学　　号：	0122010870302

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	林日奋
编　　号	0122010870302
指导教师	唐祖锴

2022 年 1 月 2 日

目录

、

1 任务概述

1.1 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

1.2 任务内容

（1）阅读、理解和标注样例代码

（2）分析和学习代码质量特征、设计方法和编程风格

（3）运用所学方法，对开源代码进行标注

（4）对样例工程进行简单功能扩充和维护

2 任务分析

2.1 工作重点

本次任务主要内容是基于“ world-of-zuul”游戏的原始内容展开软件开发，进而学习软件工程实践的一系列任务内容。首先需要我们通过阅读和描述样例工程，精读并理解代码结构，仿照已有注释完善不足的注释部分，结合 UML 知识进行类图建模等等。然后需要我们找出若干设计缺陷和改进点，并进行修正或扩充，选择列举的若干个功能进行扩充实现。最后对自己的扩充内容进行测试类的编写进行单元测试，并完成实践报告。

2.2 工具使用方案

软件开发工具个人选择使用的是 JAVA 语言的 IntelliJ IDEA 2021.2.1 软件进行开发。版本管理工具个人选择使用 Git 并通过代码托管平台 GITHUB 实现代码管理。

2.3 技术方案

我们首先分析在第一个版本中存在的类，找出每个类的用途。工程中有 5 个类:Parser、Commandworld、Cormmand、Room 和 Game。通过阅读已有的源代码可以发现，部分类如 Game 已经有了很好的注释，读懂代码后我们需要具体注释的规范性完成剩余类的注释。完成所有注释后只需要阅读每个类前面的注释就可以很好地了解其功能通过阅读源代码，查看每个类有什么样的方法，以及部分方法的功能，也有助于我们对游戏的理解。

Commandworld，该类通过存储一个命令词汇字符串数组来定义游戏中所有有效的命令。Parser。语法分析器从终端读入一行输人，将其解析为命令，并据此创建 Command 类的对象。

Command 类，一个 Command 对象代表了用户输入的命令，它有一些方法可以很容易地判断是否是有效的命令，还可以将命令中的第一个和第二个单词分离出来。

Room 类，一个 Room 对象代表游戏中的一个位置。房间可以由出口通到其他房间。

Game 类，该类是游戏的主类。它启动游戏，然后进入一个不断读取和执行输入的命令的循环。它也包括执行每一个用户命令的代码。

3 开发计划

项目开发计划共分为六个阶段，预计持续 3-4 天。

3.1 阶段一

时间：12.31-12.31 早上　预计 3 小时

主要软件工程工具：IntelliJ IDEA 2021.2.1，Git

主要内容：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。

3.2 阶段二

时间：12.31-12.31 下午　预计 3 小时

主要软件工程工具：IntelliJ IDEA 2021.2.1，Git

主要内容：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；

3.3 阶段三

时间：12.31-12.31 晚上 预计 2 小时

主要软件工程工具：IntelliJ IDEA 2021.2.1，Git

主要内容：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；

（3）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。

3.4 阶段四

时间：1.1-1.1 上午、下午 预计 6 小时

主要软件工程工具：IntelliJ IDEA 2021.2.1，Git

主要内容：功能扩充

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；

（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：

（6）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；

（7）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；

（8）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；

（9）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；

（10）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；

（11）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；

（12）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；

（13）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；

3.5 阶段五

时间：1.1 晚上 预计 1 小时

主要软件工程工具：IntelliJ IDEA 2021.2.1，Git

主要内容：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；

3.6 阶段六

时间：1.2 早上 预计 1 小时

主要软件工程工具：IntelliJ IDEA 2021.2.1，Git

主要内容：编写本实训任务的报告，对前面各阶段进行检查，最后完成所有 github 平台的任务提交。

4　软件配置计划

4.1 编码规范

（1）源文件编码格式（包括注释）必须是 UTF-8，ASCII 水平空格字符(0x20，即空格)是唯一允许出现的空白字符，制表符不用于缩进。

（2）所有标识符仅使用 ASCII 字母和数字，名称由正则表达式匹配。

（3）Javadoc 用于每一个 public 或 protected 修饰的元素。

（4）对于方法有参数、返回值、异常等信息时，必须在 Javadoc 块中描述：功能、@param、@return、@throws 等。

（5）Javadoc 标签与内容间只 1 个空格,按@param，@return，@throws 顺序排列。

（6）文件头注释必须包含版权许可，顶层 public 类头有功能说明、创建日期

（7）禁止空有格式的方法头注释

（8）注释正文与其下的各个 Javadoc tag 之间加 1 个空行；类级成员注释与上面的代码之间有一个空行；注释符与注释内容间要有 1 空格；右置注释与前面代码至少 1 空格。

