ZJU OOP Project 2 Lab Report

November 29, 2019

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Chen Yi Hui | Major: | Computer Science |
| Student id: | 3180300684 | Course: | Object Oriented Programming |
| Email: | 3180300684@zju.edu.cn | Instructor: | Weng, Kai |
| Start date: | November 4th, 2019 | Due Date: | December 7th ,2019 |

1. Project Topic:
   1. Project Name.

Game of Castle.

* 1. Project Description.

To design a CLI game. The castle in the game should have many levels and many rooms. There should be two NPCs(Non-player character) in the game, they are Monster and Princess. Two NPCs should be in separate rooms, the rooms NPCs in should be randomly selected. The two rooms with NPC should not predictable. Rooms should have different exits to other places. Player must explore rooms in the castle, find Princess and rescue her from the castle (take her to the lobby where the game starts). If player encounters Monster, the game is over.

When game starts, and when player enters a room, there should be message about the room player enters and instructions for player like below:

“Welcome to the lobby. There are 3 exits as: east, west and up.”

“Enter your command:”

The player then can input ”go” followed by an exit name. The program should display a message once the player finds any NPC, and the only way to leave castle is via the lobby.

All printed messages and user input are in English to simplify the code.

* 1. Project Requirement:
* At least three different kinds of room;
* At least five rooms;
* The room with monster or princess is randomly set.

1. Project Design:
   1. Program description.

The program includes Character, Plane, Floor, and Room class.

Character class is for NPC objects.

Plane class is used as an abstract two-dimensional plane for Floor class and Room class to inherit from.

Floor class is inherited from Plane, for floor level objects.

Room class is inherited from Floor, for room objects.

In the main.cpp, program initializes Characters, Rooms and Floors. After that, program connects Room objects and Floor objects according to floor plans, then selects two separate Rooms to contain (pointer to Character) two NPCs. NPCs are “Princess” and “Monster”. Game will begin and player explore Rooms and Floors as a Floor pointer. Every time player enters a room, program will display info about the room, next instructions and room exits. Player then input where to go, program will bring player to the next Room or Floor according to player’s valid input. When player finds NPC Princess, program displays a message alerting player that win condition is acquired. From then, player will bring along Princess to everywhere player goes, player wins game by bringing “Princess” to Lobby (start Room). Once player finds the room with “Monster” pointer, program ends game by putting Monster in Lobby, and player will point to Lobby next. After game ends, program examines which Character the Lobby points to. If Lobby points to Princess, display a message that player wins the game. If Lobby points to Monster, display a message that player loses the game. If Lobby points to nothing, game must be ended by user, message displays about user interruption.

