OOP 大作业——酒店客房管理系统

赵龙飞-18030100257

目录

1 题目分析

1.1 题目描述

题目要求实现一个酒店客房管理系统。

客房　酒店客房可以分为三个等级：豪华、标准和普通，并且不同等级的客房床位数和价钱数也不同。

客房等级	床位数	价格
豪华	4	400
标准	2	200
普通	1	100

顾客　顾客分为金卡、银卡、普通和非会员，他们享有不同的折扣优惠。

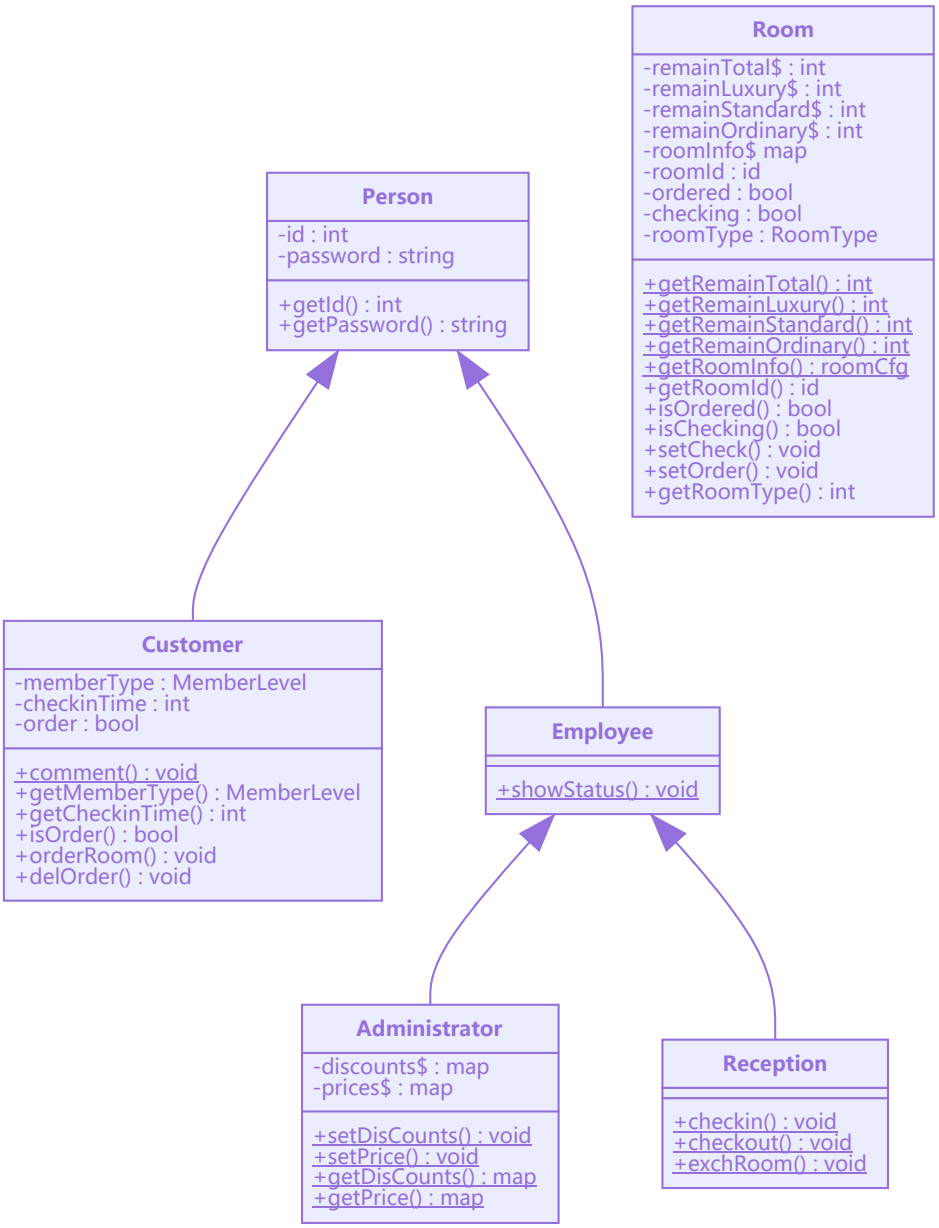
会员类型	折扣
金卡	8 折
银卡	85 折
普通	9 折
非会员	不打折

1.2 需求分析

1. 管理员：登陆系统，查询当前客房入住和预定情况，并且可以设置客房的价格和顾客的优惠政策。
2. 前台：登陆系统，查询当前客房入住和预订情况，可为顾客办理入住、退房和换房服务，在顾客退房时提供收银服务。
3. 顾客：当第一次登陆系统时，顾客需要注册账号，登陆系统后，顾客可根据自己的需要预订客房，并且预订之后，入住时间一天前可以取消预订；当顾客退房后，顾客可发表评论。

2 类图设计及说明

2.1 类图设计



注：属性成员名后面加'$' 表示该成员为静态成员，函数成员加下划线说明此函数成员为静态函数成员。

2.2 说明

1. 本系统中共涉及三类需要登陆系统进行不同操作的人，他们都可以登录系统，而这三类人 (Person) 可以分为两大类：酒店职员 (Employee) 和顾客 (Customer)。而酒店职员可以细分为前台 (Perception) 和管理员 (Administrator)。

   (a) 因此，首先定义 Person 类，Person 类有账号 (id) 和密码 (password) 属性以及对应的 getId() 和 getPassword() 方法。