4.2 命名规范

（1）包名称小写以点号隔开，包名允许有数字。

（2）类、枚举和接口名称采用首字母大写的驼峰命名法

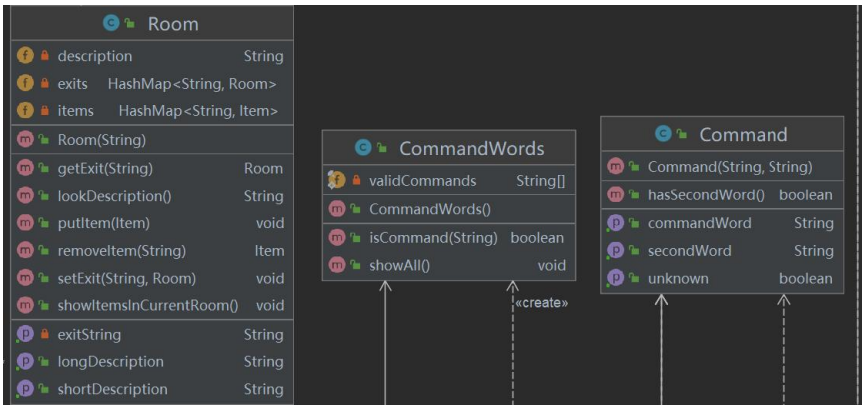
（3）方法名称采用首字母小写的驼峰命名法

（4）常量名称全大写单词以下划线分隔

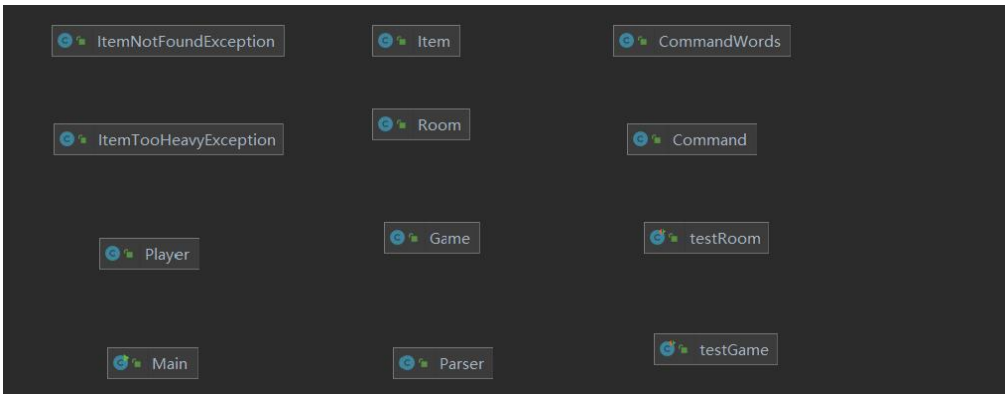
（5）非常量字段名称采用首字母小写的驼峰命名法

（6）避免使用否定的布尔变量名

代码中所涉及的部分类、方法、属性命名如图所示：



（图 1）



（图 2）

4.3 分支管理规范：
(1) master 分支:为主分支（保护分支），禁止直接在 master 上进行修改代码和提交，此分支的代码可以随时被发布到线上;
(2) dev_master 分支:为测试分支或者叫做合并分支，所有开发完成需要提交测试的功能合并到该分支，该分支包含最新的更改;
(3) dev_分支:为开发分支，大家根据不同需求创建独立的功能分支，开发完成后合并到 dev_master 分支;
(4) dev_fix 分支:为 bug 修复分支，需要根据实际情况对已发布的版本进行漏洞修复。

由于版本迭代较快，项目前期迭代版本间修改内容较少，在项目开发前期未注重该分支管理规范，开发始终在 master 分支下进行合并提交，注意到后在项目后期严格执行该规范。

4.4 提交规范
（1）git 提交 commit 的注释统一为“Version(x).(x)_新增内容”。
（2）每次完成或新增特定功能后均需要进行版本迭代和提交。

5 测试计划

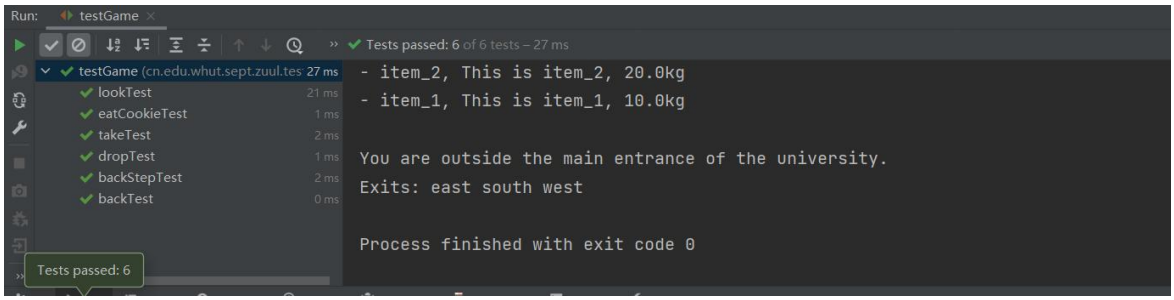
个人在实验过程中使用的测试工具为 Junit4，主要使用 Junit4 对 Game 类和 Room 类中的若干个主要逻辑方法进行单元测试。所编写的两个测试类分别命名为 testGame 和 testRoom。

