DAWN

Software Design Model

Version 1.0

**By:**

Dawn Group

2019-04

**Group Member:**

Zihan Xu

Yi Kuang

Chenyu Yang

Yuting Lan

Jianzhen Cao

**Document Language:**

English

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2019-4-16 | 1.0 | Finish the 1st edition of Software Design Model | Zihan Xu, Yi Kuang, Chenyu Yang, Yuting Lan, Jianzhen Cao |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1 Introduction 1](#_Toc6648951)

[1.1 Purpose 1](#_Toc6648952)

[1.2适用范围 Domain 1](#_Toc6648953)

[1.3定义 Definition 1](#_Toc6648954)

[1.4参考资料Reference 1](#_Toc6648955)

[1.5概述 cjz Overview 1](#_Toc6648956)

[2． 用例视图Use Case View 2](#_Toc6648957)

[3. 逻辑视图Logical View 2](#_Toc6648958)

[3.1 系统结构System structure 2](#_Toc6648959)

[3.2 用例视图Use Case View 5](#_Toc6648960)

[3.3 设计视图Design View 10](#_Toc6648961)

[3.4 其他图(NULL) 13](#_Toc6648962)

[4. 实现视图Implementation View 13](#_Toc6648963)

[5. 进程视图 Process View 19](#_Toc6648964)

[6. 部署视图 Deployment View 20](#_Toc6648965)

Software design patten

# 1 Introduction

## Purpose

说明：编写这份软件设计模型的目的，并指出预期的读者。

The purpose of this software design model document is to organize various design models of our game. Based on the previous requirements specification and system analysis, the structure of the system is displayed in detail, laying the foundation for the subsequent software implementation. We will construct logical views, implementation views, process views and deployment views. We will be able to develop the whole project based on it.

## 1.2适用范围 Domain

The Dawn game system. The characteristics, subsystems and models that related to the system meet the content of this document

## 1.3定义 Definition

列出本文件中用到的专门术语的定义和外文的首字母缩写词。可以引用项目词汇表来提供。

See more details in glossary document.

## 1.4参考资料Reference

列出本文中各处引用的文档资料，包括每个文档的标题、文档编号、发表日期和出版单位并列出能够得到这些文件资料的来源。

<<Object-Oriented Software Engineering Practice Guide-2>> Shanghai Jiao Tong University Press, 2016

<<Object-Oriented Software Engineering - Using UML, Patterns, and Java>> (3rd edition), Tsinghua University Press, 2011

## 1.5概述Overview

说明本文件中其他各部分包含的内容，与本文件的内容组织方式。

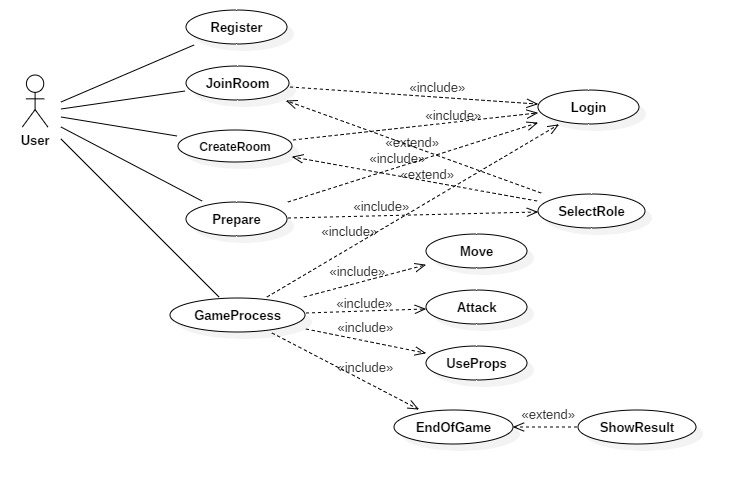
This document includes introduction, Use-case Diagram, Logic Diagram, Implementation Diagram, Process View and Deployment View. The Use-case Diagram displays functions provided by our app. Logical View talks about classes of each subsystem and Implementation Diagram displays how the functions are realized with some components. Process View lists each thread of the process. Deployment View shows an arrangement in physical level.

# 2． 用例视图Use Case View

尽管Use-Case主要是分析阶段的产物，但是将Use-Case图放在模型中，便于理解后续的设计。在此部分中，需要提供一张用例图。

可以补充活动图来说明业务逻辑。

The complete use case model:

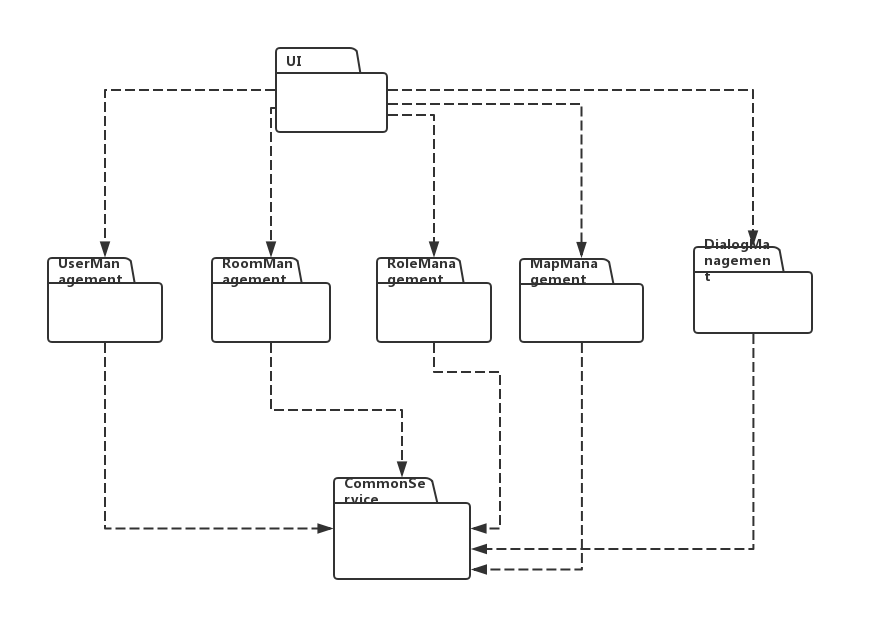


# 3. 逻辑视图Logical View

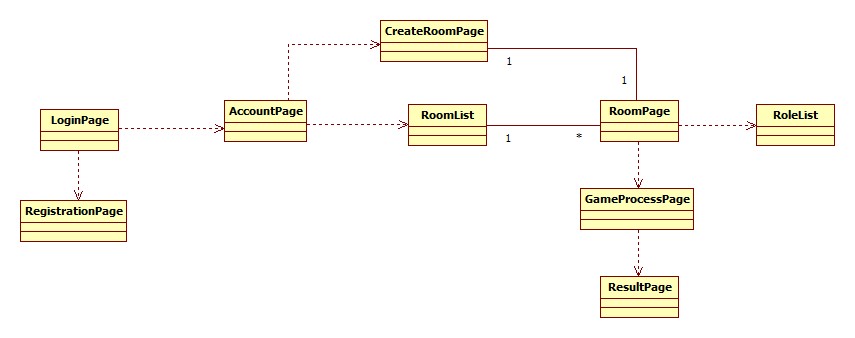
## 3.1 系统结构System structure

首先利用UML的包图，画出一个系统架构的表示图。

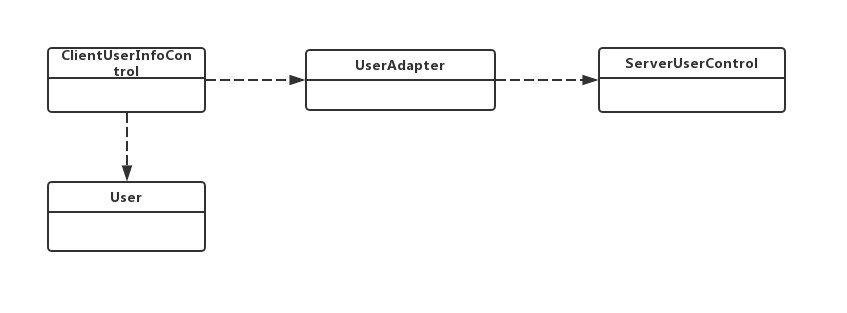
针对每一个包，画一张它所包含的类的类图。



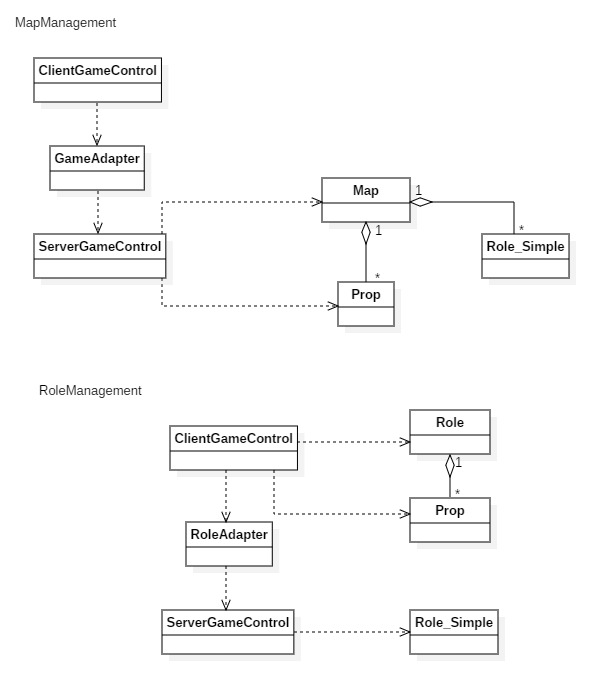
1. User-Interface Subsystem



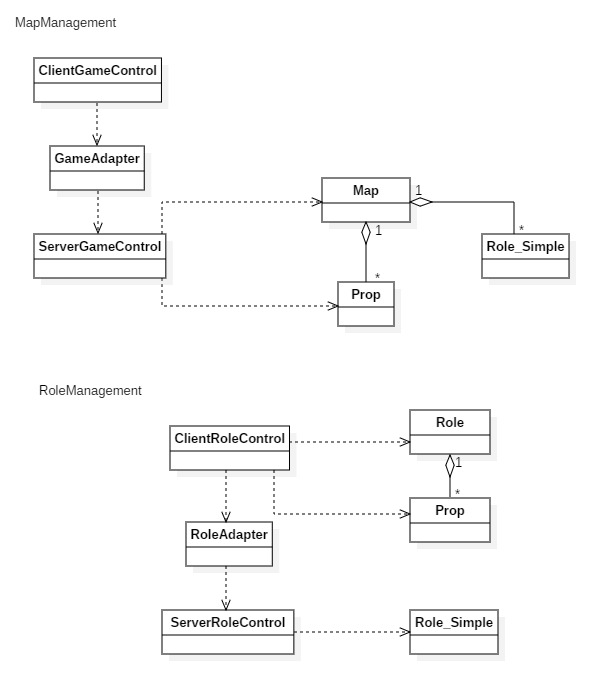
1. User-management Subsystem