   (b) Person 类派生出 Customer 和 Employee 两个子类。

      i. Customer 类有成员类型 (memberType)、入住时间 (checkinTime) 及是否预订了房间标志 (order) 属性以及对应的 getMemberType()、getCheckinTime() 和 isOrder() 方法。除此之外还有用于预定房间 orderRoom() 方法和用于取消预订的 delRoom() 方法。

      ii. Employee 类中定义有一个查询当前客房入住和预定情况的函数成员 showStatus()，由于此方法全体管理员和前台都适用，故将此函数设置为静态函数成员。

   (c) Employee 类派生出 Administrator 子类和 Perception 子类。

      i. Administrator 类中定义有静态数据成员折扣信息 (discounts) 和客房价格 (prices)，以及对应的静态函数成员 getDiscounts() 和 getPrices()，除此之外还设有静态函数成员 setDiscounts() 和 setPrices() 分别用来实现设置优惠政策和设置客房价格的操作。

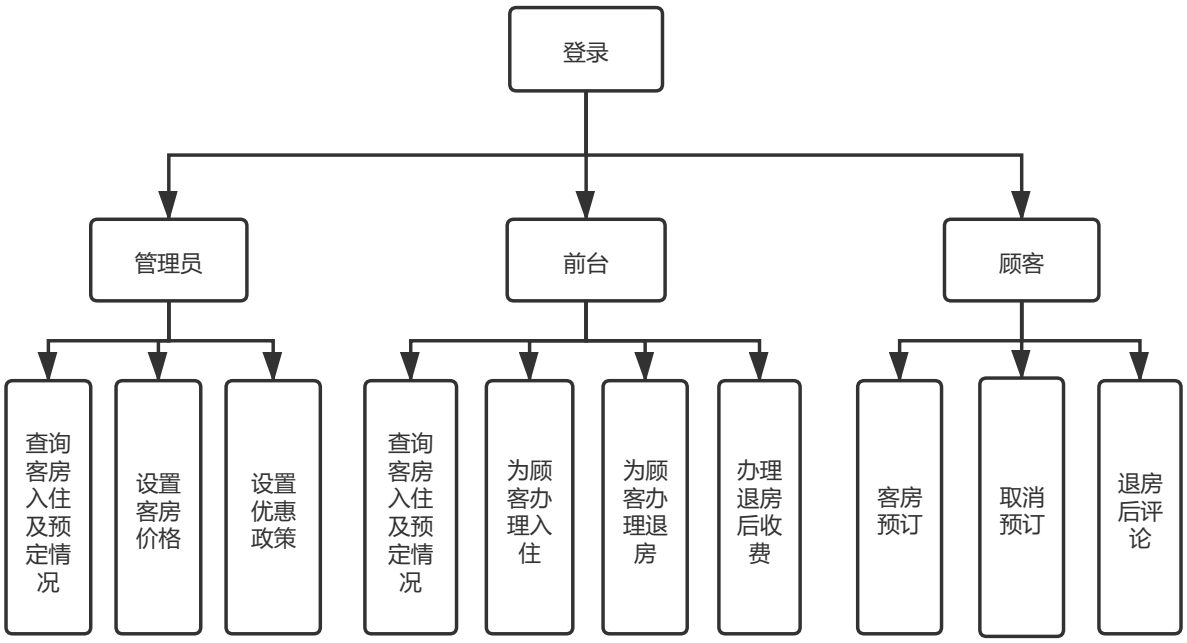
      ii. Perception 类中定义有三个静态函数成员，checkIn() 实现了前台对顾客提供入住服务的操作，checkout() 实现了前台为顾客办理退房服务的操作，exchRoom() 实现了前台为顾客办理换房服务的操作。

2. 此系统中的客房抽象为 Room 类：

   (a) 它定义有五个静态属性成员剩余房间总量 (remainTotal)、剩余豪华房数量 (remainLuxury)、剩余标准房数量 (remainStandard)、剩余普通房数量 (remainOrdinaty) 以及用来存储房间信息 (床位数)；与之对应设有 getRemainTotal()、getRemainStandard()、getRemainOrdinary() 等静态函数成员。

(b) Room 类中有房间号 (roomId)、房间类型 (roomType)、当前此房间是否被预定标志 (ordered) 和当前此房间是否被入住 (checking) 标志，与之对应的有 getRoomId()、getRoomType()、isOrdered()、isChecking()、setOrder() 和 setCheck() 等函数成员。