testGame 中对要求补充完善的 look 命令，back 命令，高级 back 命令，take 命令，drop 命令和 eat cookie 命令进行了测试，如图所示测试均通过。

```java
/**
 * 测试Game类的drop方法.
 */
@Test
public void dropTest(){
    Game game = new Game();
    Command take = new Command("take","item_1");
    game.processCommand(take);
    Command drop = new Command("drop","item_1");
    game.processCommand(drop);
}

/**
 * 测试Game类的eatCookie方法.
 */
@Test
public void eatCookieTest(){
    Game game = new Game();
    Command eatCookie = new Command("eat","cookie_1");
    game.processCommand(eatCookie);
}
```

（图 3）



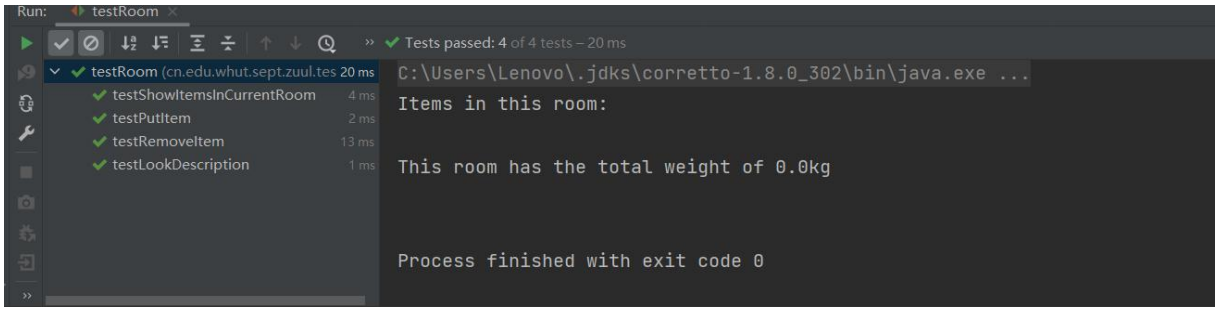
（图 4）

testRoom 中主要对要求补充和完善的 items 命令和 look 命令进行了测试，同时对从 Room 中删除和添加 Item 的两个方法也进行了测试，如图所示测试均通过。

```java
public class testRoom {
    /**
     * 测试 Room类的putItem方法.
     */
    @Test
    public void testPutItem(){
        Room room = new Room( description: "outside the main entrance of the university");
        Item item = new Item( name: "item_1", description: "This is item_1", weight: 10.0);
        room.putItem(item);
    }

    /**
     * 测试 Room类的removeItem方法.
     */
    @Test
    public void testRemoveItem() throws ItemNotFoundException {
        Game game = new Game();
        Command take = new Command("take","item_1");
        //game.processCommand(take);
        String itemFromCommand = take.getSecondWord();
        Item newItem = game.getPlayer().getCurrentRoom().removeItem(itemFromCommand);
    }
```

（图 5）

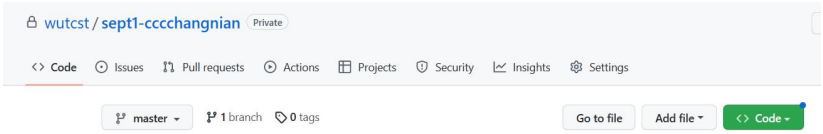


（图 6）

6　实施情况

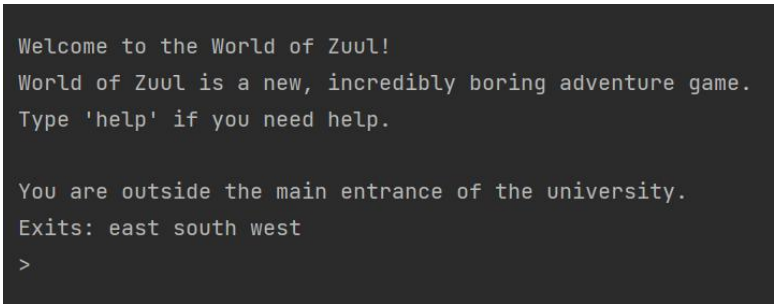
6.1 阅读和描述样例工程

(1)根据 PPT 指示在 github 申请并领取了班级任务后，将初始版本的代码 fork 到了自己仓库 sept1-cccchangnian.



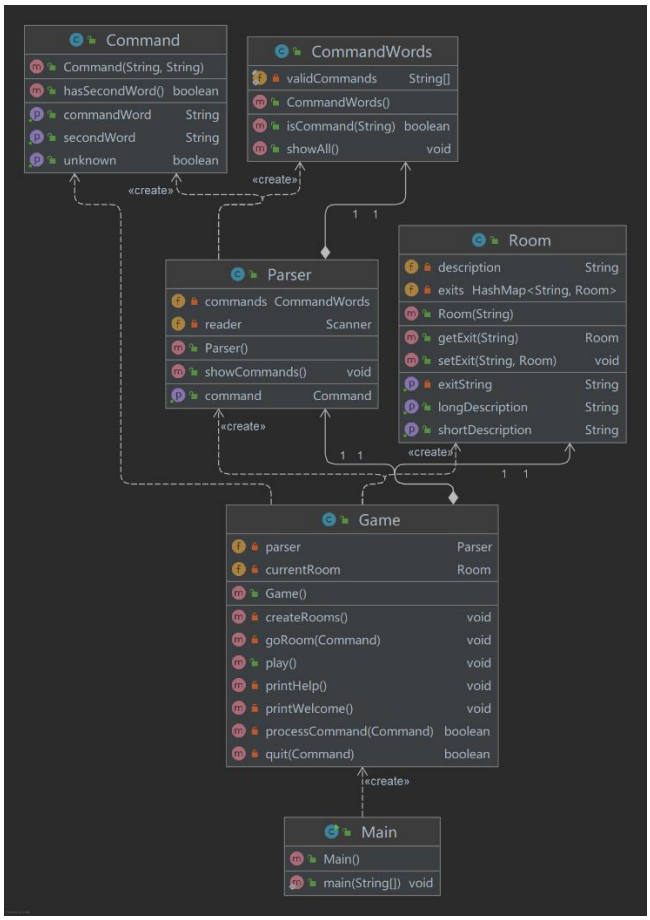
(图 7)

（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，目前游戏内容较简单，只有 go.quit,help 三个功能。



（图 8）

（3）结合所学的 UML 建模技术，建立了以下类图：



（图 9）

6.2 标注样例工程中的代码

（1）仔细阅读了代码之后，学习 Game 类中已有的注释，在剩余类中的 javadoc 中添加了注释。

```java
/**
 * 检查给定的字符串是否是有效的命令字.
 * @return 如果命令是合法的，返回 true，否则返回 false
 * 遍历匹配有效的字符数组 validCommands 即可
 */
public boolean isCommand(String aString)
{
    for(int i = 0; i < validCommands.length; i++) {
        if(validCommands[i].equals(aString))
            return true;
    }
    // 此处代表未匹配到任何有效字符
    return false;
}
```

（图 10）

（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。

类图1_version1.1.png	Version1.1_阅读和描述样例工程，标注代码	10 hours ago
类图1_version1.1.uml	Version1.1_阅读和描述样例工程，标注代码	10 hours ago
类图2.uml	Version1.1_阅读和描述样例工程，标注代码	10 hours ago
类图2_version1.1.png	Version1.1_阅读和描述样例工程，标注代码	10 hours ago
随机生成cookie.png	Version1.8_添加eatCookie功能	4 hours ago
项目架构重构.png	Version1.2_扩充和维护样例工程，重构代码	9 hours ago

（图 11）

6.3 扩充和维护样例工程

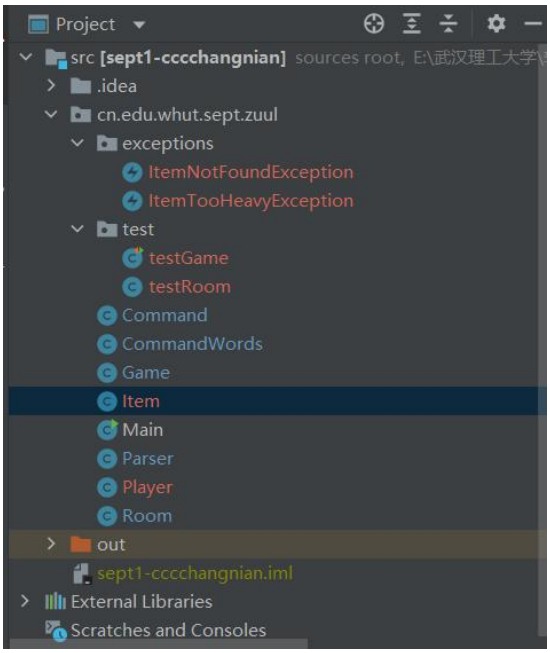
（1）修改 processCommand 方法。

```java
//将多if条件判断改为switch语句
switch (commandWord){
    case "help": printHelp(); break;
    case "go": goRoom(command); break;
    case "quit": wantToQuit = quit(command); break;
    case "look": look();break;
    case "back": back();break;
    case "backStep": backStep();break;
    case "items": printItems();break;
    case "eatCookie":
        try {
            //randomCreateCookie();
            eatCookie(command);
        } catch (ItemNotFoundException e) {
            e.printStackTrace();
        }
        System.out.println(player.getCurrentRoom().lookDescription());
        break;
    case "take":
        try {
            takeItem(command);
        } catch (ItemTooHeavyException | ItemNotFoundException e) {
            System.out.println(e.getMessage());
        }
        System.out.println(player.getCurrentRoom().lookDescription());
```

