

学 号:

0122010870321

武汉理工大学

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

王宇轩

编 号

0122010870321

指导教师

唐祖锴

2023 年 1 月 3 日

目录

1	任务概述.....	1
2	任务分析.....	3
3	开发计划.....	3
4	软件配置计划.....	5
5	测试计划.....	7
6	实施情况.....	9
7	实施过程问题记录与分析.....	16
8	任务总结.....	18
9	参考文献.....	19

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

任务内容及要求：

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；

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. 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；

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.

5. 编写测试用例

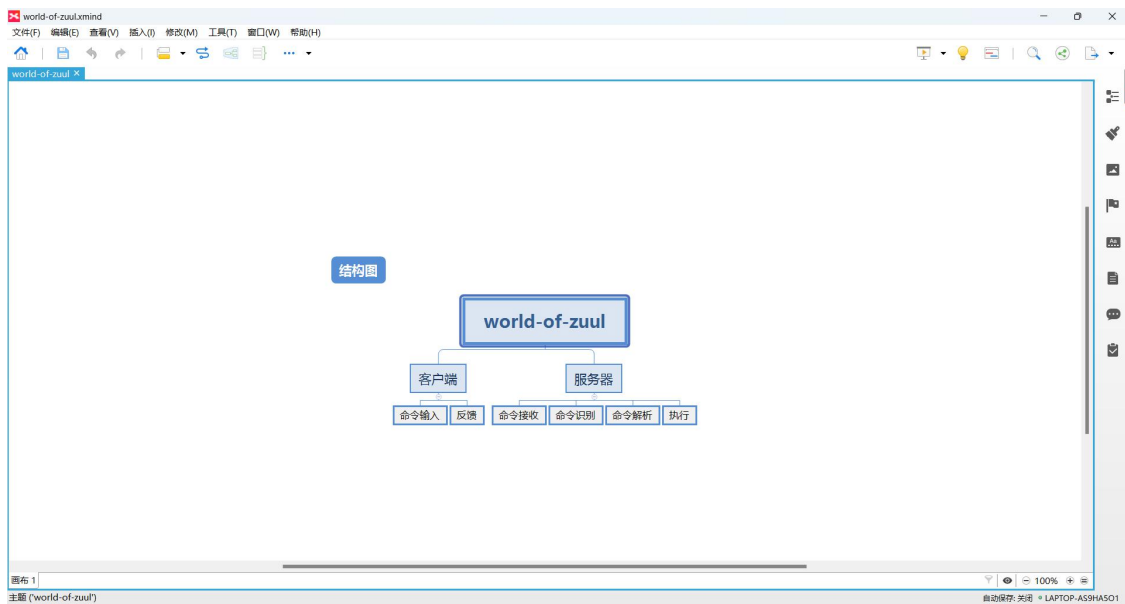
针对功能改进和扩充，在项目结构中编写单元测试用例，对代码执行单元测试。

2 任务分析

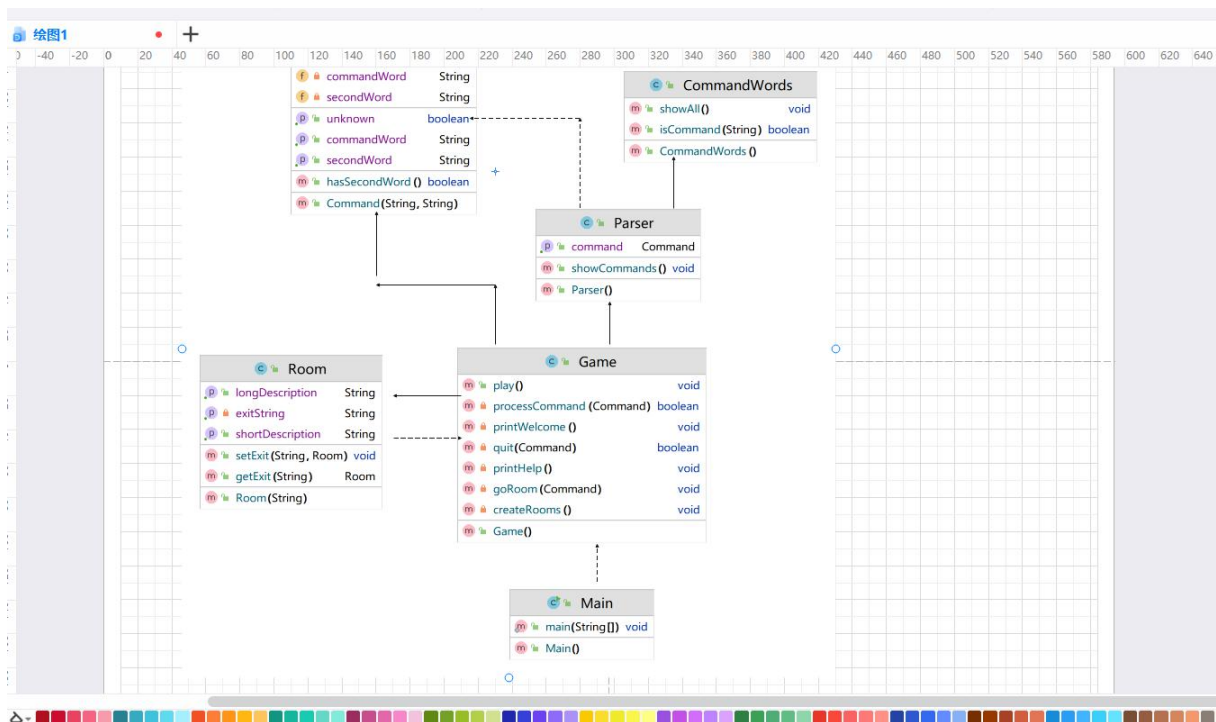