3 系统功能设计



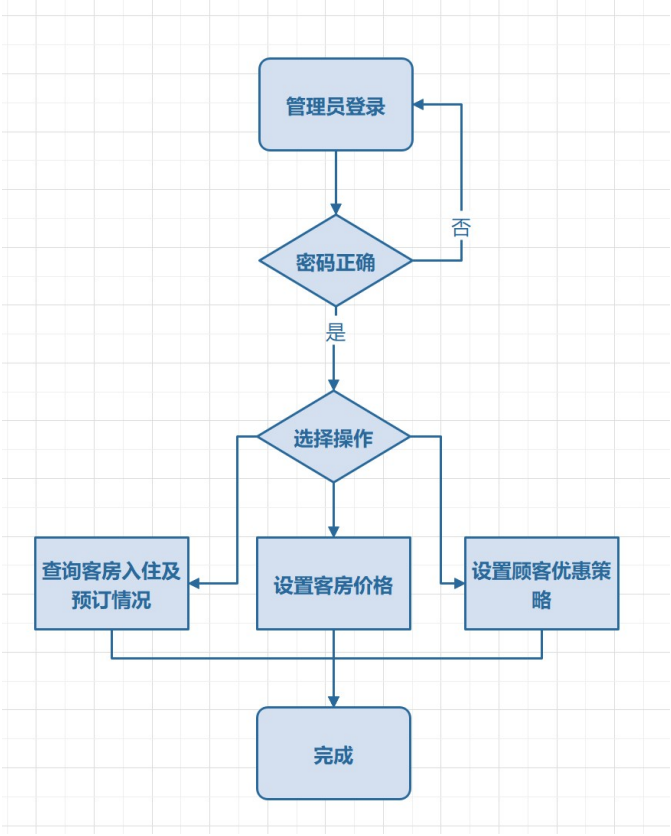
1. 使用文件读写来记录数据，当系统开始运行时从文件读取恢复数据，当程序运行结束向文件写数据保存。
2. 若当顾客第一次登录系统时，系统将帮顾客完成注册并自动登录。
3. 不同身份的人登录系统之后，所能做的操作不同。
4. 管理员登录系统后，可以通过 showStatus() 来查看当前酒店入住和预定情况，并且可以通过 setPrice() 和 setDiscounts() 来分别设置酒店客房价格和优惠折扣。
5. 顾客登录系统后，可以通过 ordelRoom() 来预订客房，并且当他之前预订过客房之后，若满足取消预订条件 (距离入住时间一天以内)，顾客可以通过 delOrder() 来取消预订

的房间。当顾客退房之后，可以通过 comment() 来留下评价，系统自动写入到评论文件。

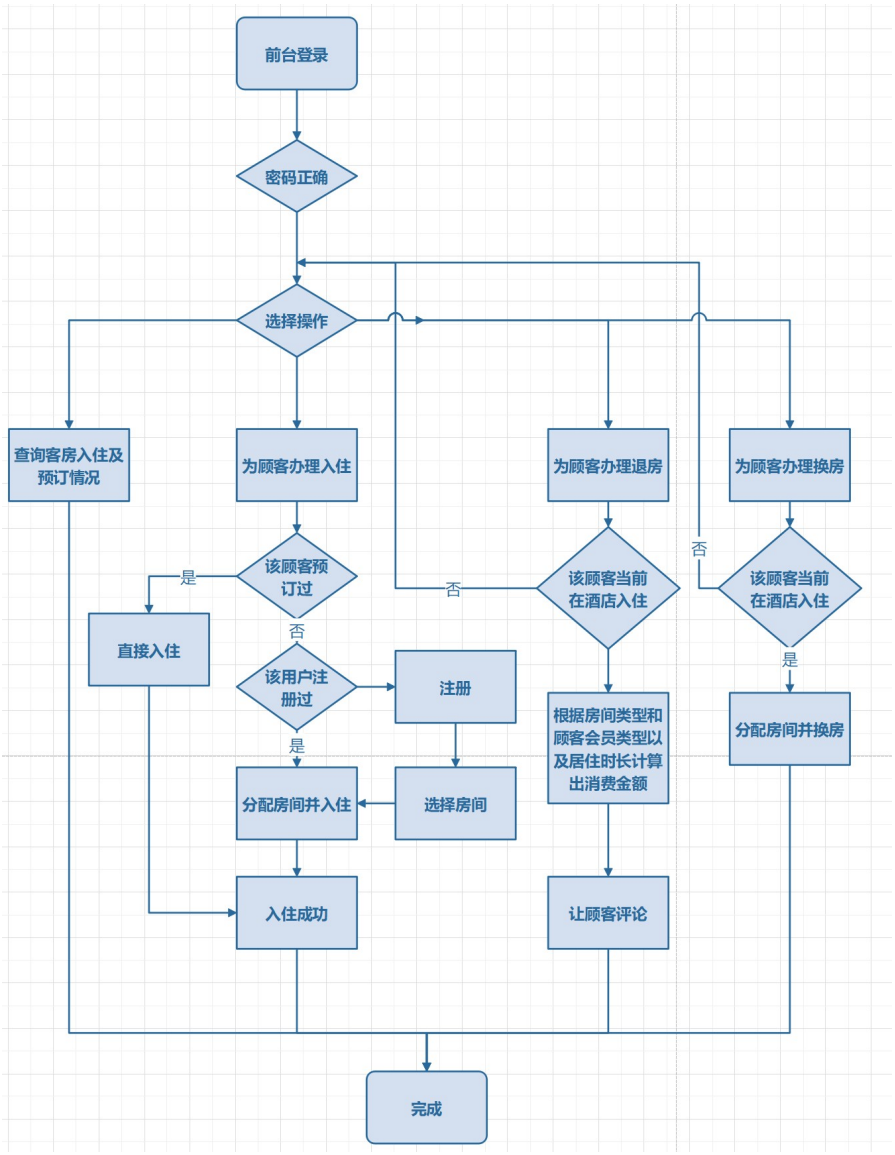
6. 前台登录系统后，可以对顾客提供服务，通过 checkIn() 可提供入住服务给顾客，若顾客预订过则直接入住；当顾客之前注册过系统但没有预订房间，则向顾客分配房间并完成入住服务；若若顾客没有注册过系统，则帮助顾客完成注册后分配房间完成入住服务。除了入住服务；前台还可通过 checkout() 实现对要退房的顾客提供退房服务，退房时根据顾客所入住的时间和房间类型以及顾客本身的会员优惠，计算出顾客的消费总额，来完成退房服务；此外，前台还可通过 exchRoom() 来提供给当前入住在酒店的顾客换房服务。