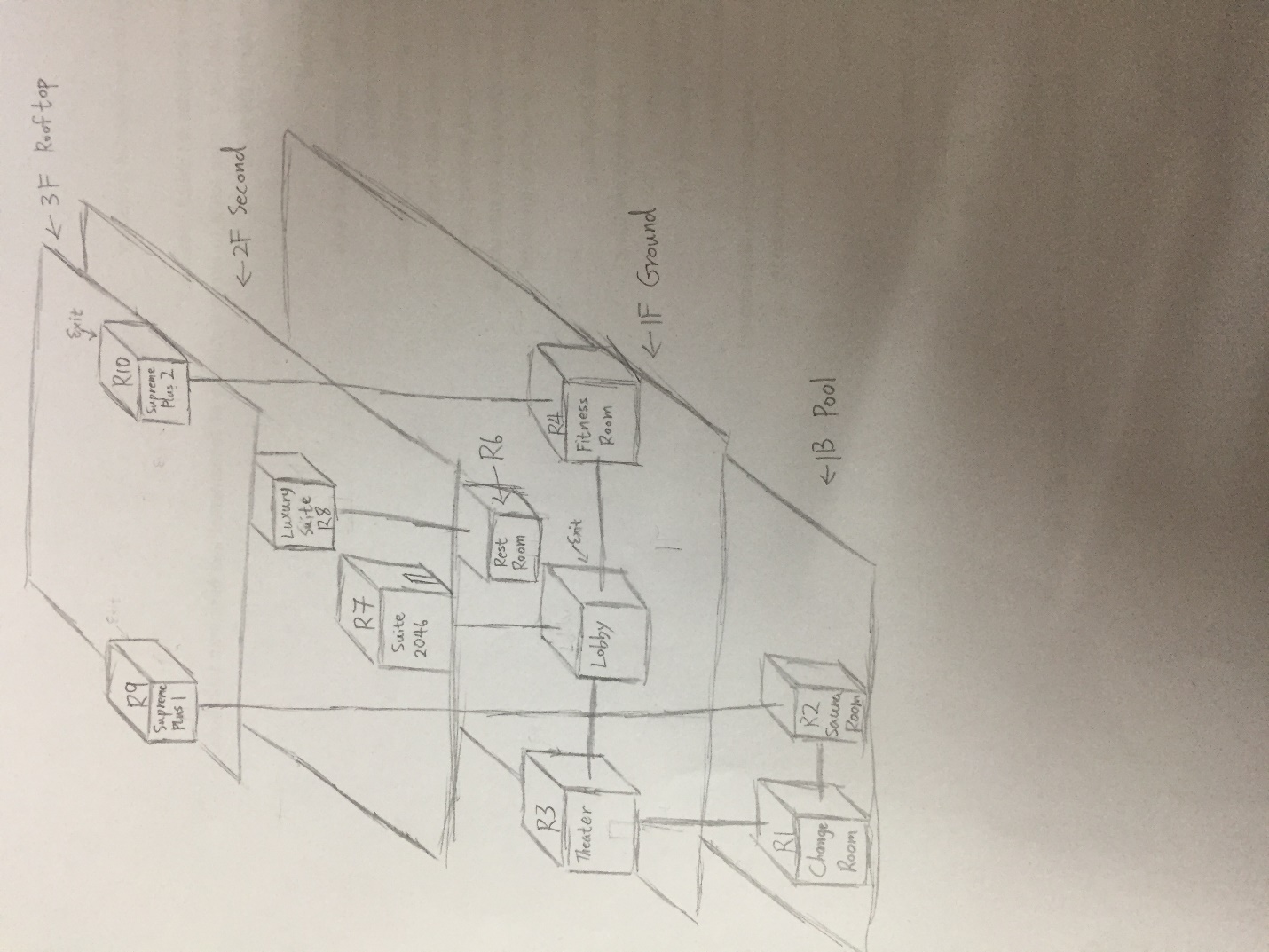


Figure 1 Castle map

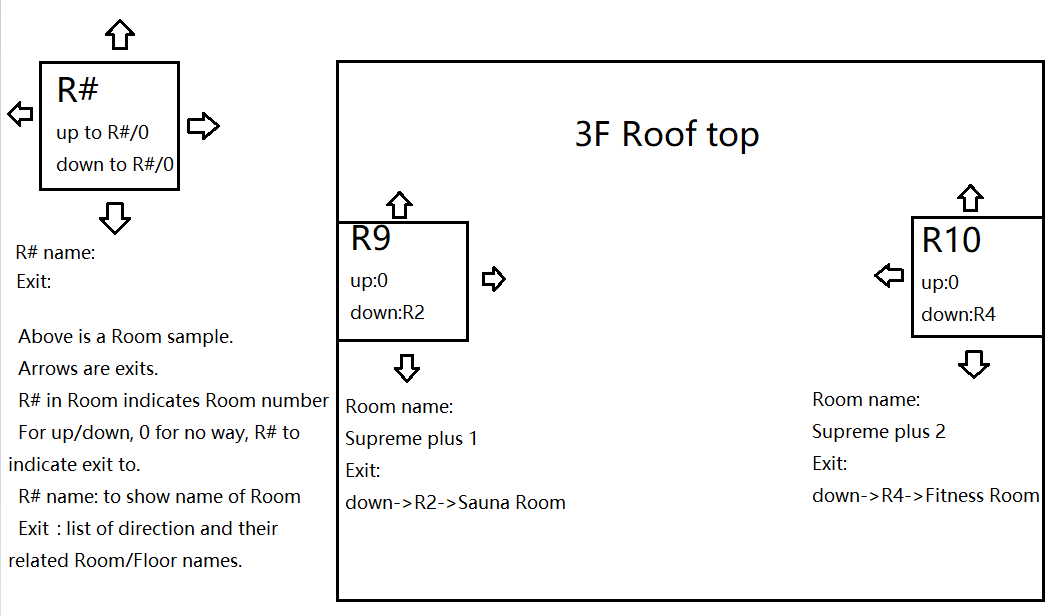


Figure 2 3F floor plan. Big rectangle represents Floor object.