本次软件工程实践主要内容围绕字符界面的探险游戏 world-of-zuul, 项目的开发对学生的软件代码编写能力、软件框架设计能力、软件管理、代码规范、运行维护都做出了相应的要求, 所有开发工作基于 github 平台完成, 每位同学在自己项目根目录下创建一个名称为 REPORT.md 的文件, 以 markdown 语法格式编写本实训过程完成的主要任务说明, 主要包含样例工程的代码结构分析 (可以用 UML 类图及文字进行说明), 以及自己改进、扩展的功能实现说明, 单元测试用例说明。我通过迭代开发, 使用 IntelliJ IDEA 2022.1.1 开发工具采用面向对象的设计方法进行设计, 首先分析结构, 在分析好结构的基础上画了类与类之间的关系 (类图), 然后按照要求写注释, 最后进行对该项目维护、功能的拓展和测试。

3 开发计划

1. 阅读实验指导书, 理解实验目的并做初步计划
2. 阅读并理解样例代码, 用 XMind 画出结构图

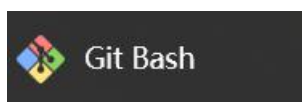


3. 通过 Javaboc 对代码中实现的方法添加注释，并用 IntelliJ IDEA 2022.1.1 自带的类图建模以及亿图图示画出类图建模。



3. 使用 IntelliJ IDEA Community Edition 2022.1.1 来进行软件开发

4. 对代码更改之后使用 Github 进行版本管理，利用 Git Bush



5. 开发完成后利用 Junit 进行代码测试

4 软件配置计划

命名规范和代码规范参考“阿里巴巴编码规范”

阿里巴巴编码规范

原创 望山。 于 2021-12-22 15:51:35 发布 1801 收藏 6

文章标签： 阿里巴巴编码规范

文章目录

- 命名风格
- 常量定义
- 代码规范
- OOP规范
- 集合处理
- 并发处理
- 控制语句
- 注释规范
- 其他
- 异常日志
 - 异常处理
 - 日志规约
- 单元测试
- 安全规约
- Mysql数据库
 - 建表规约