4 实现流程

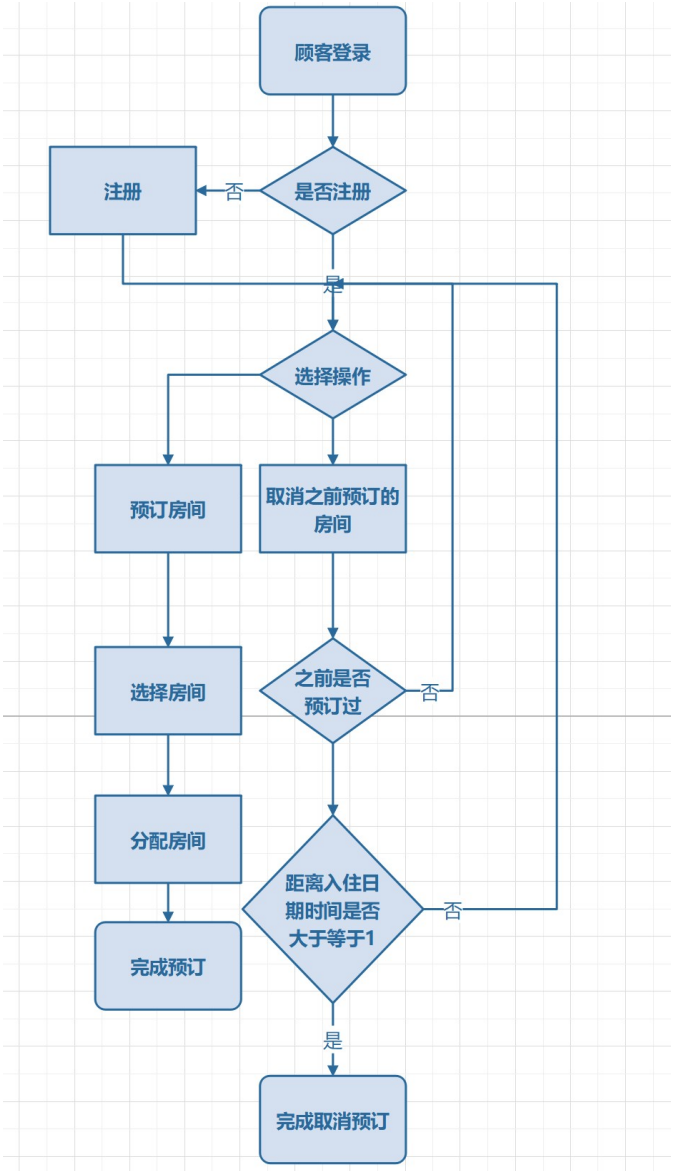
4.1 管理员



4.2 前台


前台登录
密码正确
选择操作
查询客房入住及预订情况
为顾客办理入住
为顾客办理退房
为顾客办理换房
该顾客预订过
是
否
直接入住
该用户注册过
注册
是
分配房间并入住
选择房间
入住成功
否
该顾客当前在酒店入住
根据房间类型和顾客会员类型以及居住时长计算出消费金额
让顾客评论
否
该顾客当前在酒店入住
是
分配房间并换房
完成

4.3 顾客



5 结果演示

5.1 顾客

5.1.1 顾客登录

1. 之前注册过直接登录

```
---读取数据成功---
**********************************
**********酒店客房管理系统**********
**********************************
* 顾客输入1
* 管理员输入2
* 前台输入3
* 退出输入0
1
---请输入账号密码以注册或登录---
---请输入id---
1004
---请输入密码---
123456
---登录成功---
```

2. **第一次登录系统**

```
---请输入账号密码以注册或登录---
---请输入id---
1006
---请输入密码---
123456
---您未注册，请选择要加入的会员类型来完成注册---
* 输入1选择金卡会员
* 输入2选择银卡会员
* 输入3选择普通会员
* 输入4选择不加入会员
2
---已帮您完成注册并登录---
* 输入1进行选房
* 输入2取消之前预定的房间
* 输入0返回上级
0
```

5.1.2 顾客预定房间

```
* 输入1进行选房
* 输入2取消之前预定的房间
* 输入0返回上级
1
---请选择您要预定的房间类型---
* 输入1选择豪华房：4床位
* 输入2选择标准房：2床位
* 输入3选择普通房：1床位
2
---请输入入住时间---
4
---预定成功---
```

5.1.3 顾客取消预订

```
---请输入账号密码以注册或登录---
---请输入id---
1005
---请输入密码---
12345
---密码错误---
---请输入账号密码以注册或登录---
---请输入id---
1005
---请输入密码---
123456
---登录成功---
* 输入1进行选房
* 输入2取消之前预定的房间
* 输入0返回上级
2
---请输入当前时间---
2
---取消预订成功---
```

5.1.4 顾客评论

```
------请输入评论，以回车结束---
很不错的酒店期待下次再来!
---评论成功---
```

5.2 管理员

5.2.1 管理员登陆

```
**********************************
**********酒店客房管理系统**********
**********************************
* 顾客输入1
* 管理员输入2
* 前台输入3
* 退出输入0
2
---请输入管理员密码---
0000
---以*管理员*身份登录成功---
* 输入1查询当前客房入住及预订情况
* 输入2设置客房价格
* 输入3设置顾客优惠政策
* 输入其他退出
```