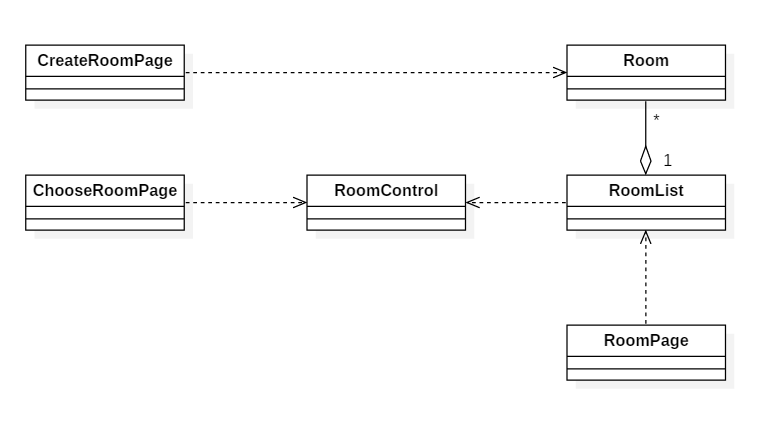
1. Role-management Subsystem



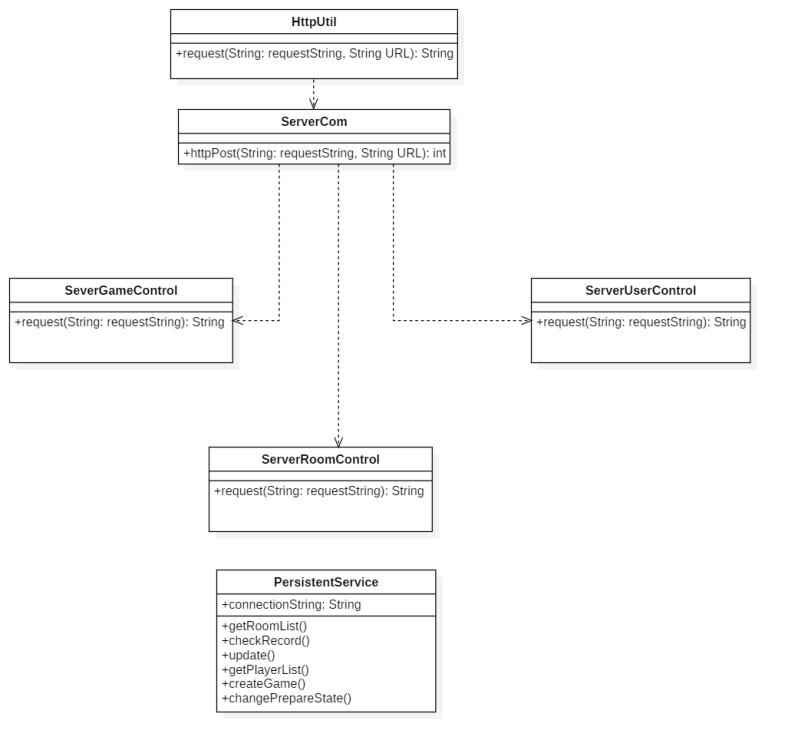
1. Map-management Subsystem



1. Room-management Subsystem



1. Common-service Subsystem



## 3.2 用例视图Use Case View

在该节中需要针对每一个Use-Case，通过交互图的方式表达相应的设计。因此，其格式可以表示为：

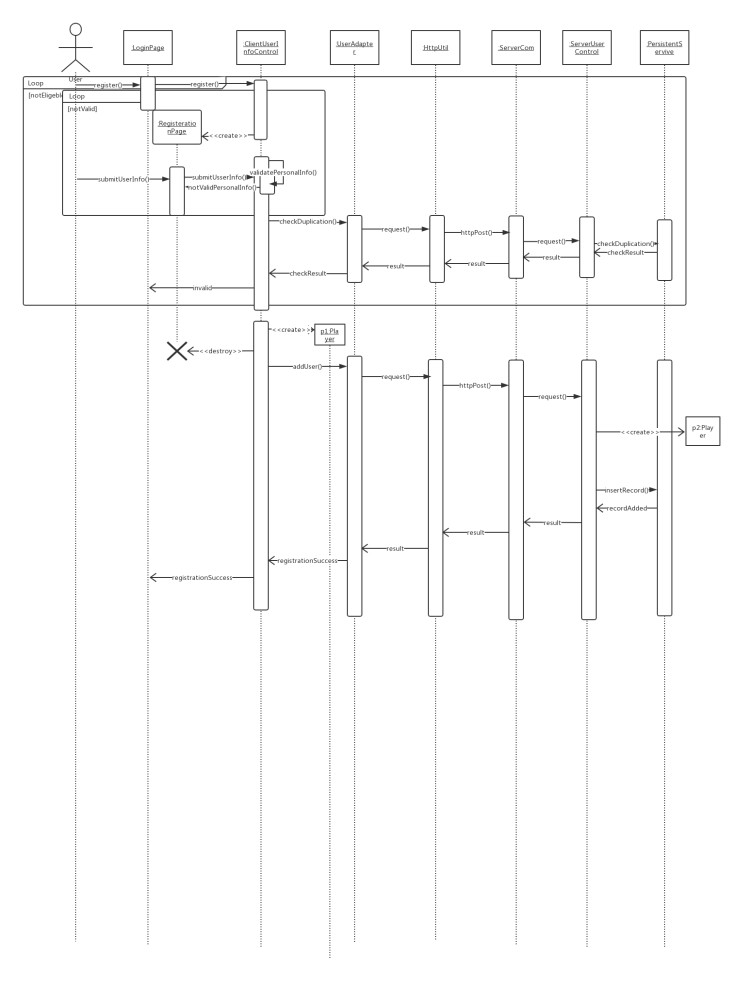
1. <Use-Case 1>实现

采用交互图的形式完整定义该Use-Case的实现过程。一个Use-Case中所有系统操作都必须得到相应的设计。如果一个用例比较复杂，可以分几个交互图进行表达。

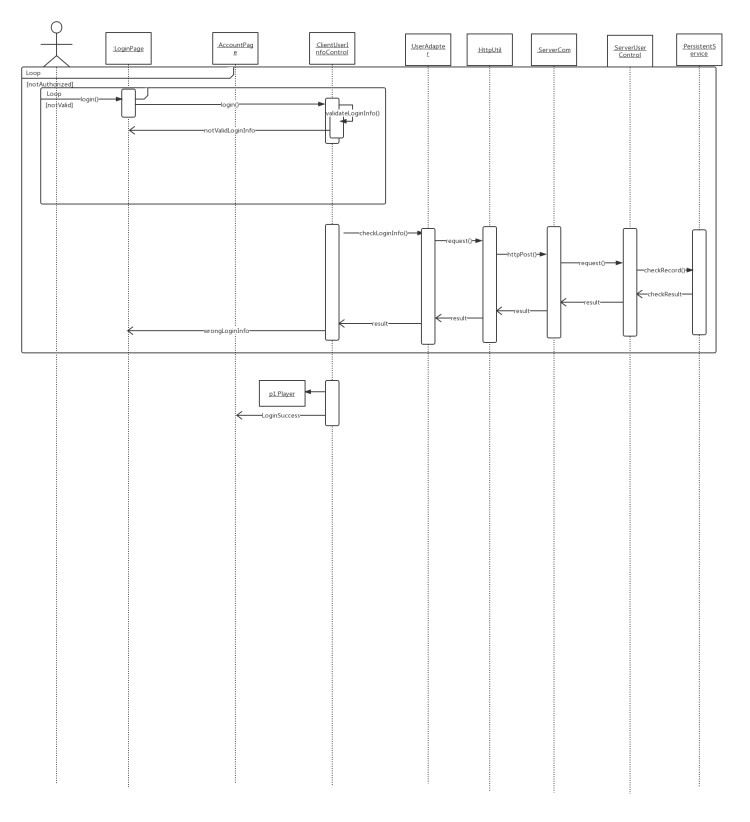
1. <Use-Case 2>实现

……

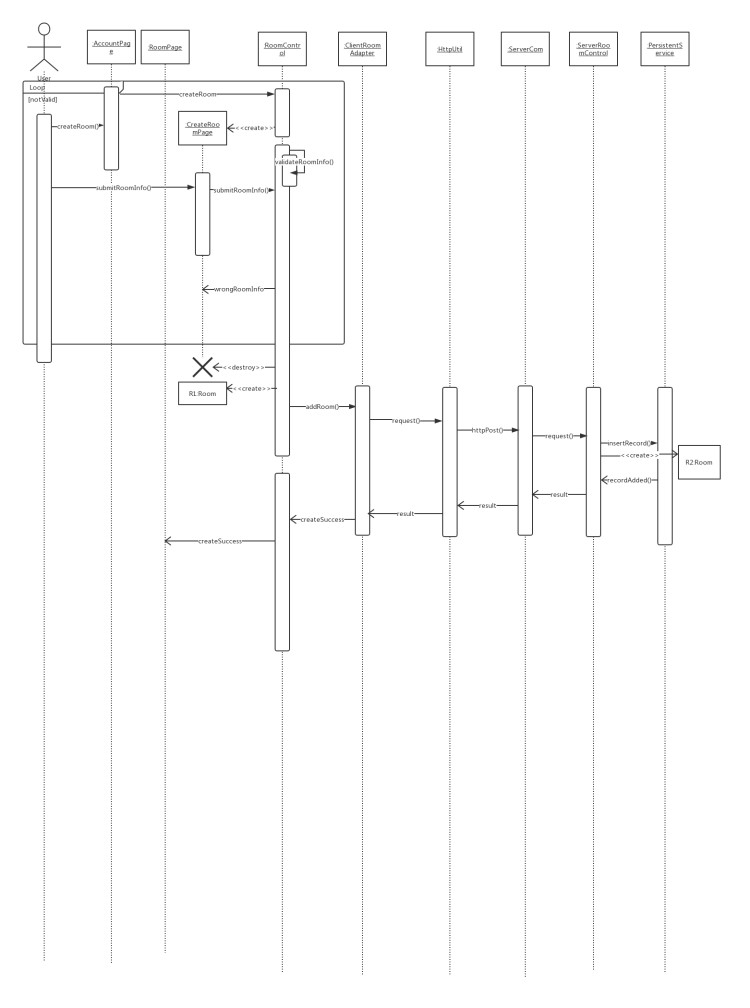
1. <Register> Implementation



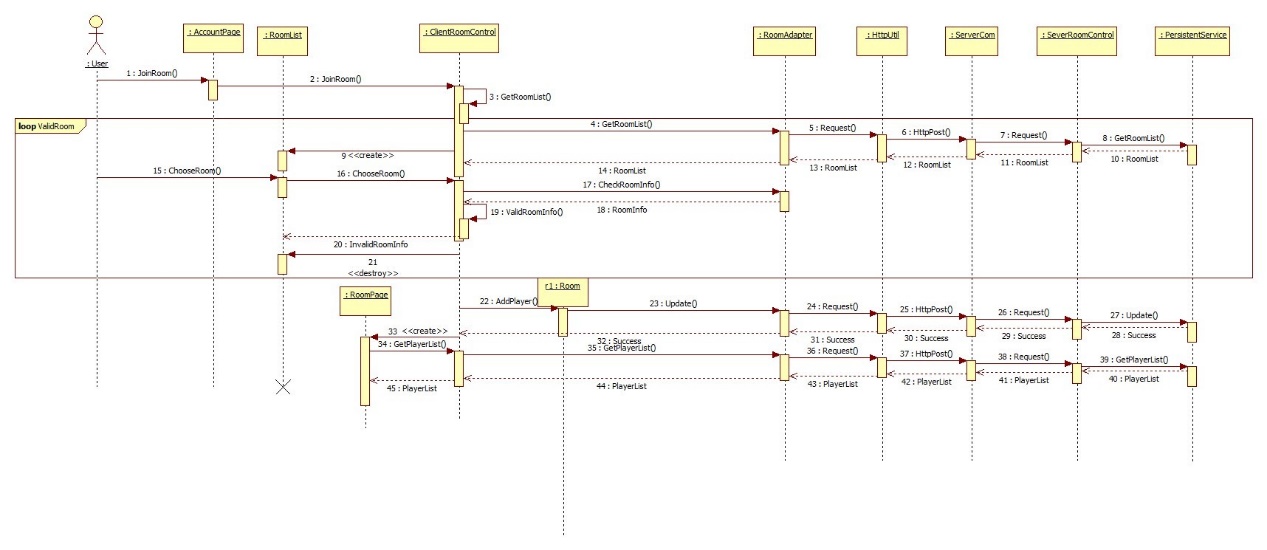
1. <Login> Implementation



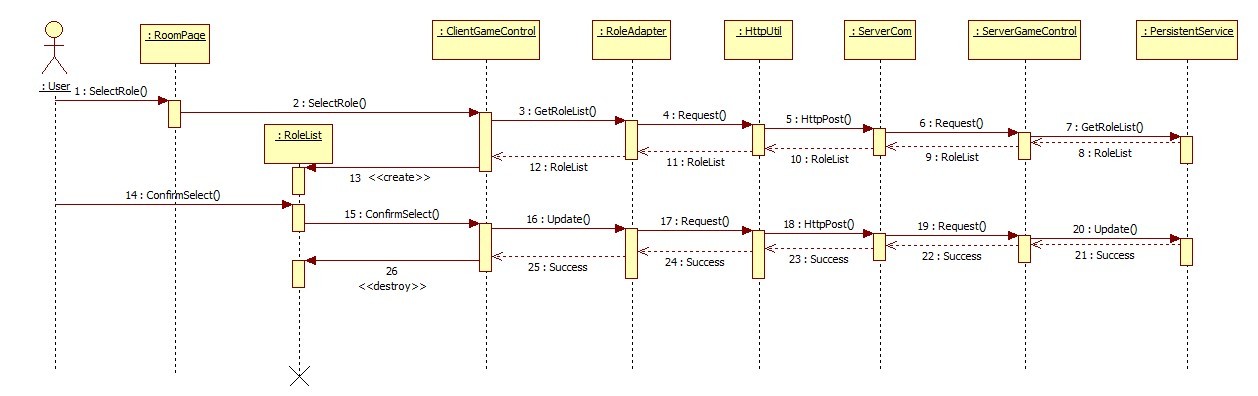
(3) <CreateRoom> Implementation



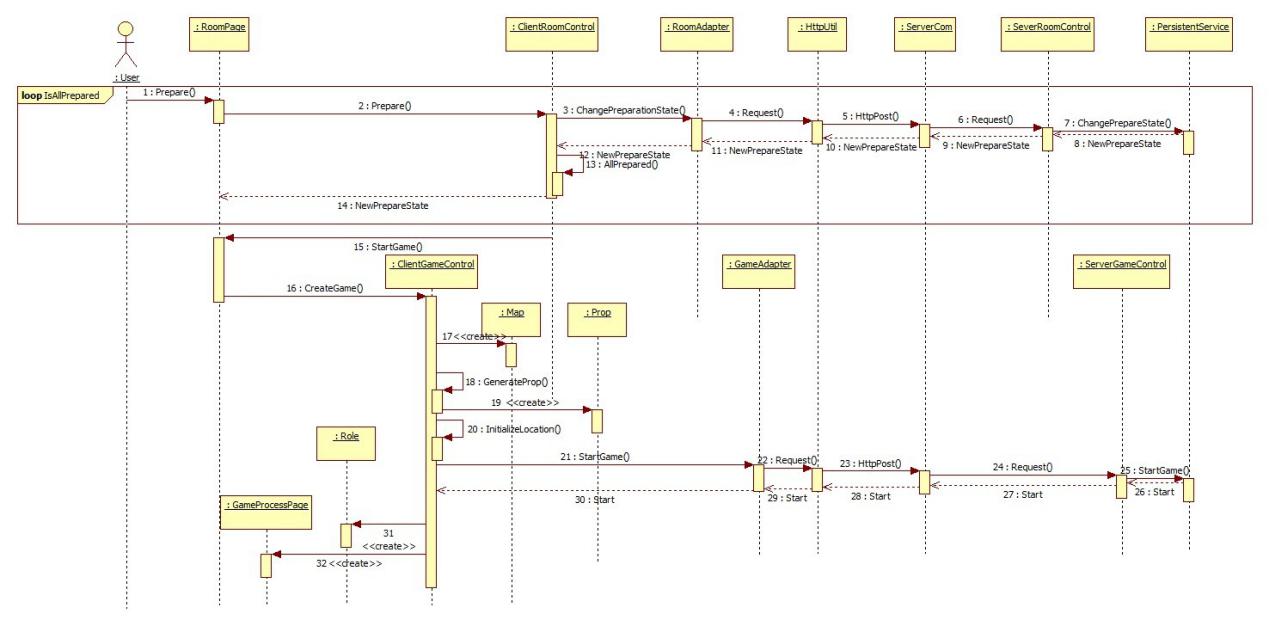
(4) <JoinRoom> Implementation



