|  |  |
| --- | --- |
| **学 号：** | 0122010870109 |

****

**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2002** |
| **姓 名** | **郭成林** |
| **编 号** |  |
| **指导教师** | **唐祖锴** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2023 | 年 | 1 | 月 | 4 | 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

**说明实践/项目/系统的主要内容与任务目的**

1.1 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

1.2 任务内容

\* 阅读、理解和标注样例代码

\* 分析和学习代码质量特征、设计方法和编程风格

\* 运用所学方法，对开源代码进行标注

\* 对样例工程进行简单功能扩充和维护

# 2 任务分析

**针对实践/项目/系统开发的主要内容，分析本次实践任务的工作重点、技术方案、工程实施方案、工具使用方案、重难点分析等详细情况**

2.1 工作重点

根据本次实践任务的要求，本次实践任务的重点工作主要有：

2.1.1 Git和GitHub工具的使用

本次实验在GitHub上完成，需要掌握Git和GitHub的基本用法。

2.1.2 阅读和描述样例工程

阅读样例工程代码，分析工程结构，理解样例工程的代码逻辑，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。

2.1.3 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。

2.1.4 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。

2.1.5 markdown

本次实验需要阅读markdown文档，同时还需要编写一份用markdown写的报告REPORT.md。因此，需要在本实践中学习并掌握markdown。

2.2 技术方案

根据本次实验任务的要求，可以使用Java作为编程语言，使用Git、GitHub进行代码管理，使用SpotBugs等代码质量分析工具进行代码维护，使用JUnit测试工具进行单元测试。

2.3 实施方案

本次实践的实施过程可以根据任务要求，按顺序进行。先阅读样例工程代码，理清项目结构，绘制项目结构图及UML图。在此基础上，使用SpotBugs等代码分析工具，找出代码中的问题，对代码进行改进、扩充，并使用Git、GitHub进行版本管理。对改进、扩充后的方法，使用JUnit进行单元测试，排除错误。

2.4 工具使用方案

实验主要在Windows电脑上进行，使用Intellij IDEA作为IDE。实验使用的其他工具已在2.2节论述。

2.5 重难点分析

根据实验要求，本实验的主要重难点是在理解项目结构的基础上，对项目代码进行扩充与维护。对项目代码进行扩充与维护必须理解项目结构并灵活运用代码设计理论。同时，修改、增添的代码也应该容易扩充、维护，便于后续操作。

# 3 开发计划

**说明项目开发计划，各阶段计划采用的主要软件工程工具，以及项目遵循的主要管理策略和方法**

项目使用Intellij IDEA作为IDE进行开发。先使用Git将项目clone到本地，再在本地运行源代码，理解项目功能；并认真阅读源代码，分析项目结构，使用绘图工具，绘制项目结构图和UML图。在代码扩充与维护阶段，继续使用IDEA作为IDE，使用Git进行版本管理，使用SpotBugs等代码分析工具找出代码中潜在的问题，使用JUnit进行单元测试。

项目主要采用多次扩展、迭代开发的管理策略和方法，每次迭代实现一个功能，实现了一个功能再在此基础上继续开发其他功能。同时，项目使用Git对代码进行版本管理。

# 4 软件配置计划

**对源代码文件及版本的编码规范、命名规范、分支管理规范、提交规范等内容进行约定说明**

4.1 编码规范

项目代码应该逻辑清晰，可读性强，尽量不要使代码过于复杂；应该在必要的地方加注释，使程序结构清晰、容易理解；应遵循面向对象、模块化设计的思想，使程序可复用性提高。

4.2 命名规范

项目命名应该与实际意义有关，以提高代码的可读性。尽量不要出现用a,b,c等通用名称定义类名和变量名的情况。

4.3 提交规范

项目提交时，应该用简明、清晰的语言写明本次提交改进的内容。

# 5 测试计划

**对开发过程中涉及的单元测试及相关测试任务进行规划，对使用的测试工具、主要测试规范进行说明**

项目使用JUnit 5对代码进行单元测试。

项目中，新建一个package，名称为test，用来存放各JUnit测试类。在测试类中，通过assertTrue()、assertFalse()、assertEquals()等方法，检查测试代码能否产生正确的返回值，验证测试代码功能。