（图 12）

（2）对整体代码进行重构。

新建 Item 类和 Player 类，将部分属性和方法依据面向对象的方式改写，同时便于后续新增功能的实现。



（图 13）

6.4 功能扩充

（1）添加并实现 look 命令。

```
/**
 * 执行look指令，查看目前所处房间内的物品信息。
 */
private void look() {
    System.out.println(player.getCurrentRoom().lookDescription());
}
```

（图 14）

（2）添加并实现 back 命令。

```
/**
 * 执行back指令，从当前房间返回上一个所处房间。
 */
private void back(){
    Room nextRoom = player.getLastRoom();
    player.setLastRoom(player.getCurrentRoom());
    player.roomPath.add(player.getCurrentRoom());//添加到访问路径中
    player.setCurrentRoom(nextRoom);
    System.out.println(player.getCurrentRoom().getLongDescription());
    //判断是否触发随机传送
    if (player.getCurrentRoom() == lab){
        send();
    }
}
```

（图 15）

（3）添加并实现 take 命令。

```
/**
 * 执行take指令，从当前房间捡起物品并放入玩家背包。
 */
private void takeItem(Command command) throws ItemNotFoundException, ItemTooHeavyException {
    String itemFromCommand = command.getSecondWord();
    Item newItem = player.getCurrentRoom().removeItem(itemFromCommand);
    if ((newItem != null) && !(player.getCapacity() + newItem.getWeight() > player.getMAXHOLD())) {
        player.takeItem(newItem);
        player.increaseCapacity(newItem.getWeight());
    } else {
        System.out.println("takeItem Error: item not exist or too heavey");
        player.getCurrentRoom().putItem(newItem);
    }
}
```

（图 16）

（4）添加并实现 drop 功能。

```
/**
 * 执行drop指令，从当玩家背包丢弃物品到当前房间内。
 */
private void dropItem(Command command) throws ItemNotFoundException {
    String itemFromCommand = command.getSecondWord();
    Item newItem = null;
    newItem = player.dropItem(itemFromCommand);
    if (newItem != null) {
        player.getCurrentRoom().putItem(newItem);
        player.decreaseCapacity(newItem.getWeight());
    } else {
        System.out.println("dropItem Error: item not exist");
    }
}
```

（图 17）

（5）添加并实现 items 功能。

```
/**
 * 执行items指令，打印显示当前房间内和玩家背包中所有物件及总重量。
 */
private void printItems(){
    //打印当前房间内所有的物件及总重量
    player.getCurrentRoom().showItemsInCurrentRoom();
    //打印玩家随身携带的所有物件及总重量
    player.printItemsInBags();
}
```

（图 18）

（6）添加并实现 eatCookie 功能。

```java
/**
 * 执行eatCookie指令，将所在房间中的cookie吃掉并增加背包容量。
 */
private void eatCookie(Command command)throws ItemNotFoundException{
    String itemFromCommand = command.getSecondWord();
    Item newItem = player.getCurrentRoom().removeItem(itemFromCommand);
    if(newItem.getName().contains("cookie")) {
        player.increaseCapacity( num: 20);
        System.out.println("You have ate the cookie, your capacity increases 10 kg.");
    }
}
```

（图 19）

（7）添加并实现房间随机传送功能。

```java
/**
 * 实现房间随机传输功能。
 */
private void send(){
    Random random = new Random();
    int randNum = random.nextInt( bound: 3);
    Room[] sendToPlaces = {theater,pub,office};
    Room nextRoom = sendToPlaces[randNum];
    String roomString = new String();
    if(randNum == 0){
        roomString = "theater";
    }
    else if (randNum == 1){
        roomString = "pub";
    }
    else {
        roomString = "office";
    }
    player.setLastRoom(player.getCurrentRoom());
    player.setCurrentRoom(nextRoom);
    player.roomPath.add(player.getCurrentRoom());//添加到访问路径中
    System.out.println("You have been randomly sent to " + roomString);
}
```

（图 20）

（8）添加并实现高级 back 功能。

```java
/**
 * 执行backStep指令，实现高级的回退命令。
 */
private void backStep(){
    for(int i=player.roomPath.size()-1; i>=0; i--){
        Room nextroom = player.roomPath.get(i);
        player.setLastRoom(player.getCurrentRoom());
        player.setCurrentRoom(nextroom);
        if (player.roomPath.get(i).getShortDescription().contains("outside")){
            System.out.println(player.getCurrentRoom().getLongDescription());
            player.roomPath.clear();
            break;
        }
        else{
            System.out.println(player.getCurrentRoom().getLongDescription());
            //break;
        }
    }
}
```

