix：***

软件正式发布以后，难免会出现bug。这时就需要创建一个分支，进行bug修补。漏洞修补分支是从main分支上面分出来的。修补结束以后，再合并进main和Develop分支。

它的命名，可以采用hotfix-\*的形式。

**提交规范：**

每次提交，Commit message 都包括三个部分：Header，Body 和 Footer。

其中，Header 是必需的，Body 和 Footer 可以省略。

<type>(<scope>): <subject>

// 空一行

<body>

// 空一行

<footer>

# 5 测试计划

***测试目的：***

在软件开发的过程中，对软件产品进行质量控制。

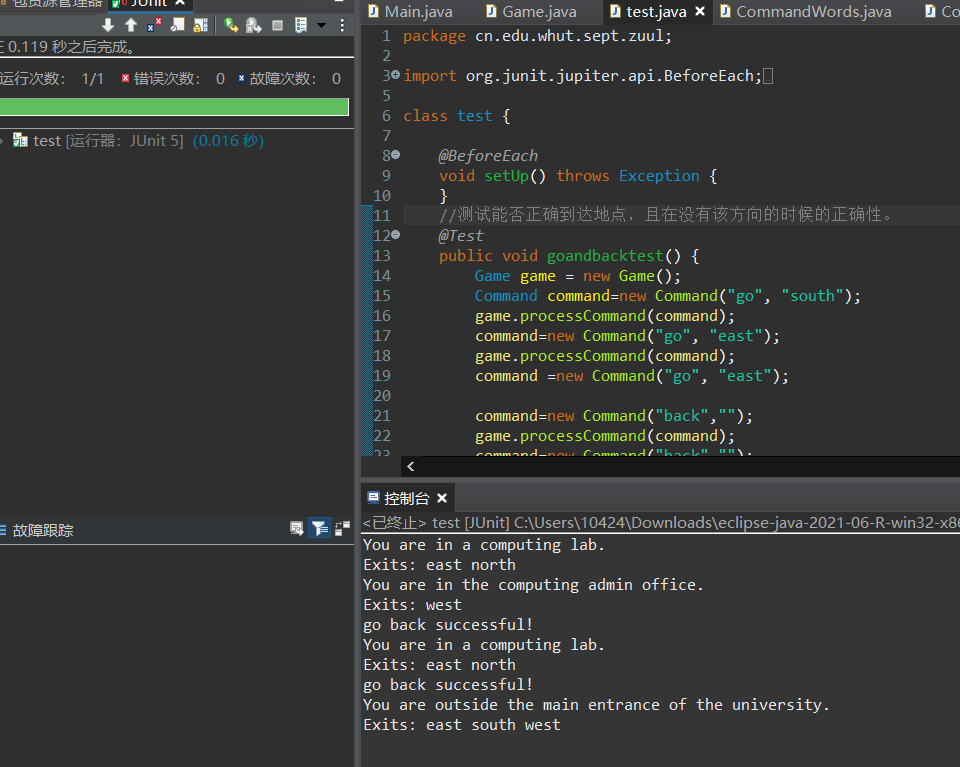
保证软件产品的最终质量

***测试范围***

新增功能LOOK和BACK功能

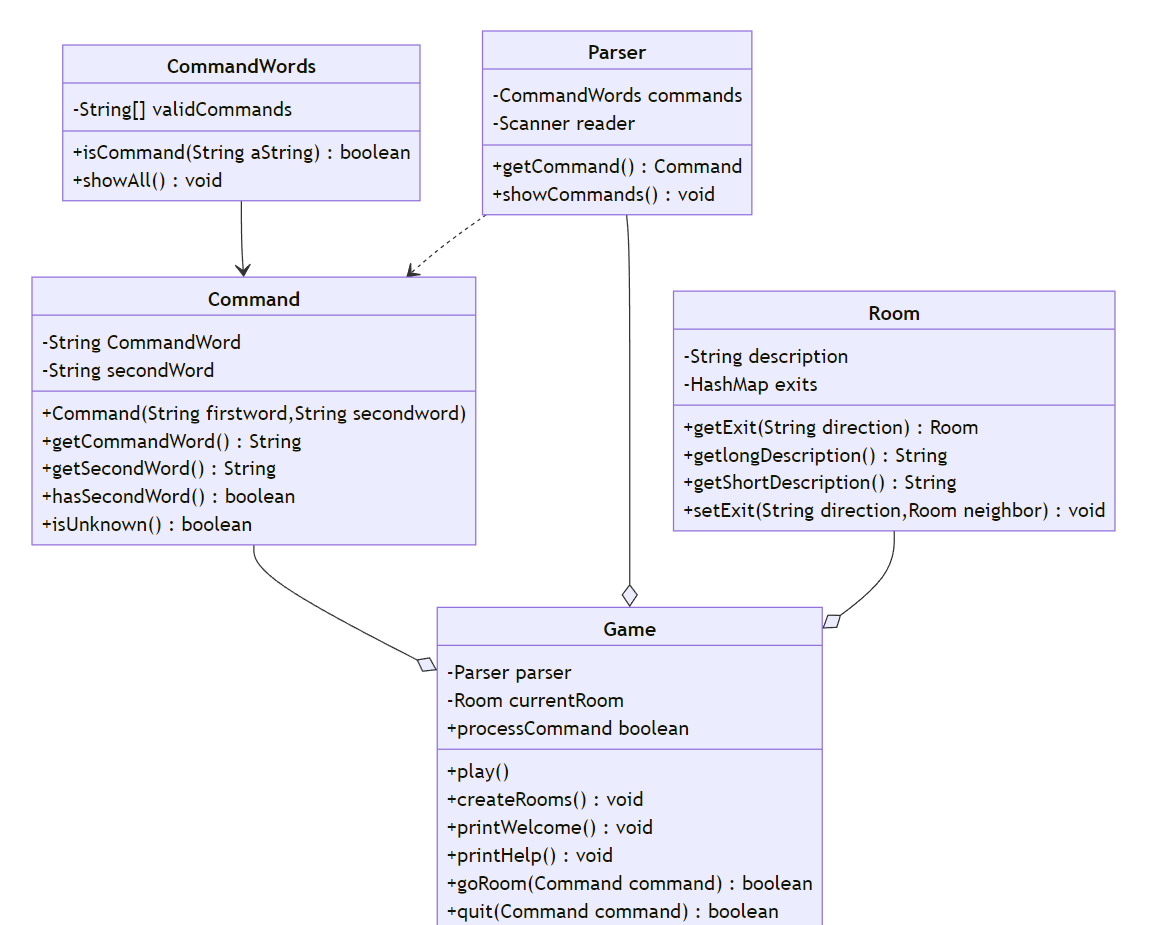
***测试方法：***

采用Junit单元测试（具体测试内容见实施情况）