# 6 实施情况

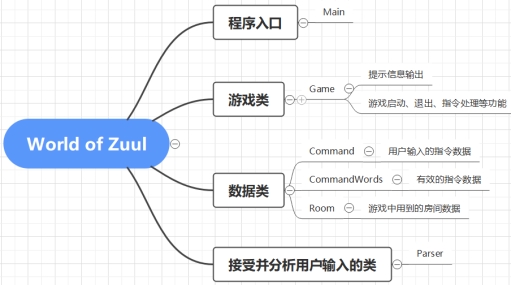
**对项目开发的过程和结果进行说明，主要包括功能设计方案及完成情况、运行结果、测试情况与修正结果、软件代码提交情况进行说明**

6.1 代码结构及部件组成

通过阅读源代码、分析源代码，可以看出，样例工程采用了面向对象的设计思想，将程序中出现的事物抽象为各种类，并为其定义相关方法。本样例工程中，共定义了Command、CommandWords、Game、Main、Parser、Room这些类。

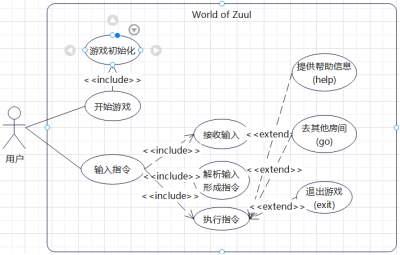
样例工程通过Main类中的main函数作为整个程序的入口。在main函数中，建立一个新的Game类对象game，并通过调用game对象的play()方法开始游戏。

在Game类中，包含了一些其他类的对象，提供了用于游戏具体逻辑实现的一些方法。这些方法通常还会调用其他类中负责逻辑实现的方法，从而实现程序运行逻辑。

项目的代码结构及部件组成可以用下图表示：

6.2 以UML图描述样例工程的组成及结构图

1 用例图

分析本样例工程，可以看出，本样例工程包括用户这一个主要人员，用户主要有开始游戏、输入指令这两个基本用例。其中，“开始游戏”用例包含了“游戏初始化”这个用例，“输入指令”用例则包含接收输入、执行指令等用例。在“执行指令”用例中，根据指令，可以扩展出一些其他用例。

2 类图

1 样例工程各类之间的关系分析

1. Main

在Main类中，生成了一个Game类对象game，并调用了game对象的方法。因此， Main和Game类间存在依赖关系，Main类依赖于Game类。

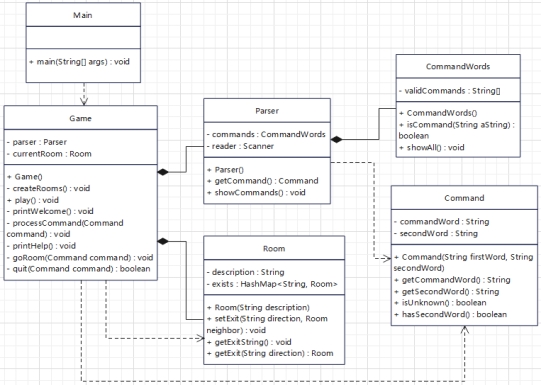
1. Game

Game类中，Parser类和Room类对象作为成员变量，表明Game和Parser、Room存在组合关系。Game类中quit()方法以Command类的对象作为参数，表明Game类与Command类之间存在依赖关系，Game类依赖于Command类。Game类中createRooms()方法建立了Room类的对象，调用了Room类方法，表明Game类与Room类存在依赖关系，Game类依赖于Room类。

1. Parser

在Parser类中，CommandWords类的对象作为成员变量，表明Parser和CommandWords类之间存在组合关系。Parser类中getCommand()方法以Command类的对象作为参数，表明Parser类与Command类之间存在依赖关系，Parser类依赖于Command类。

2 UML类图

根据上面的分析，可绘制出本样例工程的UML类图：

6.3 标注样例工程中的代码

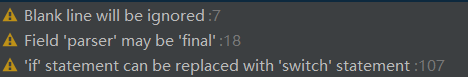
运用javadoc规范，参考Game类代码，可以将程序中的其他类代码进行注释，以提高代码的可读性。添加注释后的部分代码如图所示：