（图 21）

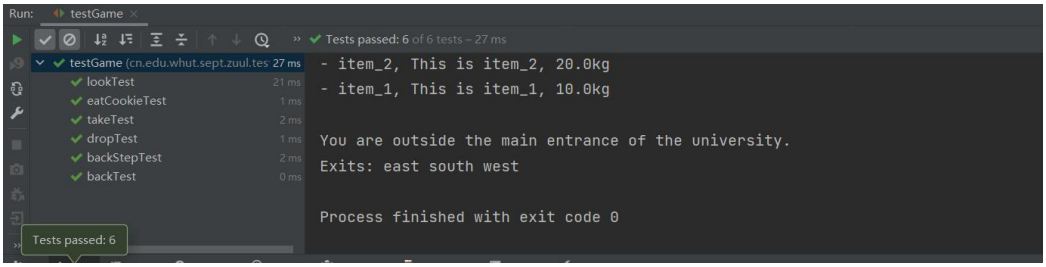
6.5 编写测试类

（1）编写 testGame 类。testGame 中对要求补充完善的 look 命令，back 命令，高级 back 命令，take 命令，drop 命令和 eat cookie 命令进行了测试，如图所示测试均通过。

```java
/**
 * 测试Game类的drop方法.
 */
@Test
public void dropTest(){
    Game game = new Game();
    Command take = new Command("take","item_1");
    game.processCommand(take);
    Command drop = new Command("drop","item_1");
    game.processCommand(drop);
}

/**
 * 测试Game类的eatCookie方法.
 */
@Test
public void eatCookieTest(){
    Game game = new Game();
    Command eatCookie = new Command("eat","cookie_1");
    game.processCommand(eatCookie);
}
```

（图 22）



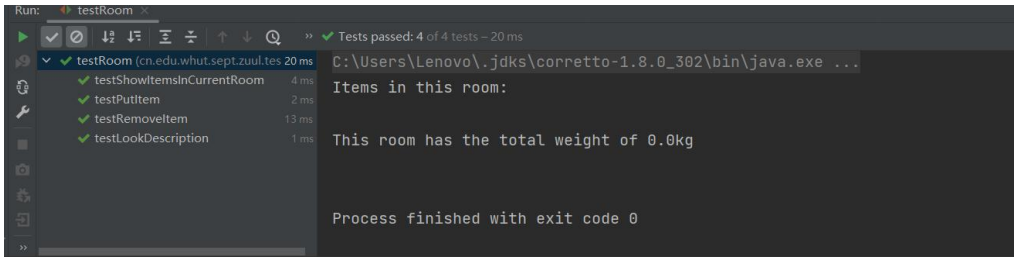
（图 23）

（2）编写 testRoom 类。testRoom 中主要对要求补充和完善的 items 命令和 look 命令进行了测试，同时对从 Room 中删除和添加 Item 的两个方法也进行了测试，如图所示测试均通过。

```java
public class testRoom {
    /**
     * 测试 Room类的putItem方法.
     */
    @Test
    public void testPutItem(){
        Room room = new Room( description: "outside the main entrance of the university");
        Item item = new Item( name: "item_1", description: "This is item_1", weight: 10.0);
        room.putItem(item);
    }

    /**
     * 测试 Room类的removeItem方法.
     */
    @Test
    public void testRemoveItem() throws ItemNotFoundException {
        Game game = new Game();
        Command take = new Command("take","item_1");
        //game.processCommand(take);
        String itemFromCommand = take.getSecondWord();
        Item newItem = game.getPlayer().getCurrentRoom().removeItem(itemFromCommand);
    }
```

（图 24）



（图 25）

6.6 代码运行结果

（1）执行 help 命令。

```
You are outside the main entrance of the university.
Exits: east south west
> help
You are lost. You are alone. You wander
around at the university.

Your command words are:
go  quit  help  look  take  drop  back  items  eatCookie  backStep
>
```

（图 26）

（2）执行 look 命令。

```
> look
You are outside the main entrance of the university.
Exits: east south west
Items in this room:
- item_2, This is item_2, 20.0kg
- item_1, This is item_1, 10.0kg
```

（图 27）

（3）执行 go 命令。

```
> go west
You are in the campus pub.
Exits: east
```

（图 28）

（4）执行 items 命令。

```
> items
Items in this room:
- item_6, This is item_6, 60.0kg- item_5, This is item_5, 50.0kg
This room has the total weight of 110.0kg

Items in player's bag :
Player's bag has the total weight of 0.0kg
```