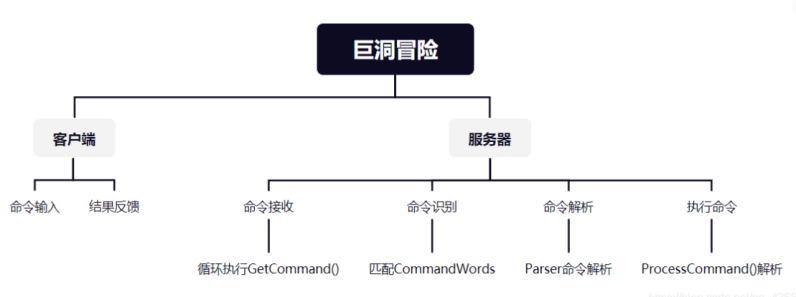


# 6 实施情况

类图



当前项目主要的实现目标就是玩家能够输入 Help、Go、Quit、back、look指令来操纵游戏，因此从用户角度来考虑系统用例就是玩家进入游戏输入命令。在系统内部包含对用户输入指令的识别、解析、然后执行，最后根据指令的类型确定是否有结果反馈。



项目整体可以看成是一种C/S架构，区分为客户端和服务器，由用户启动游戏后在终端输入一些命令。这些命令被服务器所接收识别 ，逐步经过命令接收、命令识别、命令解析、执行命令四个步骤，然后将结果反馈给用户。

整个项目主要由以下五个类构成：

Command ：用来接收用户输入的指令，内部包含基础指令(commandWord)、具体指令(secondWord)；

CommandWords ：系统内部可用指令，主要是三个指令 Quit、Go、Help；

Room ：系统中所有房间的基类，主要包含两个属性 description(房间名称)、exits(房间出口与其它房间的关系)；后续可以实现继承该类来构建不同的房间对象。