完整代码已经提交至GitHub，可在GitHub上在线查看。

6.4扩充和维护样例工程

6.4.1消除项目中的警告

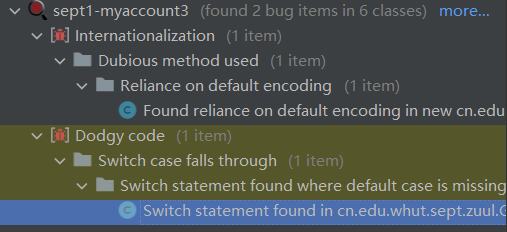
最开始的项目中有很多警告。根据警告内容，可将警告消除，从而提高代码质量。



修改后的代码已提交至GitHub。

6.4.2 使用代码质量分析工具SpotBugs检查代码

使用SpotBugs可以找到代码中潜在的问题：



6.4.3 程序优化

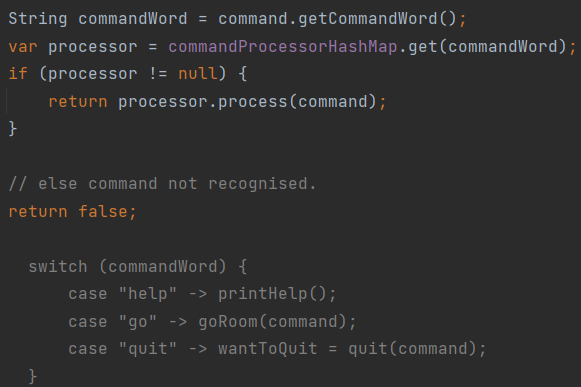
6.4.3.1 Game类中processCommand()方法的优化

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。

我的改进方法是修改对应的代码，在Game类中使用HashMap对象保存指令字符串及其对应的处理类，在processCommand()方法中通过get()函数获取处理类对象，调用该对象的process方法实现指令处理。

样例工程中，新建一个CommandProcessor接口，提供一个process()方法。指令处理类都必须实现(implements)该接口。

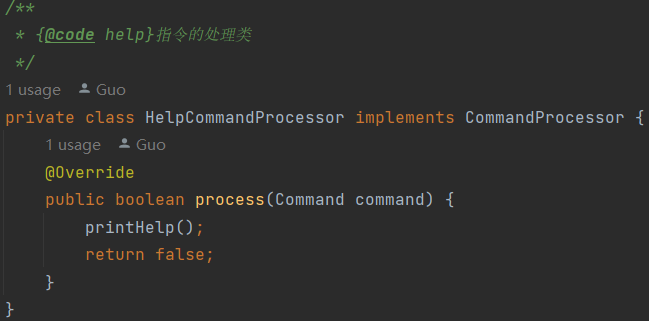
下图是修改前后的processCommand()方法对比：



可以看到，修改后的代码直接使用process()方法处理指令，避免了if-else语句和switch语句的使用，体现了面向对象的程序设计思想，将实现的具体细节封装起来，避免了面向过程的程序设计思想。

修改后的代码也提高了可复用性，以后如果要加入新的指令，这部分代码仍然适用，只需要在这部分代码之外对commandProcessorHashMap稍作修改即可，而无需像原来那样新增if-else分支或switch-case分支。

下图是新增的指令处理类，在Game类中定义：

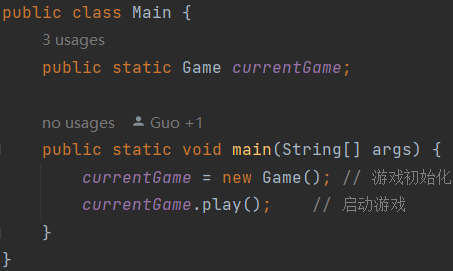


6.4.3.2对CommandWords类的优化

Game中的processCommand()方法的实现修改后，对CommandWords类也进行优化处理。

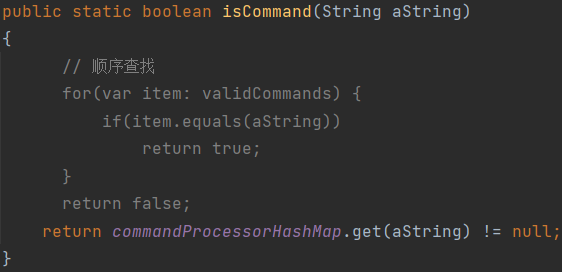
根据程序功能，CommandWords类保存所有有效的指令字符串，并提供判断指令有效性、显示索引指令的方法。由于代码修改后，在Game类中已经有保存有效的指令字符串及其处理类的commandProcessorHashMap对象，在CommandWords类中，可利用该对象改写内部方法。

