****

Hotel Management System Embedded with Time

Documentation

**Instructor: Noman Atique**

**CS-201 Computer Programming**

**Section E**

# GROUP MEMBERS:

|  |  |
| --- | --- |
| Group Members | Student ID: |
| Taban Shaukat (Leader) | 16K-3937 |
| Huzaifah Punjani | 16K-3924 |
| Fareeha Sattar Shaikh | 16K-3934 |

# ABSTRACT

This project is a Hotel Management System enriched with the concepts of Objected Oriented Programming. It gives the relation between different classes and how they are interacting with each other and with the date and time.

This project mainly uses the core concepts of OOP (Objected Orientated Programming). This also serves as a challenge for us to interlink time with classes. This documentation covers the concise design process of the code, simulations in order to verify functionality, and an effective project which is successfully implemented.

We have tried to use each and every concept of OOP to use in this project. This project provides an ease for the managers who face problems every day in booking, arrival and departure of people.

# TABLE OF CONTENTS

[GROUP MEMBERS: 1](#_Toc483781012)

[ABSTRACT 2](#_Toc483781013)

[TABLE OF CONTENTS 3](#_Toc483781014)

[1.Motivation 4](#_Toc483781015)

[2 SCOPE: 5](#_Toc483781016)

[3. LIST OF FUNCTIONALITIES: 6](#_Toc483781017)

[4. LIST THE CONCEPTS OF OOP THAT YOU'VE USED IN YOUR PROJECT ALONG WITH THE NAME OF FUNCTION, CLASSES: 7](#_Toc483781018)

[4.1 Concept of Association: 7](#_Toc483781019)

[4.2 CONCEPT OF INHERITANCE: 7](#_Toc483781020)

[4.3 CONCEPT OF AGGREGATION: 7](#_Toc483781021)

[4.4 CONCEPT OF COMPOSITION: 8](#_Toc483781022)

[4.5 CONCEPT OF static members and static member functios: 8](#_Toc483781023)

[4.6 concept of friend functions and operator overloading: 8](#_Toc483781024)

[4.7 concept of virtual functions: 8](#_Toc483781025)

[4.8 concept of copy constructor: 8](#_Toc483781026)

[4.9 CONCEPT OF VIRTUAL DESTRUCTOR: 9](#_Toc483781027)

[4.10 concept of constant MEMBERS AND CONSTANT functions: 9](#_Toc483781028)

[4.11 CONCEPT OF STATIC CAST: 9](#_Toc483781029)

[4.12 SEPARATE HEADER FILES and IMPLEMENTATION: 9](#_Toc483781030)

[4.13 CONCEPT OF EXCEPTION HANDLING 9](#_Toc483781031)

[5. CLASS DIAGRAM: 10](#_Toc483781032)

# 1.Motivation

Achieving effective and efficient Hotel Management System Embedded with time is the key to human Service sector and development nowadays, it has emerged as an urgent industry concern. The realization of the limited availability of Management systems that are capable of cultivation under time management condition has caused us designing of this project.

# 2 SCOPE:

The scope of our project is for the newly start up hotels. It provides a dedicated management of customer with respect to food, room and with the association of date and time. Our project is not generalized for every type of management. It is for specialized purposes.

# 3. LIST OF FUNCTIONALITIES:

In this system, we had make extensive use of core object oriented concepts of C++. We have a customer id in system initially. In our system we have separate implementations for

• Getting the information and Displaying Data

• Getting customer information who are lodged in

• Allocating a room to the customer

• Checking the availability.

• Ordering of Food.

• Checking in and out of Customer According to Time and Date.

• Preparing a billing through calculation for the customer according to Date of Check-in and

Check-Out.

# 4. LIST THE CONCEPTS OF OOP THAT YOU'VE USED IN YOUR PROJECT ALONG WITH THE NAME OF FUNCTION, CLASSES:

## 4.1 Concept of Association:

The following classes uses the relationship of association:

1. FoodItems with orderFood.

**orderFood::orderFood(int q,int i,Fooditems\*F)**

1. Bill with Check
2. Bill with Customer

**float Bill::calculateRoomBill(Customer \*C,Check\* DT)**

1. OrderFood with Customer

**void Customer::setOrderedFood(orderFood\*\* O)**

1. FoodItem with customer

**Fooditems\* getOrderedFoodItems(int);**

## 4.2 CONCEPT OF INHERITANCE:

The following classes uses the relationship of inheritance:

1. Employee is inherited from person.

**class Customer:public Person**

1. Customer is inherited from Person.

**class Employee:public Person**

## 4.3 CONCEPT OF AGGREGATION:

The following classes uses the relationship of Aggregation:

1. Room with customer

**void Customer::BooksRoom(Rooms \*R)**

## 4.4 CONCEPT OF COMPOSITION:

The Check class is composed of DateTime.

**Check::Check()**

**{ dateIn=new DateTime;**

**dateOut=new DateTime;**

**}**

## 4.5 CONCEPT OF static members and static member functios:

1. In class Bill we have made static member functions of getBillNo() and getPricePerDay().
2. In class Bill we have made the member variable billNo.
3. In class Rooms the total variable is made static.
4. In class Rooms the member function set/getTotalEmptyRooms are static.
5. In class orderFood the member variable quantity is static and its getters and setters are also static member functions.

## 4.6 concept of friend functions and operator overloading:

1. We have overloaded “<<” and have made this function friend of the class Customer.
2. We have overloaded “()” inside the class of DateTime
3. We have overloaded “++” and have made this function friend of class Bill
4. We have overloaded “- - “and have made this function friend of class Bill
5. We have overloaded “=” inside the class of Rooms.

## 4.7 concept of virtual functions:

We have made a virtual function “printInfo()” in Person Class.