Parser ：解析终端用户输入命令的工具类。和CommandWords类是组合关系。内部主要通过Scanner对象获取用户输入，然后对输入的命令进行解析，解析成两个单词，前一个单词必须是系统内部可用指令之一，后一个单词表示具体的命令。比如说房间移动命令 go east，就表示启用 go 命令，然后向东方向移动。

Game ：游戏的主类，用户进入游戏后项目就执行此类的 play() 方法开始执行游戏。在初始化 Game类对象时执行 createRooms()方法来初始化房间，同时初始化加载 Parser工具类对象来执行命令解析的操作。

后来的test类用的是juint来单元检测go和back指令



7 实施过程问题记录与分析

测试中如何接收用户输入，首先来了解java的标准输入输出流。java.lang.System中有in(键盘输入流),out(显示器输出流),err(标准错误流)三个标准数据流: 其中System类不能创建对象，只能直接使用它的三个静态成员；当main方法执行时自动创建上面三个对象。我们在控制台交互的时候使用的就是in和out这两个标准数据流在BACK命令的测试中使用了该方法接收用户输入。如何实现BACK的回退功能，开始时遇到该问题没想到解决方案，后来查阅相关书籍发现Stack类可以很好地解决这个问题，在Game中添加栈track来记录走过的路径，每次变更房间时会将上个房间入栈，而运行Back指令时，则将回退的房间进行出栈。

# 8 任务总结

本次软件实践课程在以往计算机理论知识的学习基础上，完成了一次真正意义上的个人独立开发，开发出了zuul洞穴文字探险系统，从开始项目规范的制定到软件体系结构设计分析到后来的项目维护和拓展最终进行测试并交付运行，涵盖了软件开发工程领域的基本上所有知识，考察的不仅是编程者的代码编写能力，更是一次对编程者综合编程能力的全面考察。

本项目的开发中，对于各个流程的运作和意义也有了更加深入的了解，本次项目在锻炼我代码编写能力的基础上，以工程化的手段培养了我们的作为编程者在实际代码代码编写时应具备的能力。相信之前很多人和我会有一样的误区，在软件开发过程只有代码的编写最为重要，但事实并非如此，在软件开发过程中，需求分析、设计、开发、测试、维护等步骤都必不可少，甚至比重优于代码的编写，在系统开发过程中设计作为整个软件的骨骼，是软件开发的指南，唯有好的设计加上血肉方可产出好的产品，除此之外，软件的存活离不开持续的测试和维护，环环相扣。

在开发过程中，也认识到了软件开发规范为软件开发带来的便利，在之前的软件学习中，我只是单纯的学习了计算机和编程的相关知识，并没有切身体会到其在工程中的具体规范。在实际软件开发中，面对每个复杂的系统，项目中不同成员对于系统的设计往往都有自己的理解，这时如果没有一个统一的设计规范就可能造成了不同设计的模块之间接口与实现的差异化，而软件工程的规范化很好的解决了这一问题，通过对系统的统一的规范化描述，对于系统的开发做出了规范，规范的同时，清晰地层次和关系结构也方便后续的开发，在软件设计中起到了重要的作用。

通过实践项目，我发现自己对于很多学过的知识有些浅尝辄止，并且以前学过的知识没有完全弄明白，对有些知识虽然理论上知道，但就是用不到实际中。在进行代码编写和软件测试时，很多程序员应该具备的能力掌握还是存在问题，但是在本次实践后，很多问题都迎刃而解，对项目的开发和理解更加深入。

# 9 参考文献

[1] 邹欣. 构 建 之 法: 现 代 软 件-Y-~ Y[J]. 2014.

[2] 苏杰. 人人都是产品经理[M]. Dian zi gong ye chu ban she, 2012.

[3] Git操作<https://blog.csdn.net/u011535541/article/details/83379151>

[4] Javadoc <https://monday.blog.csdn.net/article/details/80296136>

[5] Git 提示fatal: remote origin already exists 错误解决办法<https://blog.csdn.net/top_code/article/details/50381432>

[6] Markdown思维导图<https://blog.csdn.net/jake_tian/article/details/105906542>

[7] 策略模式<https://blog.csdn.net/qq_39588630/article/details/80434240>

[8] Stack类详解：<https://blog.csdn.net/weixin_37730482/article/details/70757369>

[9] Codestyle设置<https://zhuanlan.zhihu.com/p/107656705>

**《软件工程实践（一）》成绩评定表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓 名** | | **雷霆杰** | **学 号** | **0122010880228** | |
| **专业、班级** | | **软件工程、软件2011** | | | |
| **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**