源代码文件规范：[1]

1. Java 源文件的后缀必须是 .java
2. Java 允许在一个 Java 源文件中编写多个类，但至多只能有一个类使用 public 修饰。
3. 如果有一个类是 public 类，那么源文件的名字必须与这个类的名字完全相同且扩展名是 .java 。

4. 如果源文件中有多类，但没有 `public` 类，那么源文件的名称只要和某个类的名称相同（与源文件同名的类被默认为 `public` 类），并且扩展名是 `.java` 就可以了。

5. Java 源文件名可以不是主类名，但一定是 `public` 类名。

6. 编译源文件时将有多类的源文件生成多个扩展名为 `.class` 的字节码文件，每个字节码文件的名称与源文件中对应的类的名称相同，这些字节码文件的被存放在与源文件相同的目录中。

命名规范：[2]

代码命名不能以下划线或者 `$` 开头或者结尾

代码命名不能以中文拼音或者中文拼音与英文混合方式

类名使用 `UpperCamelCase` 风格，但 `DO`、`PO`、`DTO`、`VO`、`BO` 等除外

方法名、参数名、变量名统一使用 `lowerCamelCase`，必须遵守驼峰命名

常量名全部大写，单词间用下划线隔开

抽象类必须以 `Abstract` 或者 `Base` 开头，异常类必须以 `Exception` 结尾，

测试类以测试的类的名称开头 `Test` 结尾

类型与中括号紧挨相连标示数组

`POJO` 类中布尔类型变量不要加 `is` 前缀

包名统一小写，点分隔符有且有一个自然语义单词

避免在父子类和不同代码块中采用相同变量名

避免不规范的缩写命名

在对元素命名时用完整单词组合表达其意

常量和变量命名时，表示类型放在词尾，如：`idList`、

TERMINATED_TREAD_COUNT

接口、类、方法、模块使用设计模式，命名时要体现具体模式

接口类中的方法和属性不要加任何修饰符，并加上有效的 javadoc。

代码规范

如果大括号代码为空直接' {}'，大括号内有代码则：左大括号左侧不换行，右侧换行；右大括号右侧换行，左侧如果不跟 else 等代码换行，否则不换行

小括号和字符之间不能有空格，括号内字符和运算符之间有空格 如：if (a == b)

if、for、while、do、switch 与括号之间必须有空格

任何二目、三目运算符前后必须有空格

采用 4 个空格，禁止使用 tab

注释的双斜线和内容要有空格

强制类型转换时，右括号与强制转换值之间不用空格

单行字符不超过 120 个，超过要换行

方法在定义和传参时，必须要加空格

IDE 的 text file encoding 设置为 UTF-8；IDE 中 文件的换行符使用 Unix 格式

单个方法尽量不超过 80 行

不同逻辑、不同语义、不同业务之间的代码插入一个空行分隔符

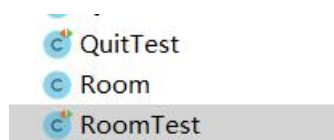
5 测试计划

软件测试的目的：[3]

软件测试的目的是为了保证软件产品的最终质量，在软件开发的过程中，对软件产品进行质量控制。测试可以达到很多目的，但最重要的是可以衡量正在开发的软件的质量。测试是为了证明程序有错，而不能保证程序没有错误。事实上，在软件运行期间测试活动从未间断，只是在软件产品交付给用户之后，将由用户继续扮演测试的角色而已。GlenMyers 在关于软件测试的优秀著作《TheArtofSoftwareTesting》中陈述了一系列可以服务于测试目标的规则，这些规则也是被广泛接受的，主要有以下 3 点。测试是一个程序的执行过程，其目的在于发现错误。一个好的测试用例很可能会发现至今尚未察觉的错误。一个成功的测试是发现至今尚未察觉的错误的测试。同时，测试不仅仅是为了发现软件缺陷和错误，也是为了对软件质量进行度量和评估，以提高软件的质量。软件测试是以评价一个程序或者系统属性为目标的活动，以验证软件满足用户的需求的程度，为用户选择与接受软件提供有力的依据。此外，通过分析错误产生的原因还可以帮助发现当前开发工作所采用的软件过程的缺陷，以便进行软件过程改进。同时，通过对测试结果进行分析整理，还可以修正软件开发规则，并为软件可靠性分析提供依据。