## 4.8 concept of copy constructor:

We have made the copy constructor in Room, Person, Customer, Check and OrderFood Class.

**Person::Person(const Person& obj)**

**Rooms::Rooms(const Rooms& obj)**

**Customer::Customer(const Customer& obj)**

**orderFood::orderFood(const orderFood& obj)**

**Check::Check(const Check& obj)**

## 4.9 CONCEPT OF VIRTUAL DESTRUCTOR:

1. We have made customer and employee class as the child classes of Person’s Class that contains the virtual destructor.

## 4.10 concept of constant MEMBERS AND CONSTANT functions:

1. In class Bill we have made the variable “tax” **const static**
2. In class Bill we have made the variable “PricePerDay” **const static**

## 4.11 CONCEPT OF STATIC CAST:

Static cast is used in the Login header file to cast char to int.

**while (static\_cast<int>( temp = getch() ) != 13)**

## 4.12 SEPARATE HEADER FILES and IMPLEMENTATION:

A separate header file is made for the declarations of the complete program, and another header file is made for the Login purposes. The implementation file is provided separately to achieve the good software engineering.

## 4.13 CONCEPT OF EXCEPTION HANDLING

We have also used the concept of Exception Handling by using throw, catch and try function.

# 5 header file

#include <iostream>

#include <string>

#include <iomanip>

#include <vector>

#include<windows.h>

#include<fstream>

using namespace std;

class Rooms, Persons, Customer, Employee, Fooditems, orderFood, Bill,Check,DateTime;

class Rooms {

static int total;

protected:

int roomNo;

bool isEmpty;

// int arrayEmptyRooms[];

public:

Rooms();

Rooms(int r, bool IE);

Rooms(const Rooms& obj);

bool getIsEmpty();

void setIsEmpty(bool);

void setRoomNo(int);

void CustomerInfo(Customer& obj);

void UpdateRoomInfo(int,bool);

int getRoomNo();

//void CheckAvailability();

//void BookRoom(Rooms&);

static int getTotalEmptyRooms();

static int setTotalEmptyRooms(int);

void operator =(const Rooms& obj1);

};

int Rooms::total=0;

class Person {

protected:

int ID;

char\* Name;

long int Telephone\_num;

char Gender;

string Address;

public:

Person(int ,char\* ,long int ,char ,string );

Person(const Person& obj);

void setName(char\* n);

void setTelephoneNum(long int tn);

void setGender(char g);

void setAddress(string a);

void setID(int id);

int getID();

char\* getName();

long int getTelephoneNum();

virtual void printinfo();

char getGender();

string getAddress();

virtual ~Person();

};

class Customer:public Person {

protected:

static int index;

Rooms\* roomAlloted;

orderFood\*\* orderedFoods;

Check \*checkInOut;

public:

Customer(int,char\*,long int,char,string);

Customer(const Customer& obj);

Rooms\* getRoomAlloted();

void printinfo();

void setCheckInOut(Check \*C);

Check\* getCheckInOut();

void CheckIn();

void CheckOut();

//void PayBill();

// void OrderFoodItems();

bool BooksRoom(Rooms \*R);

void setOrderedFoodItems(Fooditems \*F);

void setOrderedFood(orderFood \*\*O);

Fooditems\* getOrderedFoodItems(int);

orderFood\*\* getOrderedFoods();

friend ostream& operator << (ostream &out, const Customer &obj1);

~Customer();

};

int Customer::index;

class Employee:public Person {

protected:

string Designation;

float Salary;

int Hours;

public:

Employee();

void setDesignation(string);

void setSalary(float);

void setHours(int);

string getDesignation();

float getSalary();

int getHours();

void printinfo();

~Employee();

};

class Fooditems {

protected:

int FoodId;

float PricePerItem;

string Name;

// string foodType;

public:

Fooditems(int,float,string);

int getFoodId();

void setName(string);

string getName();

float getPricePerItem();

void setPricePerItem(float);

void displayMenu();

};

class orderFood {

protected:

int orderNo;

int quantity;

Fooditems\* Foodordered;

public:

orderFood(int q,int i,Fooditems\*F);

orderFood(int,int);

orderFood(const orderFood& obj);

void setFoodQuantity(int q);

void setFoodOrdered(Fooditems\* F);

Fooditems\* getFoodOrdered();

int getFoodQuantity();

int getOrderNo();

};

class Bill {

protected:

float amount;

const static float tax=0.19;

const static int PricePerDay=5000;

Customer\* Client;

Check\* checkInOut;

public:

static int billNo;

Bill();

float getAmount();

float getTaxedAmount();

float calculateRoomBill(Customer \*C,Check \*DT);

static int getBillNo();

void DisplayBill();

void setAmount(float n);

static int getPricePerDay();

float calculateFoodBill(Customer \*C);

friend Bill operator +(Bill & lhs,Bill &rhs);

friend Bill operator -(Bill & lhs,Bill &rhs);

friend float operator ++(Bill & obj);

friend float operator --(Bill & obj);

float operator ++(int);

float operator --(int);

};

class Check {

protected:

DateTime \*dateIn;

DateTime \*dateOut;

public:

Check();

Check(const Check& obj);

DateTime\* getDateIn();

DateTime\* getDateOut();

//void makeRoomEmpty(Customer &C);

void setDateTimeIn(int h,int m,int d,int mo,int y);

void setDateTimeOut(int h,int m,int d,int mo,int y);

};

class DateTime {

protected:

int hour,min,day,month,year;

public:

DataTime();

DateTime(int,int,int,int,int);

int getHour();

int getMin();

int getDay();

int getMonth();

int getYear();

void setDateTime(int hour,int min,int day,int month,int year);

void DisplayDateTime();

};

# 5. CLASS DIAGRAM:

