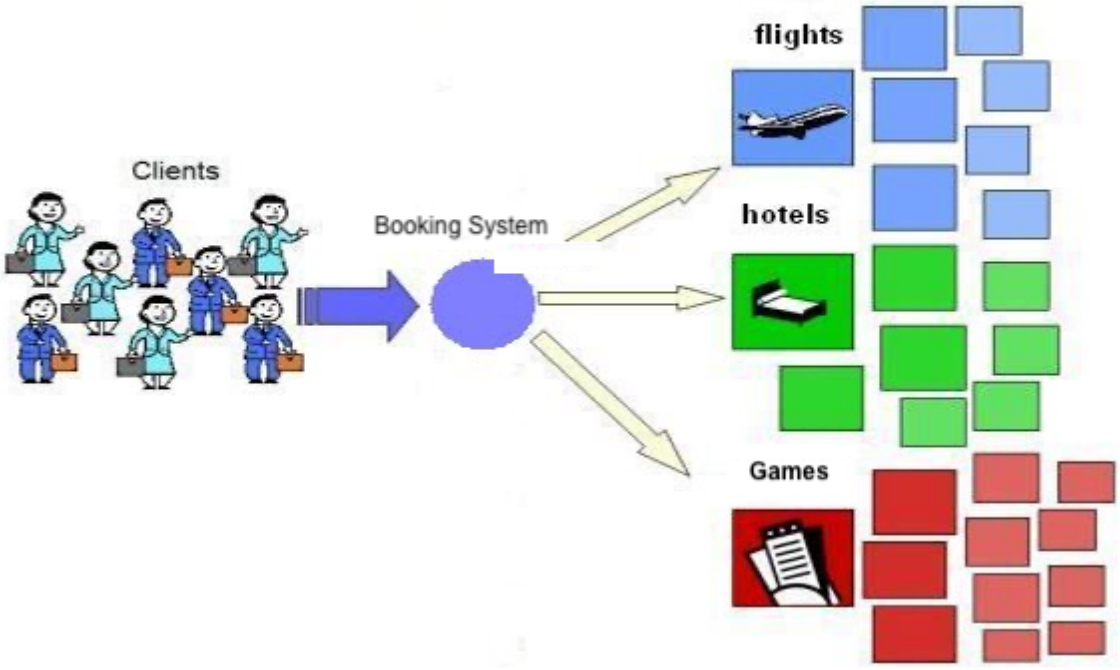
COMP 2014 Object Oriented Programming

Assignment 2

Submit Deadline：5:00pm Friday 14 Oct 2022

1. Problem: Smart booking system

Assume that you are creating a **smart booking system**, selling packages for the 2026 FIFA World Cup. Each package consists of three categories of goods: *flights*, *hotels,* and game *tickets*. The world cup 2026 is hosted by multiple cities. To simplify this task, we assume, that all the games are hosted in one city New York.



The World cup lasts ten days from day 0 to day 9. Assume that you have negotiated with the airlines, hotels and game hosts for flight tickets, hotel rooms, and game tickets for the time period.

Flight tickets: The full price for a one-way ticket between Sydney and New York is $2000. However, a discount can be given depending on the days of departure and arrival.

The discount for a ticket from Sydney to New York is *day*5%*. The discount for a return ticket from New York to Sydney is *(9-day)*5%.* For instance, if you travel to New York on day 3 and return on day 13, the price of your tickets is:

$2000(1 – 3 * 5%) + $2000*(1- (9-8) * 5%) = $3600.*

The idea here is "*the closer to the end, the cheaper the fly-in tickets; the closer to the end, the dearer the fly-out tickets"*. It is assumed that the supply of flight tickets is unlimited.

Hotel rooms: There are three different types of hotel rooms in the World cup Village: regular, bronze and gold. See the following table for the information about hotel rooms:

Types of rooms	Full price (per room per night)	Total rooms available (quota) each day	Discount
Regular	$150	25	No discount
Bronze	$220	20	20% if vacancy > 50%
Gold	$310	40	40% if vacancy > 50% 20% if vacancy > 20%

Hotel rooms are paid up-front. They are sold in vouchers. Each voucher is for a single day stay. For instance, if you travel to New York on day 3 and return on day 8, you will need to buy five hotel vouchers, one for each day. You could be allocated to different types of rooms in different days.

Game tickets: There are a total of 15 games in 10 days. See the following table for detailed information:

No	Game day	Games	Ticket price	Quantity available (quota)
0	0	Opening	$2000	60
1	1	Round of 16-Group A	$170	45
2	1	Round of 16-Group B	$80	40
3	2	Round of 16-Group C	$80	45
4	2	Round of 16-Group D	$150	50
5	3	Round of 16-Group E	$480	45
6	3	Round of 16-Group F	$110	55
7	4	Round of 16-Group G	$130	45
8	4	Round of 16-Group H	$90	35
9	5	Quarter-finals-1	$80	20
10	5	Quarter-finals-2	$100	25
11	6	Semi-final	$300	35
12	7	Third place play-off	$100	45
13	8	Final	$100	30
14	9	Closing	$800	40

Travel package: A travel package contains one fly-in air ticket (from Sydney to New York), one fly-out ticket (from New York to Sydney), the hotel rooms and a set of game tickets. A package is *feasible* if it satisfies the following conditions:

- The date of fly-in ticket is earlier than the date of fly-out ticket.
- The hotel rooms cover every night between the arrival and departure dates (not including the fly-out day).
- The hotel types should be the same or above the customer's desired hotel type.
- All the game tickets should be in the dates between fly-in and fly-out (can be in the same day of arrival or departure; can also attend multiple games on the same day). One game, one ticket.
- Only the games a customer requires can be included in the booking package to the customer. They can be less but each package must contain at least one game.
- The total costs should not exceed customer's budget.