测试范围：

新增的 Quit 功能和 Room 类

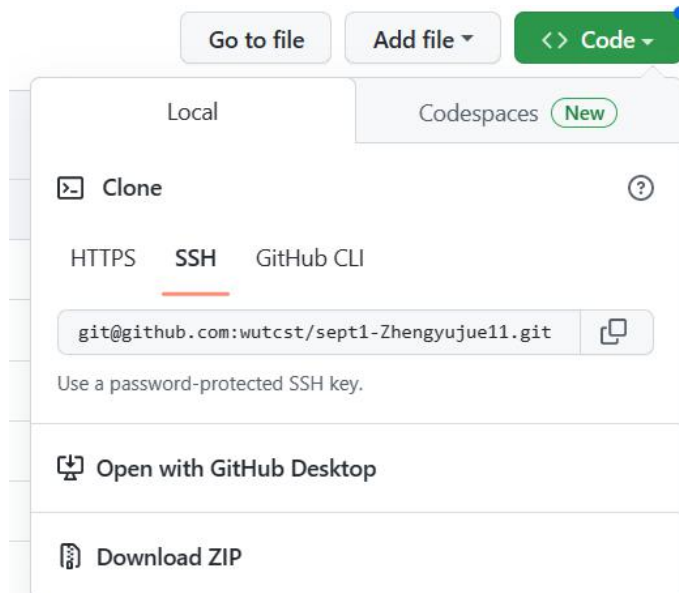


测试方法：

使用 Junit 对代码进行单元测试

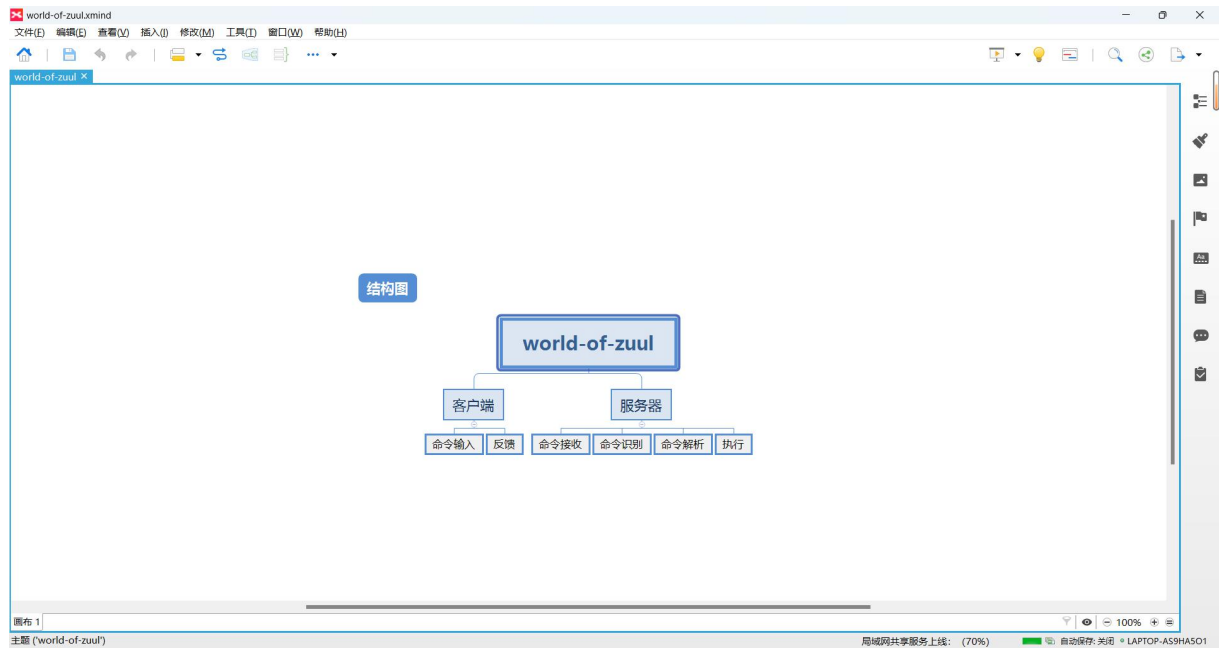
6 实施情况

1. 阅读和理解样例代码

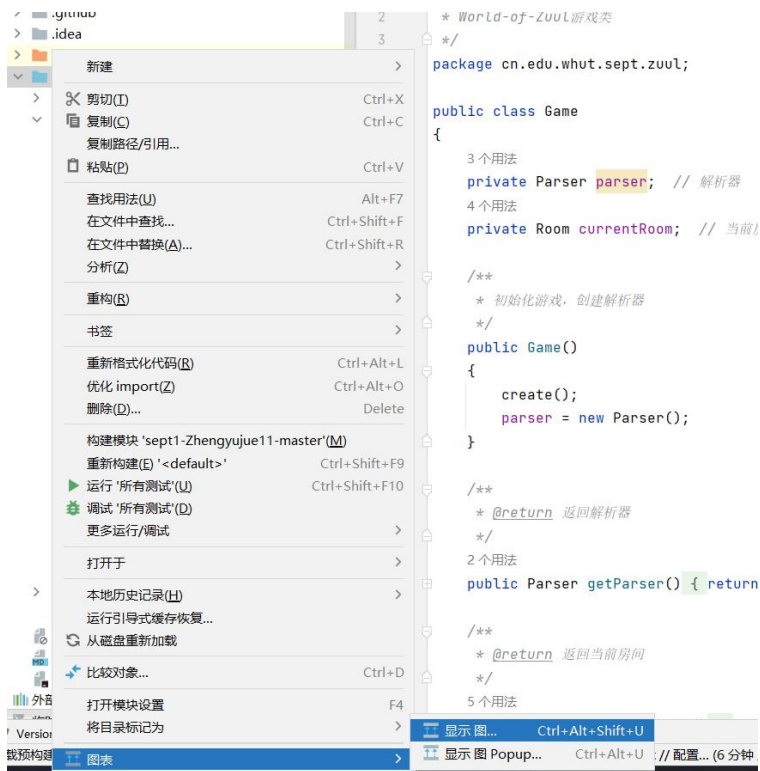


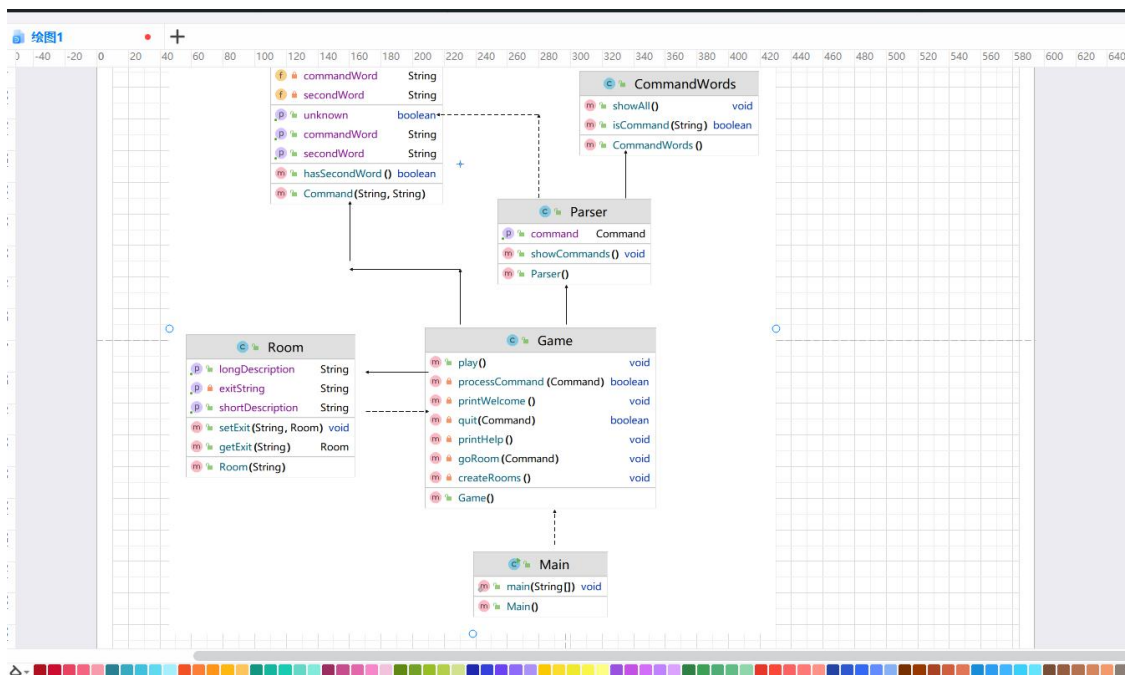
在 github 里面，Code 下点击 Download ZIP 获得老师初始给的代码框架。

然后通过阅读代码利用 XMind 画出结构图



之后利用 IntelliJ IDEA 2022.1.1 自带的类图工具结合亿图图示，绘制出类图





2. 标注样例工程中的代码

```
/**
 * 判断字符串是否为有效指令。
 * @param instructions 键入的指令
 * @return 有效指令返回true，无效返回false。
 */
1 个用法
public boolean isCommand(String instructions) {
    for(int i = 0; i < validCommands.length; i++) {