（图 29）

（5）执行 take 命令。

```
> take item_1
You are outside the main entrance of the university.
Exits: east south west
Items in this room:
- cookie_2, This is a cookie named cookie_2, 5.0kg
- item_2, This is item_2, 20.0kg
```

（图 30）

（6）执行 drop 命令。

```
> drop item_1
You are outside the main entrance of the university.
Exits: east south west
Items in this room:
- cookie_2, This is a cookie named cookie_2, 5.0kg
- item_2, This is item_2, 20.0kg
- item_1, This is item_1, 10.0kg
```

（图 31）

（7）执行 back 命令。

```
You are outside the main entrance of the university.
Exits: east south west
> go east
You are in a lecture theater.
Exits: west
> back
You are outside the main entrance of the university.
Exits: east south west
```

（图 32）

（8）执行 eatCookie 命令。

```
> eatCookie cookie_2
You have ate the cookie, your capacity increases 10 kg.
You are outside the main entrance of the university.
Exits: east south west
Items in this room:
- item_2, This is item_2, 20.0kg
- item_1, This is item_1, 10.0kg
```

（图 33）

（9）执行高级 back 命令回退两步到达起点。

```
> go south
You are in a computing lab.
Exits: east north
You have been randomly sent to office
> backStep
You are in the computing admin office.
Exits: west
You are outside the main entrance of the university.
Exits: east south west
>
```

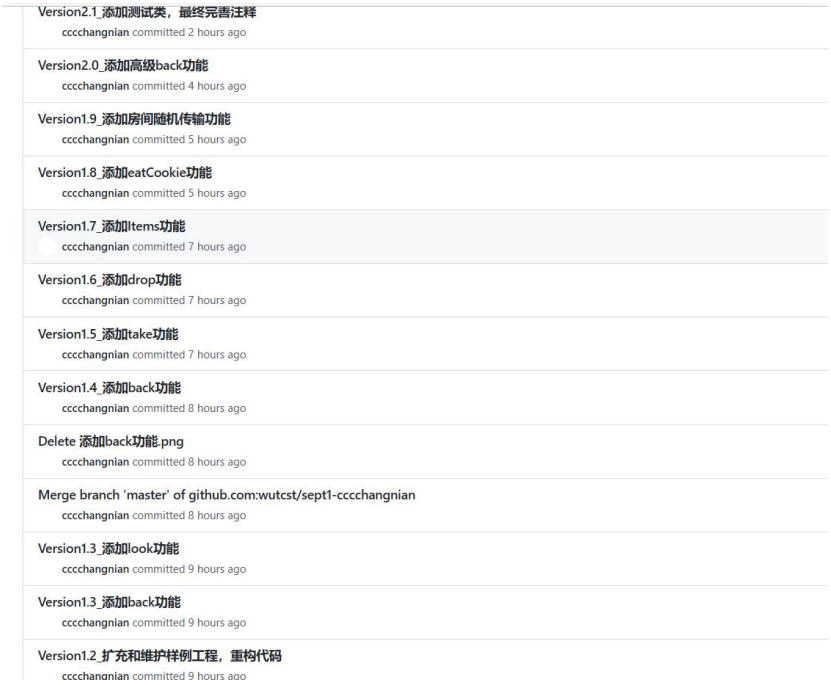
（图 34）

（10）发生随机传送现象。

```
You are outside the main entrance of the university.
Exits: east south west
> go south
You are in a computing lab.
Exits: east north
You have been randomly sent to pub
```

（图 35）

6.7 代码提交结果



（图 36）

7 实施过程问题记录与分析

（1）在使用 git add 命令的时候，弹出了一个警告 warning: LF will be replaced by CRLF in ******

通过查阅资料后得知原因是 LF 和 CRLF 其实都是换行符,但是不同的是,LF 是 linux 和 Unix 系统的换行符，CRLF 是 window 系统的换行符。这就给跨平台的协作的项目带来了问题，保存文件到底是使用哪个标准呢？ git 为了解决这个问题，提供了一个”换行符自动转换“的功能，并且这个功能是默认处于”自动模式“即开启状态的。

这个换行符自动转换会把自动把你代码里 与你当前操作系统不相同的换行的方式转换成当前系统的换行方式（即 LF 和 CRLF 之间的转换），这样一来，当你提交代码的时候，即使你没有修改过某个文件，也被 git 认为你修改过了，从而提示"LF will be replaced by CRLF in *****"

最简单的一种办法就是把自动转换功能关掉即可。

输入命令 ： git config core.autocrlf false (仅对当前 git 仓库有效）

git config --global core.autocrlf false (全局有效，不设置推荐全局）

然后重新提交代码即可。

（2）在进行 git push 的时候修改错了上次提交的内容，需要进行版本回退。

解决方法：执行 git reset --hard HEAD^

注意本地代码会被回退到上次修改前，如果本地已有修改注意备份。之后，使用 –force 或 -f 参数强制 push。

再执行 git push origin master –force

再看 git log 就没有上次的提交了。

（3）在 git bash 中键入 $ git push origin master 进行提交的时候出现如下错误：error: failed to push some refs to 'https://github.com/