Booking system's profit: The booking system receives 5% profit from the sales of air tickets and hotel vouchers, and 15% from the sales of game tickets for each feasible package. There is no profit or loss on unsuccessful requirements.

You will be given a list of customer requirements (randomly generated). Each customer requirement consists of:

- the budget of the customer (the upper limit the customer can pay)
- desired hotel type (regular, bronze, gold)
- a set of desired games to attend (up to 8 games)

The requirements are recorded in a text file. Download an example of requirement list from vUWS under this folder (requirementList.txt). The format of each requirement in the text file is the following:

$$budget, hotelType, [game_1, game_2, \ldots, game_n]$$

Note that there is no specification of travel dates in customer's requirements. By default, all travel package contains a return ticket, which cover all days for all the games and hotel stays. In other words, the days of games in the package are determined by the dates of travel and hotel stays.

2 Tasks and requirements

In this assignment, you are required to write a C++ program that implements an booking system to create and sell travel packages to your potential customers. You will be given a list of customer requirements or randomly generate a list of customer requirements. Your program should try to find a feasible travel package for each customer based on the availability of hotel rooms, game tickets and customer's budget. If you cannot find a feasible package for a customer, indicate that the requirement is rejected. A smart booking system should try to maximize its profit. You may translate your C++ into Java as an option of tasks, but a C++ implementation is mandatory.

You will be provided with a base program, which contains a partial implementation of **CustomerRequirement** class and **Ticket** class, a head file with constant data, and a class driver. You are required to further extend the code to complete the tasks specified in the following.

2.1 Pass level

Your program should include **at least** the following classes:

- a class called **CustomerRequirements** that specifies a single requirement of a customer, containing the information of total budget, desired hotel types and a list of the games that the customer wants to attend (*a partially implemented class has been included in the base code*).
- a class called **Ticket** that specifies the common attributes of a ticket, including the type of ticket, valid date, full price and a discounted price (*an implementation has been included in the base code*).
- three classes: **FlightTicket**, **HotelVoucher** and **GameTicket** that extends the **Ticket** class using inheritance.
- a class called **Package** that specifies a booking package, which contains two flight-tickets, a set of hotel vouchers (one for each day) and a set of game tickets.
- a class called **BookingSystem** that takes a list of customer requirements, makes feasible travel packages for all the customers if possible and reports the outcome.

Requirement of implementation:

- Read the customer requirements from the provided text file and store the data in an array of **CustomerRequirement** objects.
- Output a report of all the feasible packages you produce and the list of rejected requirements into a text file.

As a pass level program,

- You may ignore the information of price discount, the restrictions on hotel rooms and game tickets. Simply assume that all prices are in full and the supply of rooms and game tickets is unlimited.
- However, the total cost of each package should not exceed the customer's budget.

Note: You must implement all the required classes with the same names. However, you can create extra classes, or add more data items and member functions into classes whenever needed no matter for the basic functionalities or any advanced functionalities.

2.2 Credit level

For students who seek credit or above, the calculation of packages will consider all aspects of a package, including *price discounts*, *quota of hotel rooms* and *game tickets*. However, for a credit only code, you do not have to optimize the booking system's profit.

2.3 Distinction level

Students who seek a distinction or above are required to create a class called **requirementCreator** that simulates customer requirements. The class should contain a member function that is able to create randomly 100-150 customer requirements.

Each requireemnt consists of:

- Randomly creates a hotel category either regular, bronze or gold.
- Randomly creates 1-10 different games (at most one for each game).
- Randomly generates a budget:
  - If the games include the opening ceremony, randomly generate the budget between 4500+150*number-of-games to 7500+150*number-of-games.
  - If the games include the closing ceremony, randomly generate the budget between 3800+150*number-of-games to 6800+150*number-of-games.
  - Otherwise, randomly generate the budget between 3250+150*number-of-games to 5250+150*number-of-games.

  The format of each requirement should be:

  budget, *hotelType*, [*game1*,*game2*, …, *game*n]

Download an example of this list from vUWS. Your program should be able to read this file.

2.3.1 High Distinction level

Students who seek for a high distinction are requested to implement a smart booking system which can maximize its profit. Note that customer requirements are not necessarily based on first-in-first serve. The booking system can choose who to offer the packages based on their budgets, desired hotel types, desired game tickets, availability of hotel rooms and game tickets to optimize its profit.

We will also check the quality of code for all the levels.

3. Submission

You need to submit your source code to vUWS for documentation purpose but you will gain your marks and feedback from your demonstration. The code should be purely written by yourself. No part of the code can be written by any other persons or copied from any other source. Submit the following declaration with your code (in a text file or world file).

DECLARATION

I hereby certify that no part of this assignment has been copied from any other student's work or from any other source. No part of the code has been written/produced for me by another person or copied from any other source.

I hold a copy of this assignment that I can produce if the original is lost or damaged.

Both the declaration and source code **must** be submitted to vUWS before the deadline. Your programs (.h, .cpp or .java) can be put in separate files (executable file is not required). All these files should be zipped into one file **with your student id as the zipped file name**. Submission that does not follow the format is not acceptable.

Email submissions are not acceptable.

5. Demonstration

You are required to demonstrate your program during **your scheduled** practical session in Week 13. **You will receive no marks if you fail the demonstration.** Note that it is your responsibility to get the appropriate compilers or IDEs to run your program. **The feedback to your work will be delivered orally during the demonstration.**

The program you demonstrate should be the same as the one you submit except that the comments in your program should be taken off before the demonstration.

The task of this assignment looks complicated when you read it at the first time. Please read this specification at least three times to get a full understanding of the problem. You will enjoy it then ☺.