        if(validCommands[i].equals(instructions)) // 判断是否为有效指令
            return true;
    }
    return false;
}
```

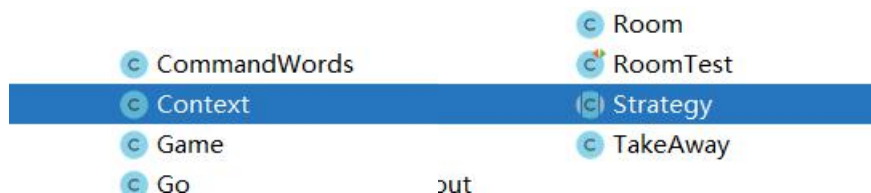
类似上图，使用/**然后回车就会自动生成注释，然后根据具体情况填充注释内容。填充过程中参考上文写的规范。

3. 改进和维护样例工程

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。

由于之前没有接触过，所以查阅了相关资料，发现代码中存在多个 if-else 语句，那么当项目中再增加多个有效指令时就会产生许多 if-else，不仅代码不美观而且性能、安全性都比较差。因此可以采取**策略模式**来减少代码中的 if-else 语句，实现算法和具体上下文的分离。策略模式需要定义一个上下文对象 Context、抽象的策略 Strategy。



具体的策略对象 go

```
/* go指令，向房间的指定方向出口移动，如果该出口连接了另一个房间，会进入该房间
*/
3个用法
@Override
public Object copeWithCommand() {
    if(!command.isSpecificCommand()) {
        // 如果用户只输入了一个动作，比如go，那么我们就无法知道去哪个方向的哪个房间，输出提示信息
        System.out.println("Where are you going?");
        return "I don't know where do you want to go.";
    }

    String direction = command.getSecondWord();

    Room nextRoom = game.getCurrentRoom().getExit(direction);

    if (nextRoom == null) {
        System.out.println("There is no exit, Please look for other exits.");
    }
    else {
        game.setCurrentRoom(nextRoom);
        System.out.println(game.getCurrentRoom().getDetailedDescription());
    }
    return "Success!";
}
```