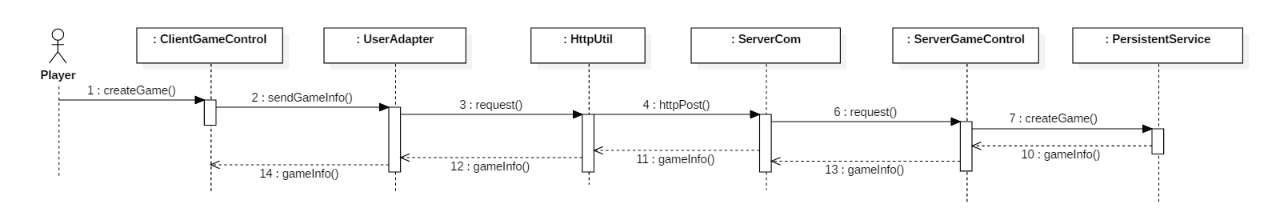
(5) <SelectRole> Implementation



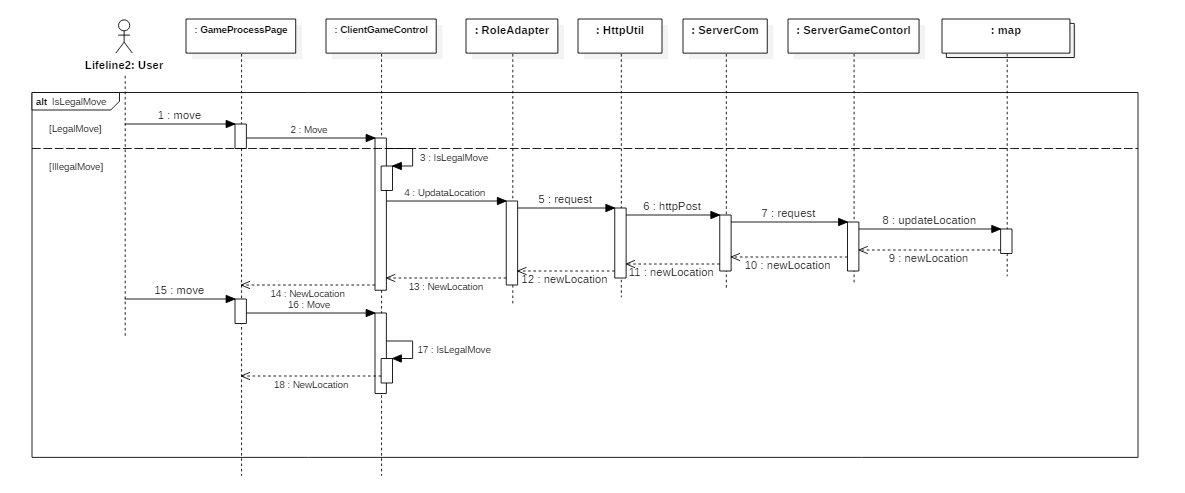
(6) <Prepare> Implementation



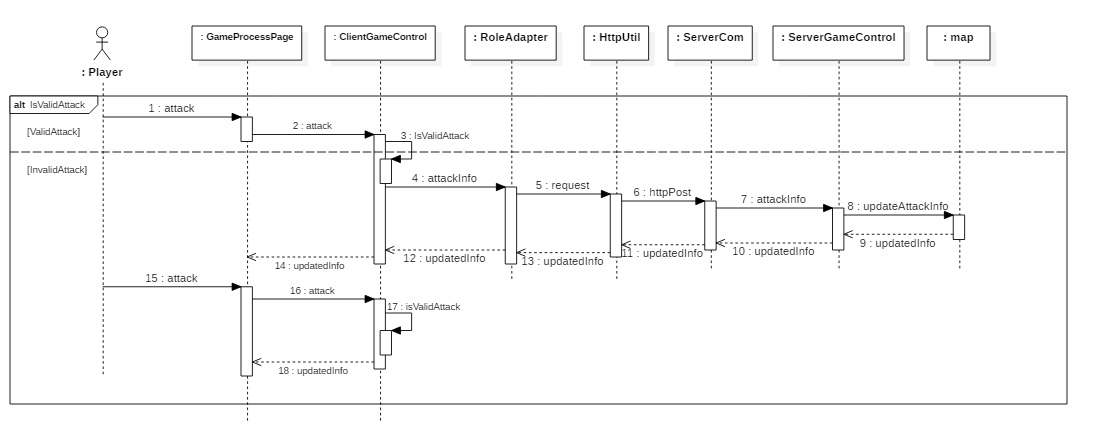
(7) <GameProcess> Implementation



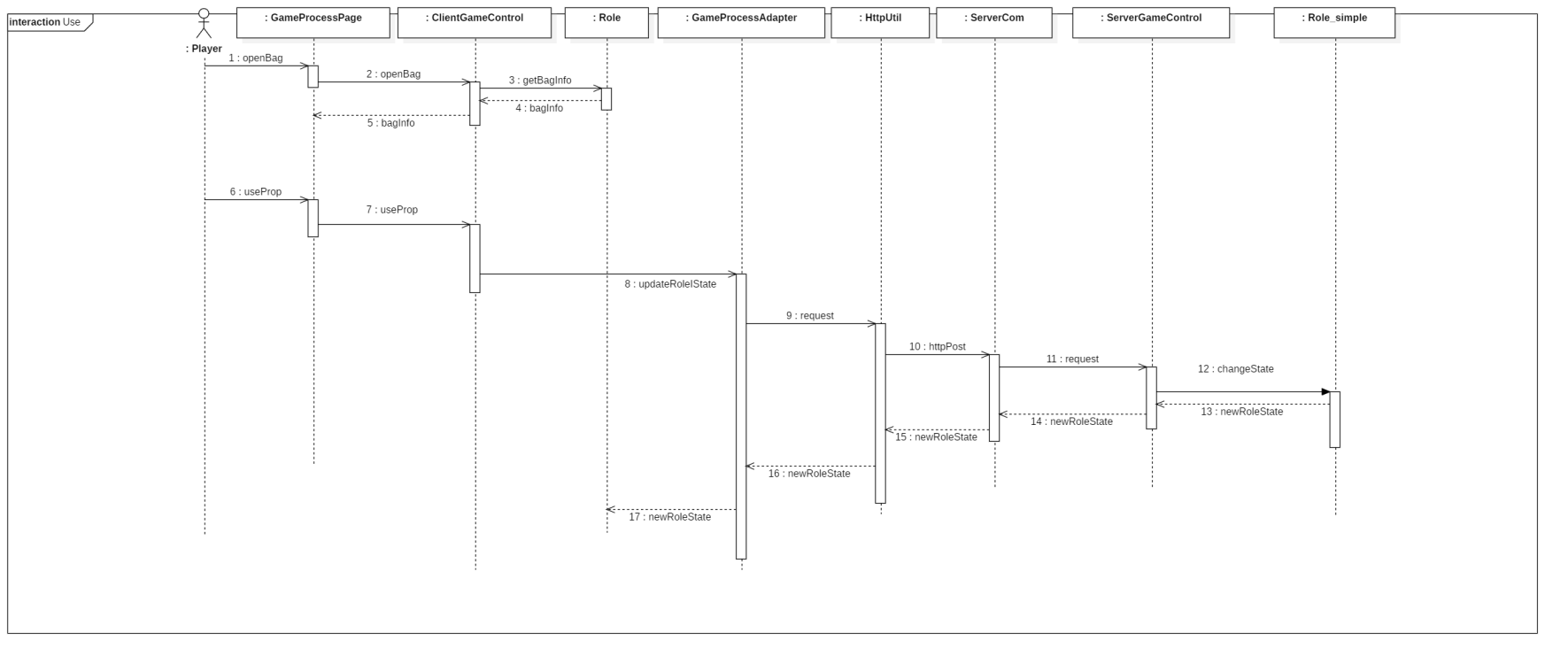
(8) <Move> Implementation



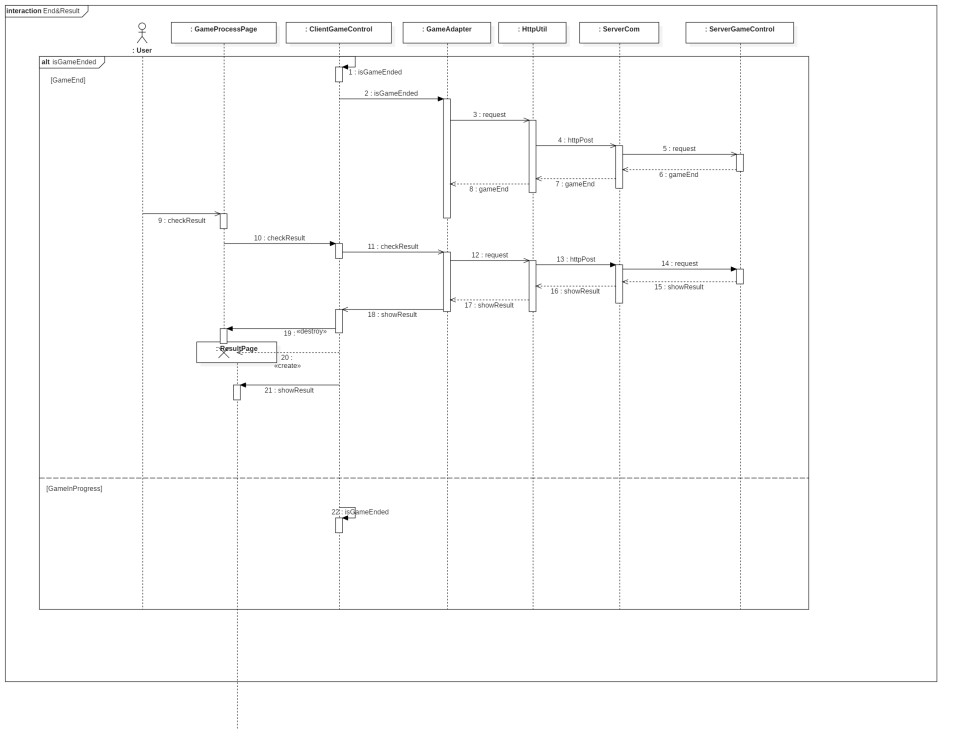
(9) <Attack> Implementation



(10) <Use> Implementation



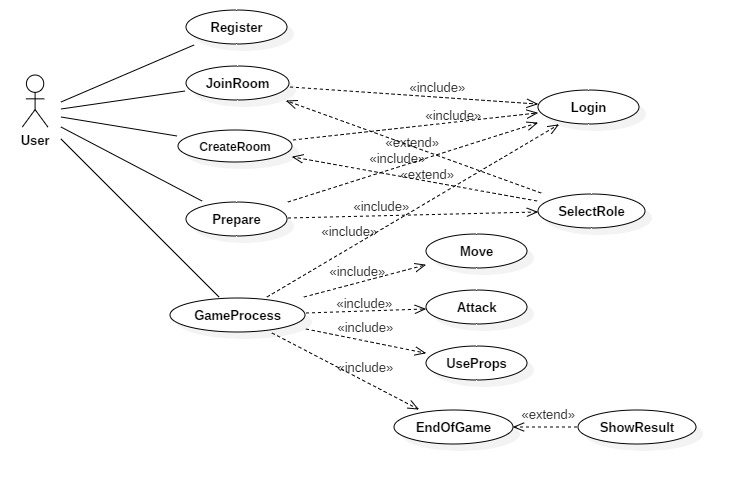
(11) <End&Result> Implementation



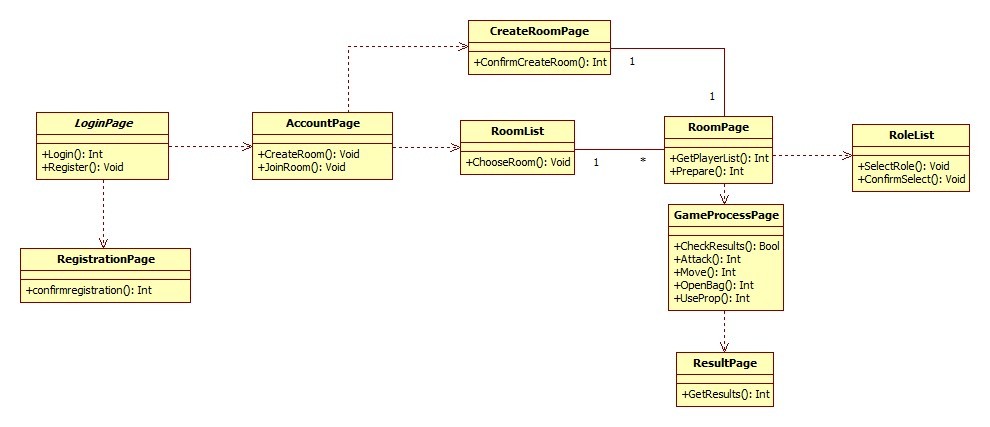
## 3.3 设计视图Design View

将Use-Case实现中涉及到的所有设计类以及它们的关联关系，画在一张设计类图中。如果类比较多，我们可以先画一个包图，然后画出各个包中包含的类图。

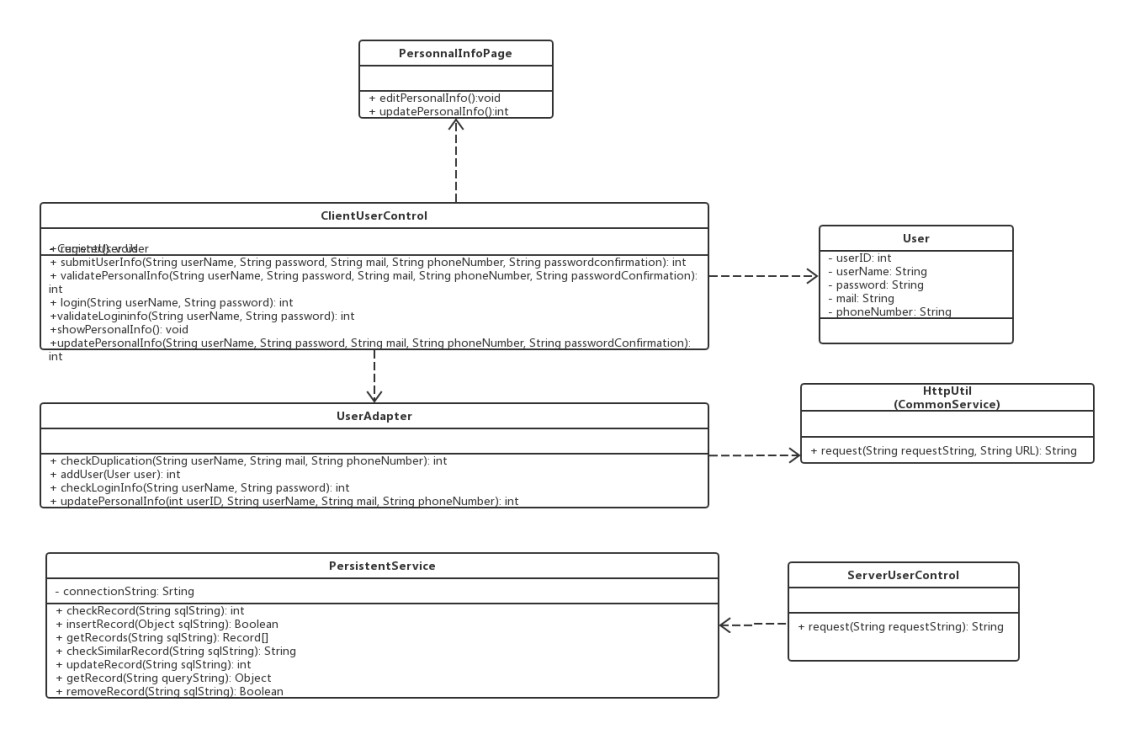
Since there are lots of classes, we decided to create a Package Diagram and then draw Class Diagrams on the basis of subsystems.



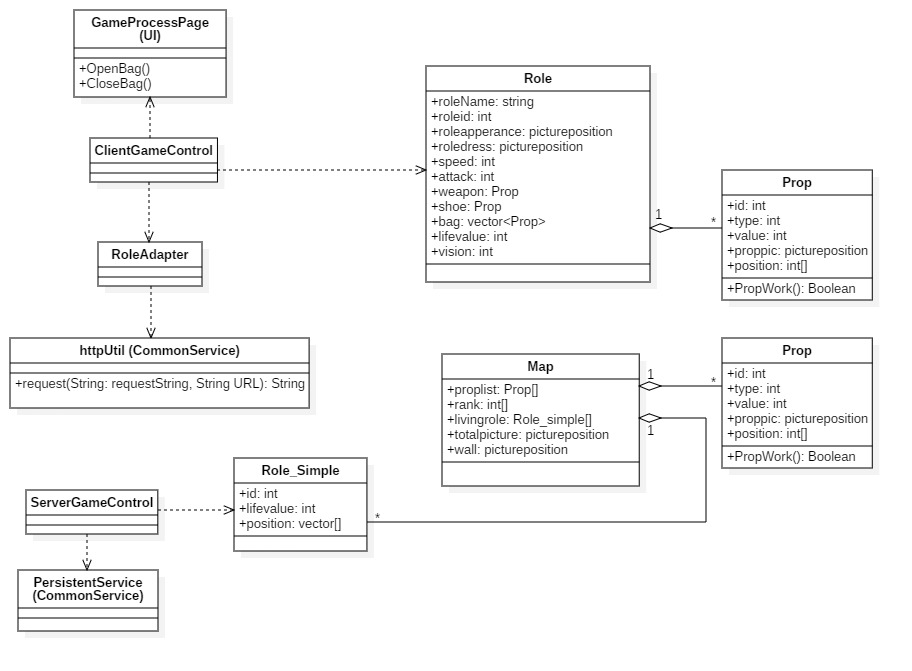
1. User-Interface Subsystem



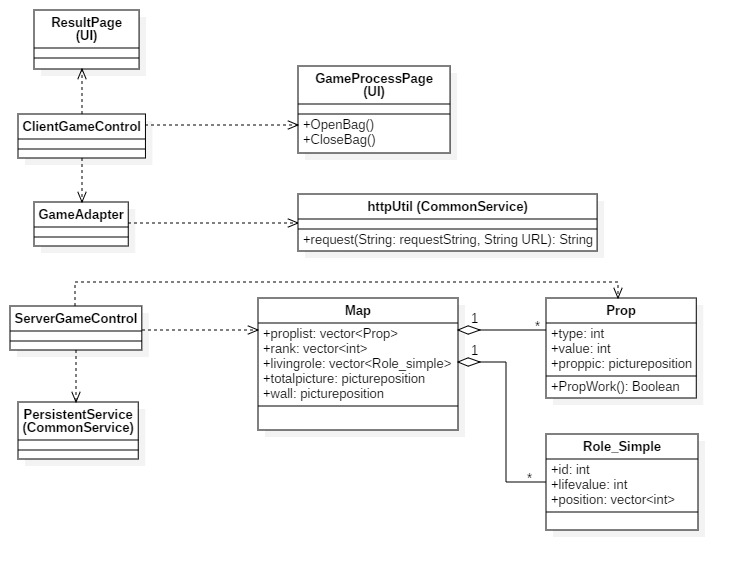
1. User-management Subsystem



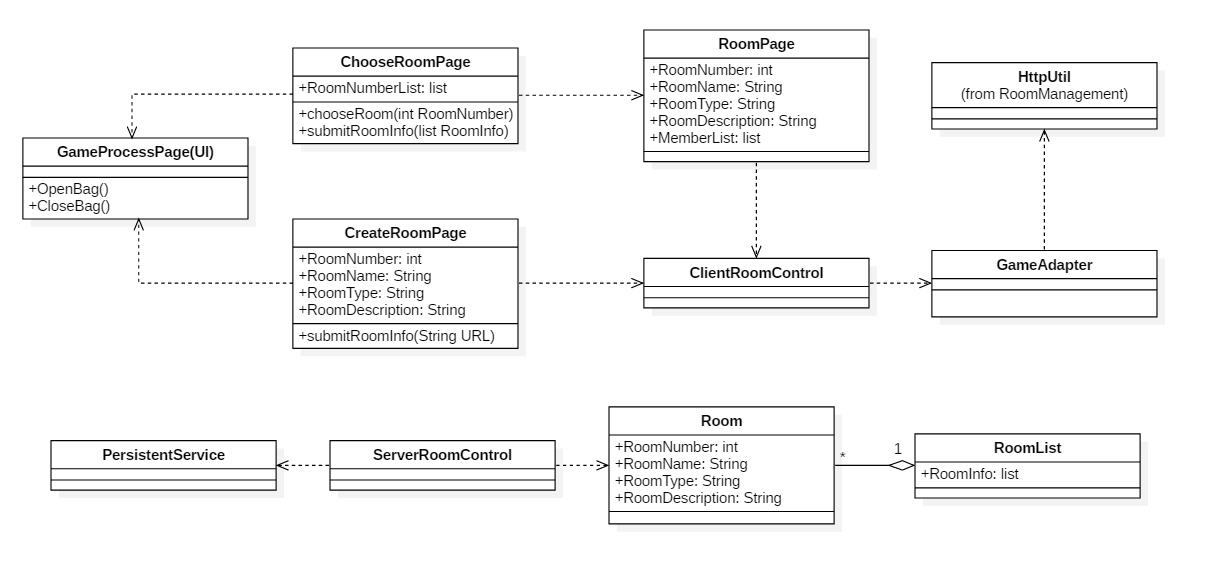
1. Role-management Subsystem



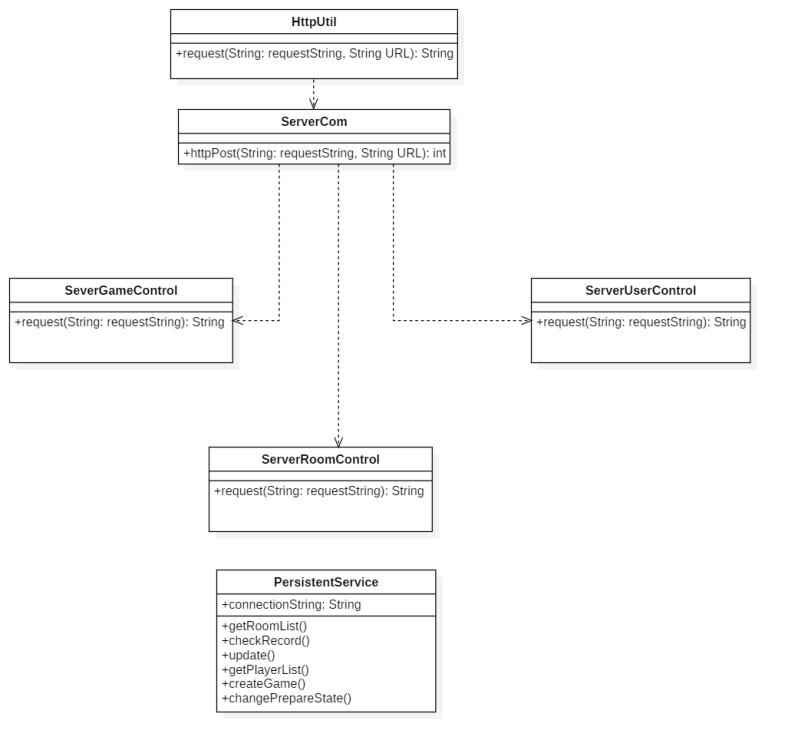
1. Map-management Subsystem



1. Room Management Subsystem



1. Common-service Subsystem



## 3.4 其他图(NULL)

如果需要，针对某一个类给出其状态图。

如果需要，针对某一流程构造活动图。

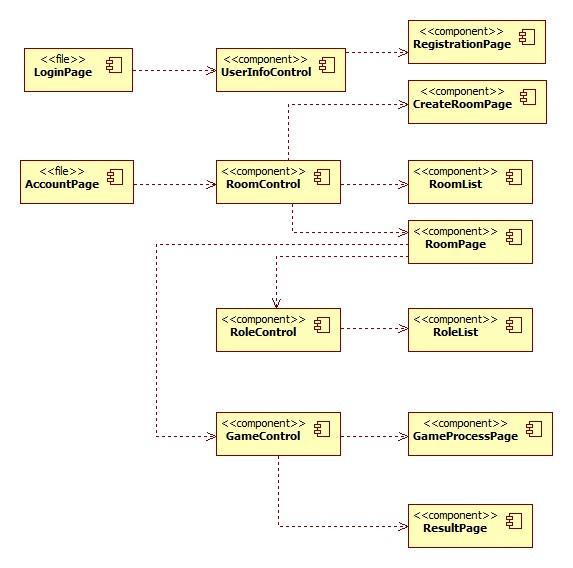
# 4. 实现视图Implementation View

针对每一个子系统，画出其对应的组件图。需要画两种组件图：

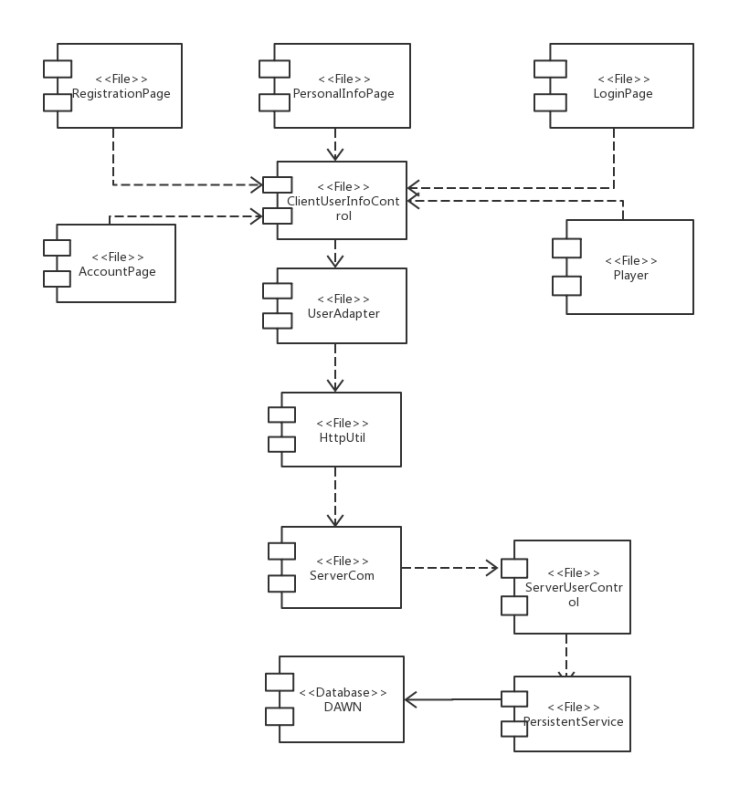
1. 一种为开发环境中的组件构成及依赖关系。开发环境中的组件指的是程序源文件以及它所依赖的其他文件。
2. 另一种为编译后生成的组件及其依赖关系。

组件内部可以画出它包含的类以及类的关系。

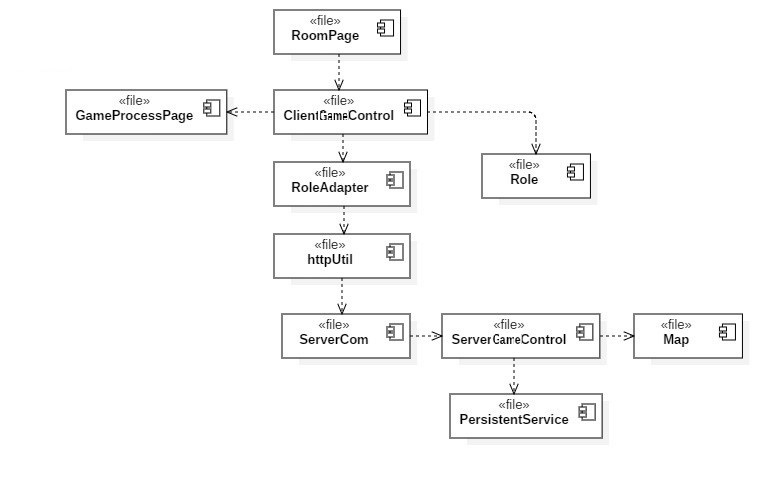
1. Development Model Composition
   1. User-Interface Subsystem



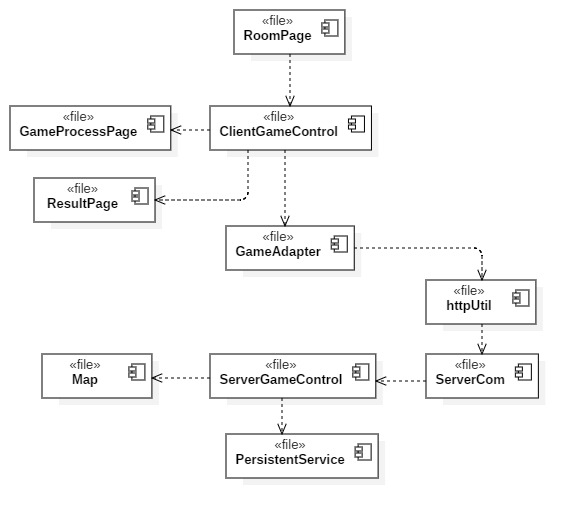
* 1. User-management Subsystem



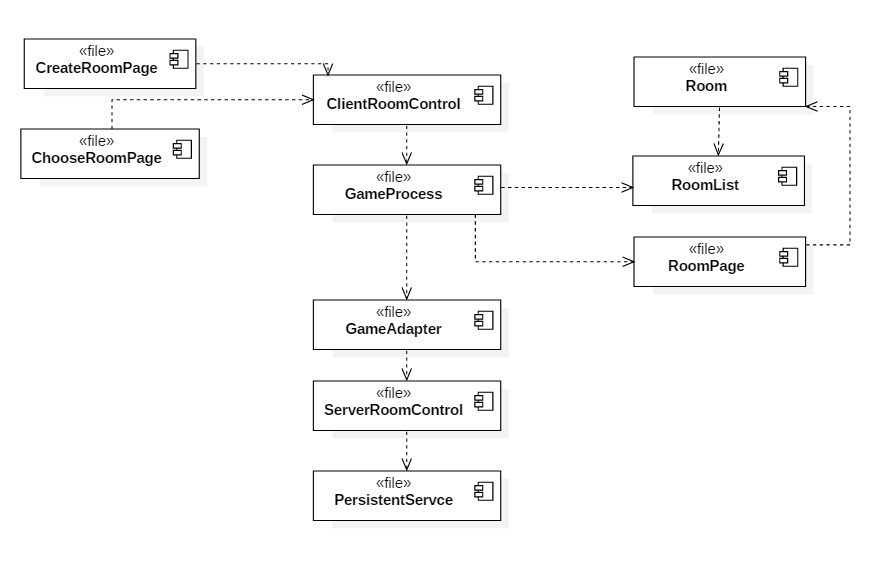
* 1. Role-management Subsystem



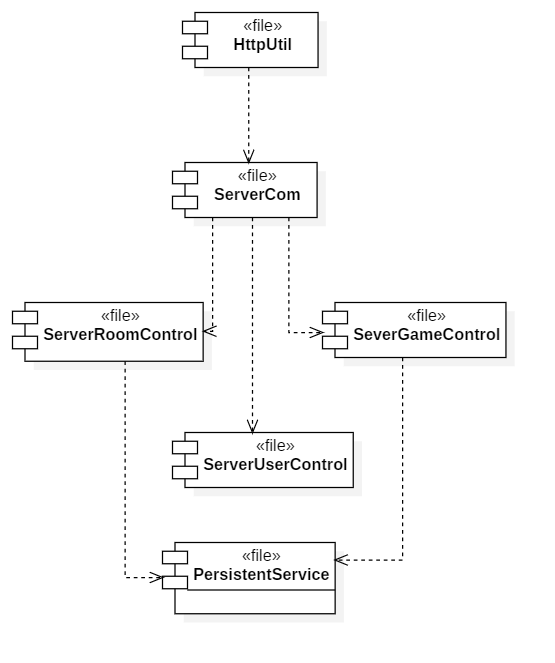
* 1. Map-management Subsystem



* 1. Room-management Subsystem



* 1. Common-service Subsystem



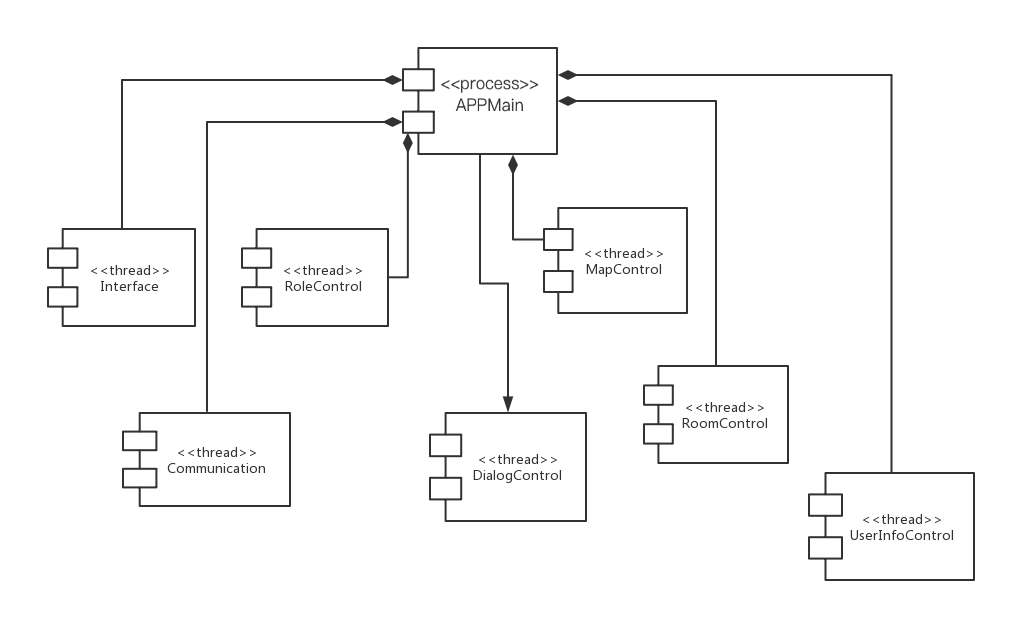
1. Post-compilation

Since each one of the .java file will be compiled to generate a corresponding .class file, the compiled component map should be identical to the development model composition.

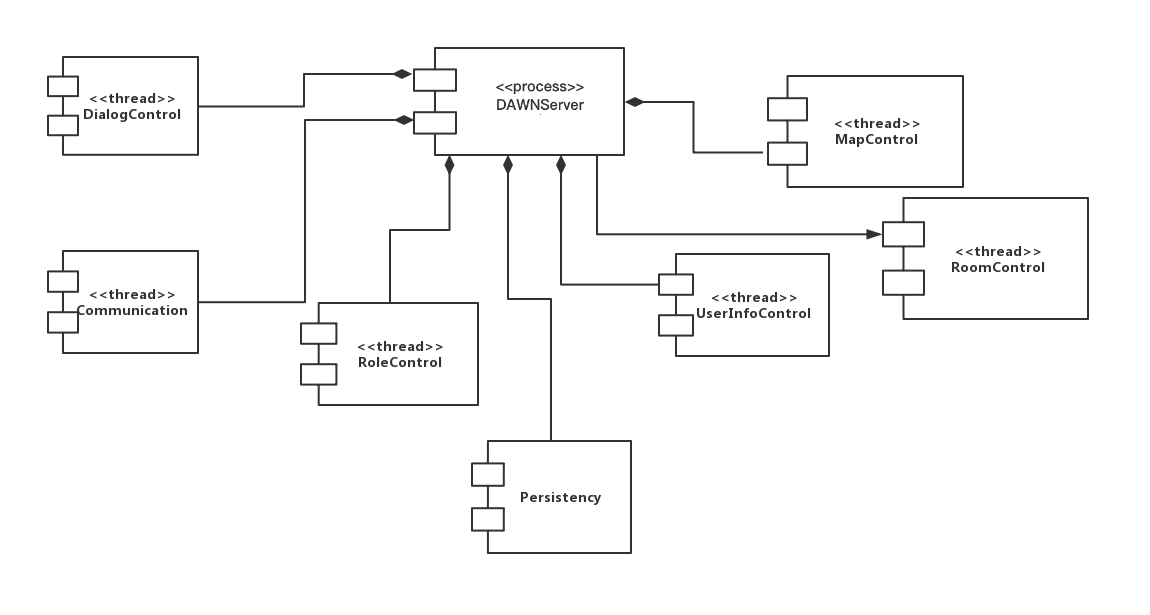
# 5. 进程视图 Process View

通过类图或者组件图的形式表示进程类、线程类及其相互关系，通过类图表示进程、线程中包含的类。

1. Client Process View



1. Server Process View



# 6. 部署视图 Deployment View

画出系统的物理部署图。