5.2.2 查询当前客房入住及预订情况

```
* 输入1查询当前客房入住及预订情况
* 输入2设置客房价格
* 输入3设置顾客优惠政策
* 输入其他退出
1
******当前客房剩余量******
总量:  75  豪华房:  19 标准房:  17 普通房:  39
******当前客房预定情况******
顾客id:1002 房间号:101
******当前客房入住情况******
顾客id:1001 房间号:303
顾客id:1003 房间号:401
```

5.2.3 设置客房价格

```
* 输入1查询当前客房入住及预订情况
* 输入2设置客房价格
* 输入3设置顾客优惠政策
* 输入其他退出
2
* 请输入豪华房的价格
400
* 请输入标准房的价格
200
* 请输入普通房的价格
100
```

5.2.4 设置优惠政策

```
* 输入1查询当前客房入住及预订情况
* 输入2设置客房价格
* 输入3设置顾客优惠政策
* 输入其他退出
3
* 请输入金卡会员的折扣
0.8
* 请输入银卡会员的折扣
0.85
* 请输入普通会员的折扣
0.9
```

5.3 前台

5.3.1 前台登录

```
**********************************
**********酒店客房管理系统**********
**********************************
* 顾客输入1
* 管理员输入2
* 前台输入3
* 退出输入0
3
---请输入前台密码---
0000
---以*前台*身份登录成功---
```

5.3.2 查询当前客房入住和预定情况

```
******当前客房剩余量******
总量:  75  豪华房:  19 标准房:  17 普通房:  39
******当前客房预定情况******
顾客id：1002 房间号：101
******当前客房入住情况******
顾客id：1001 房间号：303
顾客id：1003 房间号：401
```

5.3.3 为顾客办理入住服务

1. 之前预订过的顾客

```
* 输入1查询当前客房入住及预订情况
* 输入2为顾客办理入住
* 输入3设置为顾客办理退房
* 输入4为顾客办理换房
* 输入其他退出
2
---请输入顾客id---
1002
---此顾客已预定，直接入住---
---房间号：101---
---入住成功---
```

2. 没有预订但注册了系统的用户

```
---请输入顾客id---
1004
---请输入当前时间---
5
---请选择顾客要入住的房间类型---
---请选择您要预定的房间类型--
* 输入1选择豪华房：4床位
* 输入2选择标准房：2床位
* 输入3选择普通房：1床位

1
---房间号：402---
---入住成功---
```

3. 之前没有注册过系统的用户

```
---请输入顾客id---
1008
---请输入当前时间---
4
---请选择顾客要入住的房间类型---
---请选择您要预定的房间类型--
* 输入1选择豪华房：4床位
* 输入2选择标准房：2床位
* 输入3选择普通房：1床位

2
---当前用户未注册，请设置密码完成注册---
123456
---请输入要加入的会员类型---
* 输入1选择金卡会员
* 输入2选择银卡会员
* 输入3选择普通会员
* 输入4选择不加入会员
2
---房间号：304---
---入住成功---
```

5.3.4 为顾客办理换房服务

```
* 输入1查询当前客房入住及预订情况
* 输入2为顾客办理入住
* 输入3设置为顾客办理退房
* 输入4为顾客办理换房
* 输入其他退出
4
请输入顾客id
1003
---换房成功---
---新换的房间号为420---
```

```
******当前客房剩余量******
总量:   75   豪华房:   19 标准房:   17 普通房:   39
******当前客房预定情况******
******当前客房入住情况******
顾客id: 1001 房间号: 303
顾客id: 1002 房间号: 101
顾客id: 1003 房间号: 420
顾客id: 1004 房间号: 402
顾客id: 1008 房间号: 304
```

5.3.5 为顾客办理退房服务

```
---请输入当前时间---
1001
---请输入该顾客的id---
10010
---未找到此顾客入住信息，请重新输入---
---请输入当前时间---
10
---请输入该顾客的id---
1001
---消费金额总计: 576.00
------请输入评论，以回车结束---
```

5.4 数据文件

5.4.1 用户信息

```
0 0000 0 0 0
1 0000 0 0 0
1001 123456 1 6 0
1002 123456 2 5 0
1003 123456 4 3 0
1004 123456 1 4 0
1005 123456 3 4 0
1006 123456 2 0 0
1008 123456 2 4 0
```

0表示管理员
1表示前台

5.4.2 房间信息

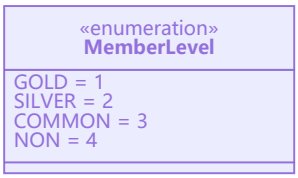
```
101 3 0 1
102 3 0 0
103 3 0 0
104 3 0 0
105 3 0 0
106 3 0 0
107 3 0 0
108 3 0 0
109 3 0 0
110 3 0 0
111 3 0 0
112 3 0 0
113 3 0 0
```

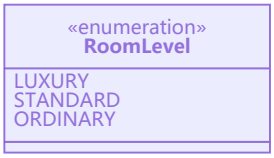
5.4.3 入住记录

```
1002 101
1003 420
1004 402
1008 304
```

6 问题和解决

1. 刚开始使用 int 类型来表示房间类型和会员类型使得代码可读性不是很好，之后改为用枚举类型实现。





2. 刚开始在系统运行之前使用数组存储从文件恢复的数据，但发现查找效率不是太高，所以改用 map 来实现，使得程序的效率提升。

```
map<int, Room> rooms;
map<int, int> ORDER;
map<int, int> IN;
map<int, Customer> USR;
```