具体的策略对象 help

```

5
6 public class Help extends Strategy{
    1 个用法
7     private Game game = getGame();
8
    1 个用法
9     public Help(Command command, Game game) { super(command,game); }
12
13     /**
14      * help指令, 打印游戏帮助信息
15      */
    3 个用法
16     @Override
17     public Object copeWithCommand() {
18         System.out.println("You wander around at the university.");
19         System.out.println();
20         System.out.println("Your command words are:");
21         game.getParser().showCommands();
22         System.out.println("");
23         return null;
24     }
25
26 }

```

优化前需要多个 if-else，代码不美观，性能低，优化后只需要新增对应的策略，然后传入 Command、Game 对象即可。

4. 扩充样例功能

在 Goods 类中增加了物品重量


```

/**
 * @return 返回物品重量
 */
1 个用法
public double getWeight() {
    return weight;
}

/**
 * 设置物品
 */
public void setDescription(String description) {
    this.description = description;
}

/**
 * 设置物品重量
 */
public void setWeight(float weight) {
    this.weight = weight;
}

```

在游戏中增加了 back 命令，可以回到上一个地方

```

/**
 * 返回
 */
package cn.edu.whut.sept.zuul;

public class Back extends Strategy{

    5 个用法
    private Game game = getGame();

    1 个用法
    public Back(Command command, Game game) { super(command, game); }

    /**
     * 执行back命令，回到上一个房间
     */
    3 个用法
    @Override
    public Object copeWithCommand() {
        String second = game.getParser().getSecond();
        if(second == null)
        {
            System.out.printf("You're in the first room.");
            return null;
        }
        Command command = new Command("go", null);
        System.out.println("You are back.");
    }
}

```

增加了 takeaway 功能


```

/**
 * 拾取
 */
package cn.edu.whut.sept.zuul;

public class TakeAway extends Strategy{
    1个用法
    private Command command = getCommand();
    1个用法
    private Game game = getGame();

    1个用法
    public TakeAway(Command command, Game game) { super(command,game); }

    /**
     * 查看当前房间信息
     * 以及房间内物品信息
     */
    3个用法
    @Override
    public Object copeWithCommand() {
        String takedes = command.getSecondWord();
        if(takedes == null) {
            System.out.println("Please input the goods.");
        }

        if(game.getCurrentRoom().deleteGoods(takedes) == true){
            System.out.println("You have taken it.");
        }
        else System.out.println("There is no such goods.");
    }
}

```

5. 编写测试用例

对于 quit 测试

```

/**
 * 退出测试
 */
package cn.edu.whut.sept.zuul;

import static org.junit.Assert.*;

import org.junit.Test;

public class QuitTest {

    2个用法
    private static Game game = new Game();
    1个用法
    private static Command command1 = new Command("quit", "");
    1个用法
    private static Command command2 = new Command("quit", "north");
}

```

对于 room 类测试

```

/**
 * 房间测试
 */
package cn.edu.whut.sept.zuul;

import static org.junit.Assert.*;

import org.junit.Test;

public class RoomTest {

    4个用法
    private Room room1 = new Room( description: "outside");
    1个用法
    private Goods goods1 = new Goods( description: "cup", weight: 1.5);

    @Test
    public void test1() {
        assertEquals( expected: "outside", room1.getBriefDescription());
    }
}

```

7 实施过程问题记录与分析

实验过程中由于我使用 Github 不熟练，经常会遇到问题

比如在把本地代码上传到 Github 上时，我使用 Git Bash 遇到了如下的问题

```

26247@LAPTOP-AS9HA501 MINGW64 ~/Desktop/实践 (master)
$ git pull origin master
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

```

后来自己思考无果，去网络上搜索了问题

解决git@github.com: Permission denied (publickey)