在Main类中，可定义一个static公有变量currentGame，main函数中新建Game对象时，即把该对象赋给currentGame变量。



CommandWords类中，通过调用Main.currentGame.getCommandProcessorHashMap()方法保存currentGame.commandProcessorHashMap，类中方法的实现只需使用这个对象即可。

下图是代码修改后CommandWords类中isCommand()方法的实现：



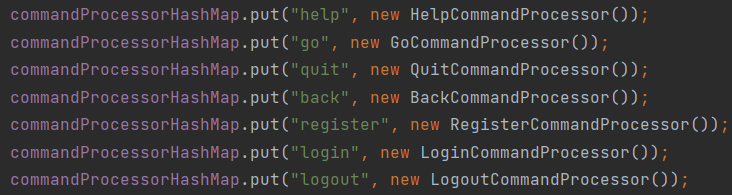
如图所示，isCommand()方法由原来的顺序查找变成了在HashMap中查找，代码量也压缩至1行。同时，相比原来的顺序查找，改进后的代码也降低了时间复杂度，提高了程序的性能。

6.5 功能扩充

6.5.1 控制台指令扩充及Player类的设计

在样例工程中，增加控制台命令及其处理类，并增设玩家(Player)类，实现程序的功能扩充。

本次指令扩充，增加了注册(register)、登录(login)、登出(logout)、返回(back)等指令，新建Player类，使程序功能更完善。

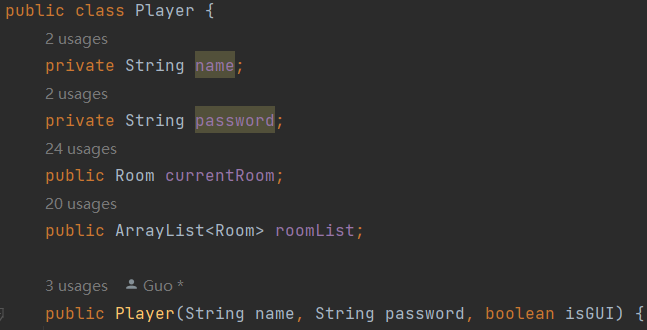


指令扩充后，程序支持用户注册、登录、登出、返回等新功能。

新功能中，注册、登录要求玩家输入用户名和密码，使用ArrayList集合保存注册后的玩家信息。程序中原来存在的go等指令则只有用户登录后才能成功执行。logout指令必须由用户登录后才能执行。back指令可以实现退回玩家到达的前一个room的功能。

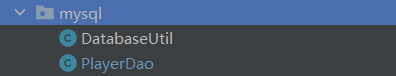
Player类保存了用户名、密码、当前房间、房间访问记录等信息。

本设计的缺点是每次停止运行后，由ArrayList保存的注册玩家信息便会丢掉。4.2将解决这个问题。



6.5.2 为游戏增加数据库功能

新建一个mysql包，存放用来连接mysql数据库及对数据库中数据进行读取、写入的类。



功能扩充后，在新用户注册时，即将用户的用户名和密码信息写入数据库。

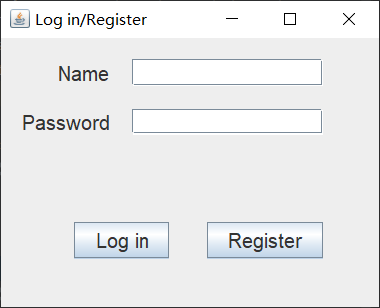
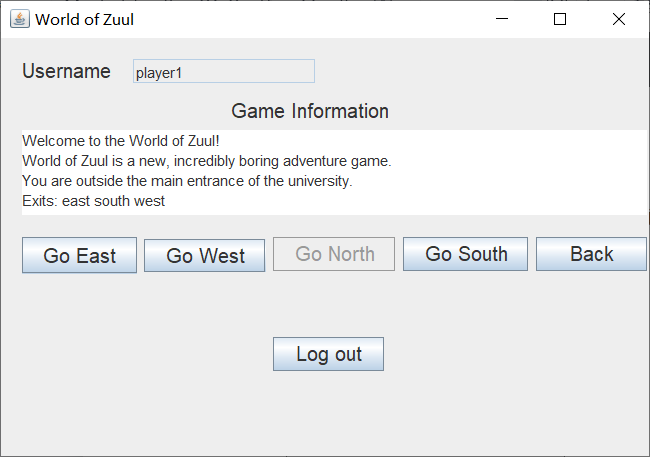
注册的玩家信息仍用ArrayList集合存储，但是是利用mysql包中的数据库读取方法，从数据库中读取数据后，生成ArrayList数据集合，以ArrayList形式返回的。