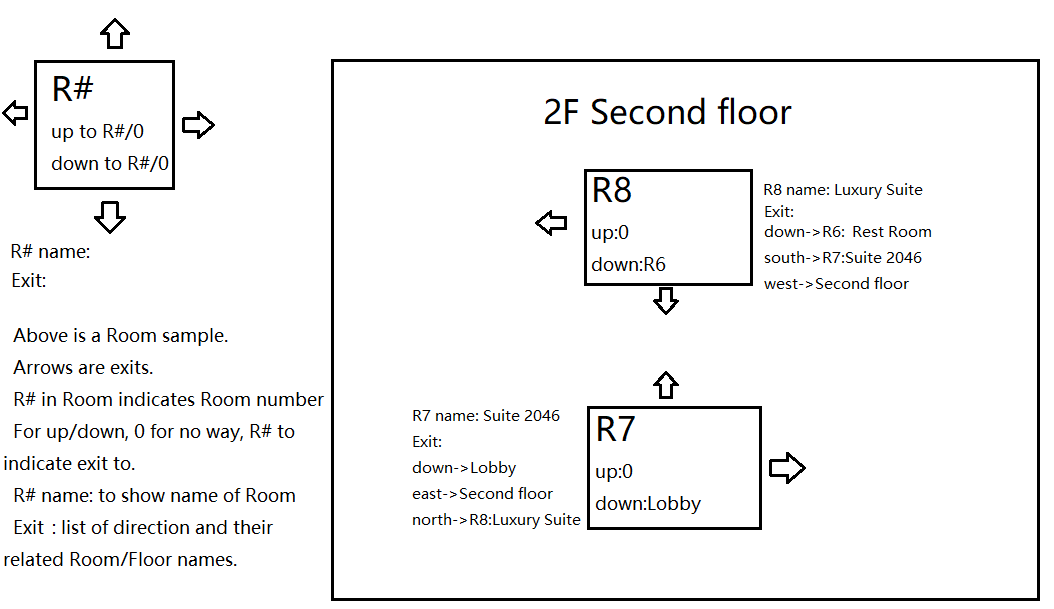


Figure 3 2F floor plan.

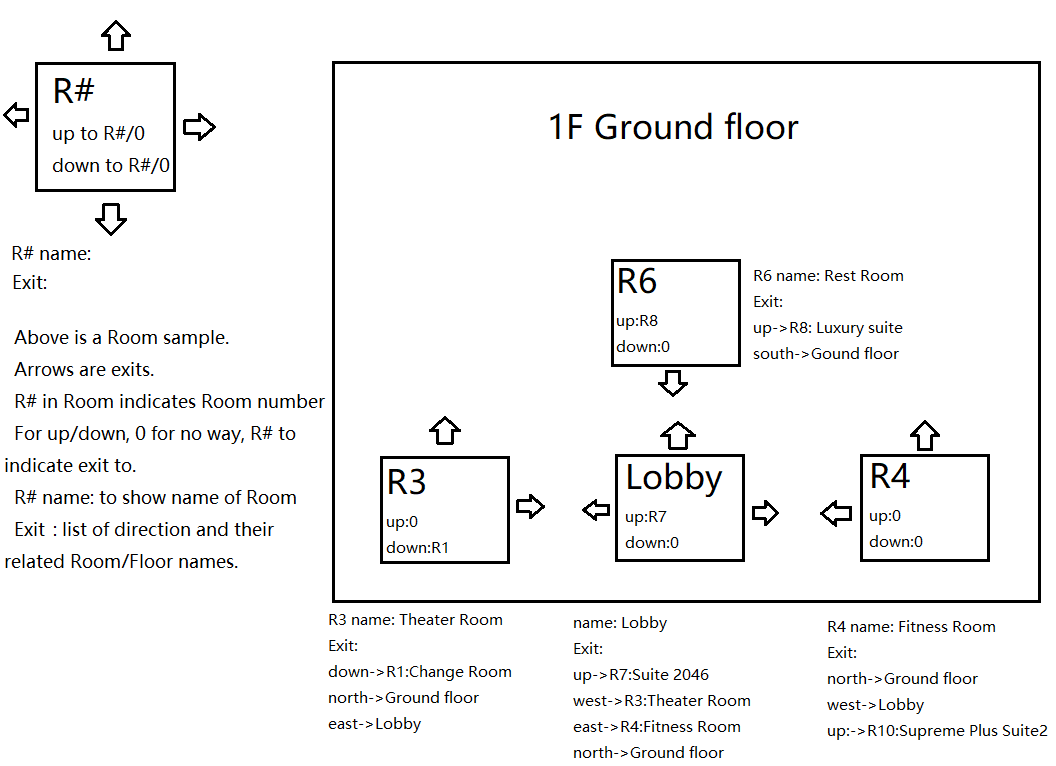


Figure 4 1F floor plan

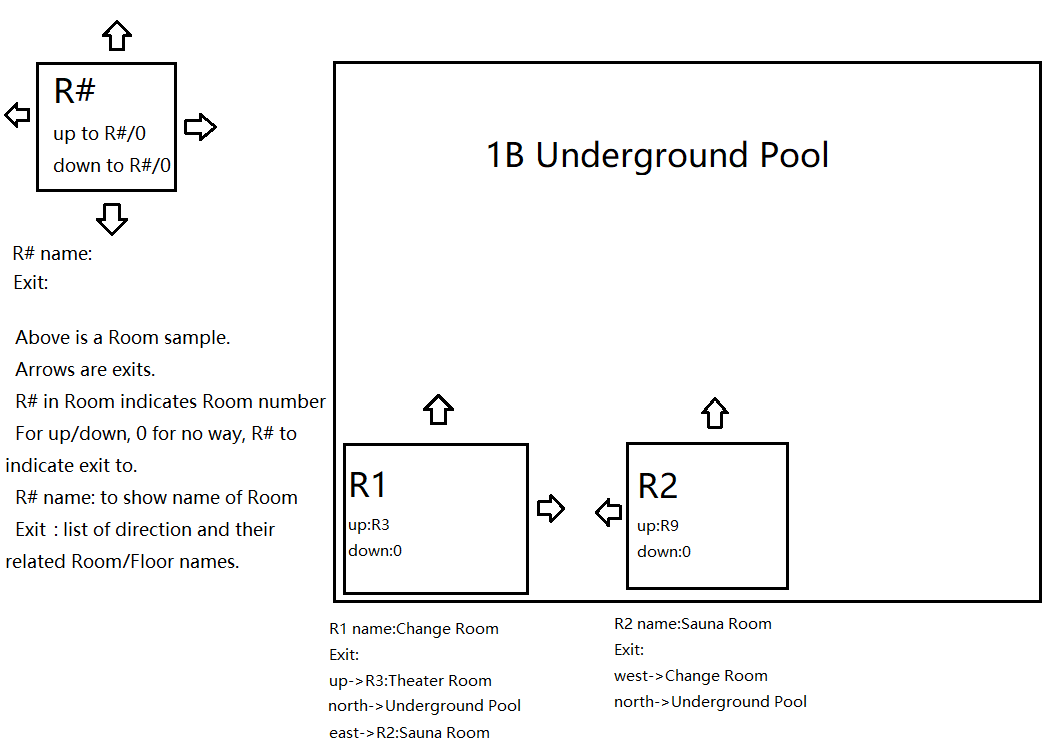


Figure 5 1B floor plan

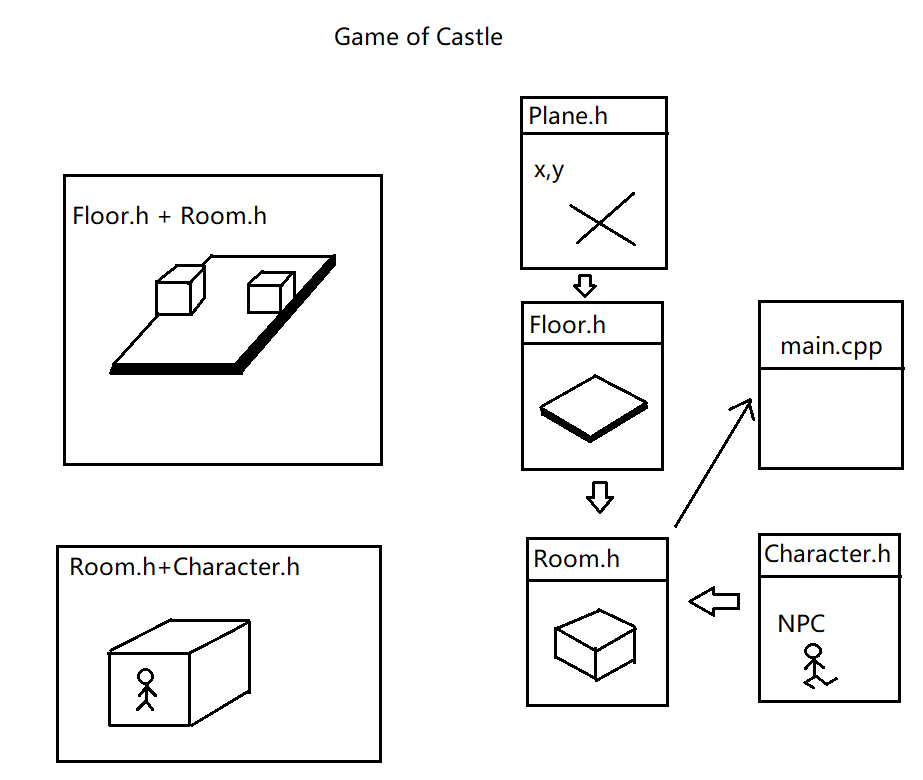


Figure 6 Abstract castle

* 1. Program data structures
* Character class

Contains private members: Name, Description, and Value. Name is the name for the Character object. Description is a string that stores information about the Character. Value is an integer that can change the game state.

Simple accessing and value setting functions are provided.

* Plane class

“Template” for Floor class and Room class. Plane class has private members: Name, Level, and state. String Name for object name, string Level to show which level object belongs, and static in state to tell game status.

Simple accessing and value setting functions are provided.

* Floor class

Floor class has protected member rooms and NPC, which allows its child class to access. “rooms” map maps strings and pointer to Floor objects, and main purpose of the map is to map Room objects by up casting, so Room objects can be connected by mapping. As well as mapping all the Room objects on a Floor object according floor plans.

Operational functions on Floor objects and Plane Objects are declared and defined here. Functions to check game state by accessing base class private static variable. Function to move NPC from one place to another. And function to make player (Floor object pointer) to go from one place to another.

* Room class

This class contains an extra private member and inherit from Floor class. Private member Lobby to indicate specific Room object be the start point of game. Purpose of this class is to provide clarity, to differ Room objects from Floor objects for later additional game development.

This class adds extra operation to bi-connect Room objects, making connections between Room objects. Connection is done with Floor class protected map member. Room class also overrides operation from Floor class, overrides function to make player go from one place to another, this provides mobility of the player and connectivity of the castle.

* 1. Problems encountered
     1. Problem connecting Room objects and Floor objects together.

Used map to map string and pointer to Floor, string to identify Floors.

* + 1. Problem compiling program with multi-header files.

Use command to run main.cpp and all related .cpp files. Then out main.

* + 1. Problem “building” castle from abstract.

Make Plane class to be a base for “building” structures on top of. “Build” Floor class from Plane, then “build” Room class on Floor.

* + 1. Problem “moving” in castle.

Set player as a Floor pointer, player can visit Room objects and Floor objects by pointing at Floor objects, upcasting involved.

* + 1. Problem of how to end the game.

Set a static variable in Plane class, change the static variable once player enter rooms with Monster. Check static variable every time before allowing player to input.

1. Discussion and Revision:

In this project I have learnt a lot about upcasting. Upcasting on objects can make objects be treated the same as base class objects, operations on a specific related class objects are selected according to specific class by providing virtual functions. For example, when player visit places in the castle, even though Floor objects and Room objects are two classes, player can move freely between the two classes by upcasting Room objects with virtual functions.