原创 lbaihao 于 2022-07-31 14:50:49 发布 2453 收藏 4 版权

分类专栏: git_gerrit 文章标签: github ssh



git_gerrit 专栏收录该内容

0 订阅 1 篇文章 订阅专栏

打开文件夹C:\Users\Administrator\.ssh (Administrator是当前用户名)，在空白处点鼠标右键选择“Git Bush Here”，打开gitbush。

1.首先，如果你没有ssh key的话，输入命令：`ssh-keygen -t rsa -C "xx@example.com"`，youremail@example.com改为自己的邮箱即可，途中会让你输入密码啥的，不需要管，一路回车即可，会生成你的ssh key。（如果重新生成的话会覆盖之前的ssh key。）

2.然后再执行命令：`ssh -v git@github.com`

最后两句会出现：

No more authentication methods to try.

Permission denied (publickey).

3.这时候再下输入：

`ssh-agent -s`

然后会提示类似的信息：

按照这篇博客完美解决了。解决之后如下

```
MINGW64: c:/Users/26247/Desktop/实践
26247@LAPTOP-AS9HA501 MINGW64 ~/Desktop/实践 (master)
$ git pull origin master
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 28 (delta 3), reused 16 (delta 0), pack-reused 0
Unpacking objects: 100% (28/28), 170.24 KiB | 357.00 KiB/s, done.
From github.com:wutcst/sept1-Zhengyujue11
* branch      master      -> FETCH_HEAD
* [new branch] master      -> origin/master
26247@LAPTOP-AS9HA501 MINGW64 ~/Desktop/实践 (master)
$
```


后来又遇到了 Windows 下 Git 使用报错：warning:LF will be replaced by CRLF in *****。自己解决不了又去博客上寻找，最后完美解决。

warning: LF will be replaced by CRLF in xxxx.xx(文件名)
The file will have its original line ending in your working directory.

翻译:

在xxx.xx文件中LF将被CRLF替换。

在工作区 (working directory) 里, 这个文件将会保持它原本的换行符。(line ending:行尾, 换行)

 复制代码

注解:

LF: Line Feed 换行

CRLF: Carriage Return Line Feed 回车换行键

1、不同操作系统下, 处理行尾结束符的方法是不同的:

windows下: CRLF (表示句尾使用回车换行两个字符, 即windows下的"\r\n"换行)

unix下: LF (表示句尾, 只使用换行)

mac下: CR (表示只使用回车)

通过这几次报错, 我熟练了对 Git 的理解和使用。

8 任务总结

通过本次实验, 我收获很多, 刚开始我并没有接触过 Github, 只是听说 Github 是一个特别好的开源和私有软件项目的平台, 但是没有具体学习, 这次软件工程实践我从注册开始一步一步了解 Github, 也上诸如b站、CSDN、百度等多方面平台来查阅和搜索相关课程, 并且请教用过的同学, 最终完成了第一次使用 Git 进行代码管理。在代码分析和功能扩展中, 我发现了自己的不足, 对于自己开发一个小型项目来说, 思路是很关键的, 我没有思路, 这种情况下我上网查阅了相关资料, 包括这个游戏是做什么的、代码优化的策略、单元测试的知识等等, 基本完成了这个小型项目。当然我

的项目存在很多不足之处，通过这次实践我意识到了需要有条理的进行项目的开发，软件开发也不是只有敲代码，也包括其他的很多内容，比如阅读文档、总结报告、优化设计等等，这对我今后的学习或者工作帮助很大。

9 参考文献

[1] https://blog.csdn.net/weixin_56070844/article/details/124038820

[2] https://blog.csdn.net/weixin_53244569/article/details/122087653

[3] 软件测试（第 2 版）佟伟光 2015 年版 人民邮电出版社

视频资源：

https://www.bilibili.com/video/BV1yo4y1d7UK/?spm_id_from=333.337.search-card.all.click&vd_source=742cbdea4e6205927f0772e8c95869ff

《软件工程实践（一）》成绩评定表

姓 名	王宇轩	学 号	0122010870321	
专业、班级	软件工程 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