新用户注册时，既将用户信息添加到ArrayList集合，也将用户的用户名和密码信息写入数据库。

6.5.3 为游戏增加图形界面

新建一个GUI包，实现图形界面。使用java中的swing包生成图形界面。图形界面的主要功能仍参考控制台应用的实现。

下面是图形界面运行的截图：

由于图形界面所有指令均用按钮显示，可点击图形界面右上角的“×”退出，因此图形界面没有提供help和quit指令。

图形界面不仅实现了控制台应用的功能，而且更直观，人机交互更友好。同时，图形界面中的指令均通过点击按钮实现，避免了控制台应用中输入指令不合法的问题。

功能扩充后的详细代码已经提交至GitHub。

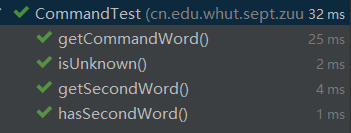
6.6 编写测试用例

利用JUnit测试工具，在程序中新增test包，存放JUnit测试类。

在测试类中，主要通过assertEquals()、assertTrue()等方法检查程序中控制台应用的各类中的方法能否输出正确结果，从而检查方法功能是否正常。



运行JUnit测试，test包中所有测试的方法均测试通过，验证了程序功能的正确性。



详细代码已提交至GitHub，可在GitHub上查看。

# 7 实施过程问题记录与分析

**对项目开发的过程中遇到的问题进行记录和分析**

7.1 Git和GitHub使用过程中的问题

实践过程中，因我对git操作还不太熟练，对实践工作造成了一定影响。另外，由于各种原因，在将代码clone到本地和push到远端时有时会出现问题。

解决方法主要有：学习git操作，熟练掌握相关内容，以应对实践任务要求；检查网络连接情况，检查是否登录GitHub；在网上查找相关资料，解决问题。

7.2 图形界面开发过程中遇到的问题

在本实践图形界面的开发过程中，有时候在点击按钮后，触发的事件响应抛出了NullPointerException。

经过分析，这主要是由于图形界面启动类MainGUI中的currentGameGUI变量未初始化，以及Player原本只针对Main类而设计有关。

解决方法主要是：将某些函数移动到构造函数外，保证currentGameGUI变量已被初始化；在Player类中新增为图形界面而设计的代码。

# 8 任务总结

**对项目实践完成的总体情况，经验或者收获等进行总结**

在本次实践中，我完成了实践任务，理解了项目的结构，对项目代码进行了扩充、维护、测试。

通过本次实践，我理解了代码规范对软件项目易用性、代码可读性的重要作用，认识到良好的代码风格、命名习惯、必要的注释可以使程序代码更容易被理解，从而也更容易扩展、维护。

我通过分析项目结构，也理解了软件项目架构设计的基本原理，在软件架构中，设计的每个类都有特定的功能，不同的类的功能不能交叉，以降低代码的耦合度，提高程序质量。同时，软件架构也不应该过于复杂，可以遵循一些比较经典的架构。

本次实践，我通过Git及GitHub的使用，熟练掌握了git的基本用法，掌握了使用Git和GitHub进行代码版本管理的基本步骤，理解了使用git进行代码管理的好处。

通过代码扩充、维护，JUnit和SpotBugs等工具的使用，我也提高了自己的代码编写能力，熟悉了软件测试工具和代码分析工具的基本使用方法。

总之，通过本次实践，我理解了代码规范的重要性，理解了代码变化对软件质量带来的影响，提高了自己的代码分析能力和代码编写能力，熟悉了Git、JUnit等工具的基本使用。

# 9 参考文献

【1】README.md (本项目)

【2】c7-样例工程结构基本说明.pdf

【3】关于Git这一篇就够了\_CSDN

【4】掌握14种UML图，清晰图示\_CSDN

【5】使用神器Spotbugs，轻松入门静态代码分析\_CSDN

【6】JUnit的使用\_CSDN

【7】Markdown入门指南【我为什么要推荐你学习Markdown？】\_CSDN

**《软件工程实践（一）》成绩评定表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓 名** | | **郭成林** | **学 号** | **0122010870109** | |
| **专业、班级** | | **软件2002** | | | |
| **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**