7 附录——源代码

7.1 头文件 Header.h

```
#ifndef HEADER_H_INCLUDED
#define HEADER_H_INCLUDED

#include <map>

enum RoomLevel {
    LUXURY = 1, STANDARD = 2, ORDINARY = 3
};
struct roomCfg {
    int bedNum;
    double price;
};

class Room {
    static int remainTotal;
    static int remainLuxury;
    static int remainStandard;
    static int remainOrdinary;
    static std::map<RoomLevel, roomCfg> roomInfo;
    int roomId;
    RoomLevel roomType;
    bool ordered;
    bool checking;

public:
    static int getRemainTotal();

    static int getRemainLuxury();

    static int getRemainStandard();

    static int getRemainOrdinary();

    static roomCfg getRoomInfo(RoomLevel);

    static int allocateRoom(std::map<int, Room> &, RoomLevel, bool);
```

```
    Room();

    Room(int, RoomLevel, bool, bool);

    int getRoomId();

    int getRoomType();

    bool isOrdered();

    void setOrder(bool);

    bool isChecking();

    void setCheck(bool);

    friend std::ostream &operator<<(std::ostream &, const Room &);
};

class Person {
protected:
    int id;
    std::string password;

public:
    Person();

    Person(int, std::string);

    int getId();

    std::string getPassword();
};

enum MemberLevel {
    GOLD = 1, SILVER = 2, COMMON = 3, NON = 4
};

class Customer : public Person {
    MemberLevel memberType;
    int checkinTime;
    bool order;

public:
    Customer();

    Customer(int, std::string, MemberLevel, int, bool);

    MemberLevel getMemberType();
```

```
    int getCheckinTime();

    bool isOrder();

    void delOrder(std::map<int, int> &, bool, int);

    void orderRoom(std::map<int, Room> &, std::map<int, int> &);

    static void comment();

    friend std::ostream &operator<<(std::ostream &, const Customer &);
};

class Employee : public Person {
public:
    static void showStatus(std::map<int, int> &, std::map<int, int> &);
};

class Administrator : public Employee {
    static std::map<MemberLevel, double> discounts;
    static std::map<RoomLevel, double> prices;

public:

    static void setDiscounts();

    static void setPrice();

    static double getDiscounts(MemberLevel);

    static double getPrice(RoomLevel);
};

class Perception : public Employee {
public:
    static void checkin(std::map<int, Room> &, std::map<int, int> &, std::map<int, int> &,
                        std::map<int, Customer> &);

    static void checkout(std::map<int, Room> &, std::map<int, int> &, std::map<int, Customer> &);

    static void exchRoom(std::map<int, Room> &, std::map<int, int> &);
};

#endif
```

7.2 Room.cc

```
#include <iostream>

#include "Header.h"

Room::Room() {}

Room::Room(int _roomId, RoomLevel _roomType, bool _ordered, bool _checking)
       : roomId(_roomId),
         roomType(_roomType),
         ordered(_ordered),
         checking(_checking) {
   if (!ordered && !checking) {
       if (_roomType == 1)
           remainLuxury++;
       else if (_roomType == 2)
           remainStandard++;
       else if (_roomType == 3)
           remainOrdinary++;
       remainTotal++;
   }
}

int Room::remainTotal = 0;
int Room::remainLuxury = 0;
int Room::remainStandard = 0;
int Room::remainOrdinary = 0;
std::map<RoomLevel, roomCfg> Room::roomInfo = {
       {LUXURY, {4, 400}},
       {STANDARD, {2, 200}},
       {ORDINARY, {1, 100}}};

int Room::getRemainTotal() { return remainTotal; }

int Room::getRemainLuxury() { return remainLuxury; }

int Room::getRemainStandard() { return remainStandard; }

int Room::getRemainOrdinary() { return remainOrdinary; }

int Room::getRoomId() { return roomId; }

int Room::getRoomType() { return roomType; }

bool Room::isOrdered() { return ordered; }

bool Room::isChecking() { return checking; }

void Room::setCheck(bool in) { checking = in; }

void Room::setOrder(bool _order) { ordered = _order; }
```

```
roomCfg Room::getRoomInfo(RoomLevel _roomlevel) { return roomInfo[_roomlevel]; }

// 分配房间
int Room::allocateRoom(std::map<int, Room> &rooms, RoomLevel _roomlevel, bool orderOrCheckin) {
    for (auto &e : rooms) {
        if (e.second.roomType == _roomlevel && !e.second.ordered && !e.second.checking) {
            if (orderOrCheckin) e.second.checking = true;
            else
                e.second.ordered = true;
            return e.second.roomId;
        }
    }
    return -1;
}

std::ostream &operator<<(std::ostream &o, const Room &r);

std::ostream &operator<<(std::ostream &o, const Room &r) {
    o << r.roomId << " " << r.roomType << " " << r.ordered << " " << r.checking;
    return o;
}
```

7.3 Person.cc

```
#include "Header.h"

Person::Person() {}

Person::Person(int _id, std::string _psw) : id(_id), password(_psw) {}

int Person::getId() { return id; }

std::string Person::getPassword() { return password; }
```