通过查阅相关资料，发现问题原因是因为远程库与本地库不一致造成的，在 hint 中也有提示把远程库同步到本地库就可以了

解决办法：使用命令行：git pull --rebase origin master

该命令的意思是把远程库中的更新合并到（pull=fetch+merge）本地库中， --rebase 的作用是取消掉本地库中刚刚的 commit，并把他们接到更新后的版本库之中。出现如下图执行 pull 执行成功后，可以成功执行 git push origin master 操作。

8 任务总结

通过本次软件工程实践课,复习巩固了之前学习的如何利用 Git 进行代码版本管理，能够熟练使用常见的 Git 命令，对于命令运用过程中遇到的一系列问题能通过快速查阅到相关资料解决。同时也学习了使用 markdown 语法格式编写实训报告 REPORT.md 文档，通过该开发过程可以梳理整个项目结构和开发过程。

另外，在理解代码的过程中，充分结合所学的 UML 建模知识，建立了类图来表达代码中类及类之间的关系。同时，也学习了 javadoc 规范标注代码，并实践了如何对包、类、方法、代码片段、参数和语句等代码层次进行注释。

通过本次实践项目的开发中，使得我个人对于软件工程师在实际项目开发的各个流程的运作和意义有了更加深入的了解。在进行大型项目开发中，多人协作和代码的版本

迭代管理都是必不可缺的内容，因此熟练掌握并运用 Git 十分重要，要学会如何通过版本迭代来有节奏地进行软件开发。同时，在开发过程中也应该注重各种规范，如编码规范、命名规范、分支管理规范和提交规范等等。总而言之，通过本次实践，培养了我们作为软件开发者工程化的思维，掌握了软件开发过程中的一系列能力，收获颇丰。

9 参考文献

[1]陈冠元.软件工程中的 UML 建模技术[J].电子技术与软件工程,2018(05):47.

[2]Larman C. Applying UML and patterns: an introduction to object oriented analysis and design and interative development[M]. Pearson Education India, 2012.

[3]Fowler M. Patterns of Enterprise Application Architecture/Martin Fowler[J]. Seventeenth printing, 2011.

[4]Martin R C. Agile software development: principles, patterns, and practices[M]. Prentice Hall PTR, 2003.

[5]使用 Eslint & standard 规范代码 https://juejin.cn/post/6844903907580182541

[6] World Of Zuul. https://sourceforge.net/projects/worldofzuul/

[7]javadoc 注释规范. https://blog.csdn.net/h_xiao_x/article/details/65936510

[8]ESLint. https://www.eslint.com.cn/

[9]markdown 语法格式. https://blog.csdn.net/VistorsYan/article/details

[10]JAVA 单元测试. https://blog.csdn.net/qq_50313418/article/details/

[11]Git 错误解决. https://blog.csdn.net/man_zuo/article/details/88651416

《软件工程实践（一）》成绩评定表

<table>
<tr><td colspan="2">姓　　名</td><td>林日奋</td><td>学　　号</td><td colspan="2">0122010870302</td></tr>
<tr><td colspan="2">专业、班级</td><td colspan="4">软件工程、软件 2002</td></tr>
<tr><td colspan="6">成绩评定：</td></tr>
<tr><td colspan="2" rowspan="2">评价内容</td><td rowspan="2">满分</td><td colspan="2">实得分</td></tr>
<tr><td>得分</td><td>小计</td></tr>
<tr><td rowspan="5">实践任务
完成情况</td><td>软件项目设计、改进与扩充</td><td>20</td><td></td><td rowspan="5"></td></tr>
<tr><td>个人软件过程与项目管理</td><td>15</td><td></td></tr>
<tr><td>代码版本管理</td><td>25</td><td></td></tr>
<tr><td>代码注释与编码规范</td><td>25</td><td></td></tr>
<tr><td>单元测试</td><td>15</td><td></td></tr>
<tr><td rowspan="5">实践报告
总评情况</td><td>学习态度与考勤</td><td>10</td><td></td><td rowspan="5"></td></tr>
<tr><td>报告格式的规范性</td><td>10</td><td></td></tr>
<tr><td>报告的逻辑结构与语言表达</td><td>15</td><td></td></tr>
<tr><td>实践内容的正确性与合理性</td><td>60</td><td></td></tr>
<tr><td>文献引用及标注</td><td>5</td><td></td></tr>
<tr><td colspan="2">总分</td><td>100</td><td></td><td></td></tr>
<tr><td colspan="6">最终评定成绩（以优、良、中、及格、不及格评定）</td></tr>
</table>

指导教师签字：

年　月　日