7.4 Customer.cc

```
#include <iostream>
#include <fstream>

#include "Header.h"

Customer::Customer() = default;

Customer::Customer(int _id, std::string _psw, MemberLevel _memberType,
                   int _checkinTime, bool _order)
        : Person(_id, _psw),
```

```
        memberType(_memberType),
        checkinTime(_checkinTime),
        order(_order) {}

MemberLevel Customer::getMemberType() { return memberType; }

int Customer::getCheckinTime() { return checkinTime; }

bool Customer::isOrder() { return order; }

void Customer::delOrder(std::map<int, int> &ORDER, bool perception, int curTime) {
    if (!ORDER.count(id)) {
        puts("---您之前未预定---");
        return;
    }
    if (!perception) {
        puts("---请输入当前时间---");
        std::cin >> curTime;
    }
    if (checkinTime - 1 >= curTime || perception) {
        this->order = false;
        ORDER.erase(id);
    } else {
        puts("---无法取消!!!---");
    }
}

std::ostream &operator<<(std::ostream &o, const Customer &c) {
    o << c.id << ' ' << c.password << ' ' << c.memberType << ' '
      << c.checkinTime << ' ' << c.order;
    return o;
}

void Customer::orderRoom(std::map<int, Room> &rooms, std::map<int, int> &ORDER) {
    order = true;
    puts(
        "---请选择您要预定的房间类型---\n* 输入1选择豪华房：4床位\n* 输入2选择标准房：2床位\n* 输入
            3选择"
        "普通房：1床位");
    int room_level;
    std::cin >> room_level;
    int remain = Room::getRemainTotal();
    switch (room_level) {
        case 1:
            remain = Room::getRemainLuxury();
            break;
        case 2:
            remain = Room::getRemainStandard();
            break;
        case 3:
```

```
            remain = Room::getRemainOrdinary();
        default:
            break;
    }
    if (remain <= 0) {
        puts("---抱歉，当前没有剩余房间！！！---");
        return;
    } else {
        int room_id = Room::allocateRoom(rooms, RoomLevel(room_level), 0);
        ORDER[id] = room_id;
        puts("---请输入入住时间---");
        std::cin >> this->checkinTime;
        order = true;
        puts("---预定成功---");
    }
}

void Customer::comment() {
    puts("---请输入评论，以回车结束---");
    char str[100];
    std::cin.get();
    std::cin.get(str, 100, '\n');
    // 以追加模式将评论写入文件
    std::ofstream out(R"(D:\win_CPP_test\OOP\room_management\comments.txt)", std::ios::out | std::
        ios::app);
    out << str << std::endl;
    puts("---评论成功---");
    out.close();
}
```

7.5 Employee.cc

```
#include <iostream>

#include "Header.h"

void Employee::showStatus(std::map<int, int> &ORDER, std::map<int, int> &IN) {
    puts("******当前客房剩余量******");
    printf("总量: %4d\t豪华房: %4d\t标准房: %4d\t普通房: %4d\n", Room::getRemainTotal(), Room::
        getRemainLuxury(),
         Room::getRemainStandard(), Room::getRemainOrdinary());
    puts("******当前客房预定情况******");
    for (const auto &e : ORDER) {
        std::cout << "顾客id: " << e.first << "\t房间号: " << e.second << std::endl;
    }
    puts("******当前客房入住情况******");
    for (const auto &e : IN) {
        std::cout << "顾客id: " << e.first << "\t房间号: " << e.second << std::endl;
    }
```

```
}
```

7.6 Administrator.cc

```
#include <iostream>

#include "Header.h"

std::map<MemberLevel, double> Administrator::discounts = {
       {GOLD, 0.8},
       {SILVER, 0.9},
       {COMMON, 0.9},
       {NON, 1.0}};
std::map<RoomLevel, double> Administrator::prices = {
       {LUXURY, 400},
       {STANDARD, 200},
       {ORDINARY, 100}};

void Administrator::setDiscounts() {
   double discounts[3];
   std::cout << "* 请输入金卡会员的折扣" << '\n';
   std::cin >> discounts[0];
   std::cout << "* 请输入银卡会员的折扣" << '\n';
   std::cin >> discounts[1];
   std::cout << "* 请输入普通会员的折扣" << '\n';
   std::cin >> discounts[2];
   Administrator::discounts = {{GOLD, discounts[0]},
                               {SILVER, discounts[1]},
                               {COMMON, discounts[2]},
                               {NON, 1.0}};
}

double Administrator::getDiscounts(MemberLevel e) { return discounts[e]; }

void Administrator::setPrice() {
   double prices[3];
   std::cout << "* 请输入豪华房的价格" << '\n';
   std::cin >> prices[0];
   std::cout << "* 请输入标准房的价格" << '\n';
   std::cin >> prices[1];
   std::cout << "* 请输入普通房的价格" << '\n';
   std::cin >> prices[2];
   Administrator::prices = {
          {LUXURY, prices[0]},
          {STANDARD, prices[1]},
          {ORDINARY, prices[2]}};
}

double Administrator::getPrice(RoomLevel e) { return prices[e]; }
```

7.7 main.cpp

```
#include <fstream>
#include <iostream>
#include <list>

#include "Header.h"

using namespace std;
map<int, Room> rooms;

map<int, int> ORDER;
map<int, int> IN;
map<int, Customer> USR;

//读入数据
void read() {
    ifstream in1(R"(D:\win_CPP_test\OOP\room_management\room.txt)");
    int _roomId, _roomType, _ordered, _checking;
    while (in1 >> _roomId >> _roomType >> _ordered >> _checking) {
        rooms[_roomId] = Room(_roomId, RoomLevel(_roomType), _ordered, _checking);
    }
    in1.close();
    ifstream in2(R"(D:\win_CPP_test\OOP\room_management\order_log.txt)");
    int k, v;
    while (in2 >> k >> v) {
        ORDER[k] = v;
    }
    in2.close();
    ifstream in3(R"(D:\win_CPP_test\OOP\room_management\in_log.txt)");
    while (in3 >> k >> v) {
        IN[k] = v;
    }
    in3.close();
    ifstream in4(R"(D:\win_CPP_test\OOP\room_management\usr.txt)");
    int _id, _member_type, _checkin_time, _order;
    string _pwd;
    while (in4 >> _id >> _pwd >> _member_type >> _checkin_time >> _order) {
        USR[_id] = Customer(_id, _pwd, MemberLevel(_member_type), _checkin_time,
                            _order);
    }
    in4.close();
    puts("---读取数据成功---");
}

// 写出数据
void write() {
```

```
    ofstream out1(R"(D:\win_CPP_test\OOP\room_management\room.txt)", ios::out | ios::trunc);
    for (const auto &e : rooms) {
        out1 << e.second << '\n';
    }
    out1.close();
    ofstream out2(R"(D:\win_CPP_test\OOP\room_management\order_log.txt)", ios::out | ios::trunc);
    for (const auto &e : ORDER) {
        out2 << e.first << " " << e.second << '\n';
    }
    out2.close();
    ofstream out3(R"(D:\win_CPP_test\OOP\room_management\in_log.txt)", ios::out | ios::trunc);
    for (const auto &e : IN) {
        out3 << e.first << " " << e.second << '\n';
    }
    out3.close();
    ofstream out4(R"(D:\win_CPP_test\OOP\room_management\usr.txt)", ios::out | ios::trunc);
    for (const auto &e : USR) {
        out4 << e.second << '\n';
    }
    out4.close();
    puts("---保存数据成功---");
}

void administrator() {
    puts("---请输入管理员密码---");
    string pwd;
    cin >> pwd;
    if (USR[0].getPassword() == pwd) {
        puts("---以*管理员*身份登录成功---");
    } else {
        puts("---密码错误! ---");
        administrator();
    }
    while (true) {
        int select;
        puts("* 输入1查询当前客房入住及预订情况\n* 输入2设置客房价格\n* 输入3设置顾客优惠政策\n* 输入其
            他退出");
        cin >> select;
        if (select == 1) {
            Administrator::showStatus(ORDER, IN);
        } else if (select == 2) {
            Administrator::setPrice();
        } else if (select == 3) {
            Administrator::setDiscounts();
        } else
            break;
    }
}

void perception() {
```

```
        puts("---请输入前台密码---");
        string pwd;
        cin >> pwd;
        if (USR[1].getPassword() == pwd) {
            puts("---以*前台*身份登录成功---");
        } else {
            puts("---密码错误! ---");
            perception();
        }
        while (true) {
            puts("* 输入1查询当前客房入住及预订情况\n* 输入2为顾客办理入住\n* 输入3设置为顾客办理退房\n* 输
                入4为顾客办理换房\n* 输入其他退出");
            int select;
            cin >> select;
            if (select == 1) {
                Perception::showStatus(ORDER, IN);
            } else if (select == 2) {
                Perception::checkin(rooms, ORDER, IN, USR);
            } else if (select == 3) {
                Perception::checkout(rooms, IN, USR);
            } else if (select == 4) {
                Perception::exchRoom(rooms, IN);
            } else break;
        }
    }

    void customer() {
        puts("---请输入账号密码以注册或登录---");
        int id;
        string pwd;
        puts("---请输入id---");
        cin >> id;
        puts("---请输入密码---");
        cin >> pwd;
        if (!USR.count(id)) {
            puts("---您未注册，请选择要加入的会员类型来完成注册---");
            puts("* 输入1选择金卡会员\n* 输入2选择银卡会员\n* 输入3选择普通会员\n* 输入4选择不加入会员");
            int mem_type;
            cin >> mem_type;
            USR[id] = Customer(id, pwd, MemberLevel(mem_type), 0, 0);
            puts("---已帮您完成注册并登录---");
        } else {
            if (USR[id].getPassword() == pwd)
                puts("---登录成功---");
            else {
                puts("---密码错误---");
                customer();
            }
        }
        puts("* 输入1进行选房\n* 输入2取消之前预定的房间\n* 输入0返回上级");
```

```
        int confirm;
        cin >> confirm;
        if (confirm == 1) {
            USR[id].orderRoom(rooms, ORDER);
        } else if (confirm == 2) {
            USR[id].delOrder(ORDER, false, 0);
        } else if (confirm == 0) return;
    }


    int main() {
        read();
        while (true) {
            puts("*********************************");
            puts("**********酒店客房管理系统**********");
            puts("*********************************");
            int login_select = -1;
            puts("* 顾客输入1\n* 管理员输入2\n* 前台输入3\n* 退出输入0");
            cin >> login_select;
            if (login_select == 1) {
                customer();
            } else if (login_select == 2) {
                administrator();
            } else if (login_select == 3) {
                perception();
            } else if (login_select == 0) {
                break;
            }
        }
        write();
        return 0;
    }
```

7.8 CMakeLists.txt

```
cmake_minimum_required(VERSION 3.17)
project(room_management)

set(CMAKE_CXX_STANDARD 14)

add_executable(room_management main.cpp Header.h Employee.cc Customer.cc Perception.cc Room.cc
Administrator.cc Person.cc)
```