

OBJECT-ORIENTED SOFTWARE ENGINEERING

CSIE7004 Spring 2023

Homework 1

P76101623 曾中柏

Project Structure

```
P76101623_Hw1
├── build
│   ├── Booking.o
│   ├── Bus.o
│   ├── Passenger.o
│   └── main.o
├── include
│   ├── Booking.h
│   ├── Bus.h
│   └── Passenger.h
├── src
│   ├── Booking.cpp
│   ├── Bus.cpp
│   ├── Passenger.cpp
│   └── main.cpp
├── Makefile
└── main
```

Bus.h

```
#ifndef BUS_H
#define BUS_H
#include <vector>
#include <string>
using namespace std;

class Bus{

private:
    const int busNum;
    // 1-indexed;
    // number assigned according to the order that the bus is
    created

    const int capacity;
    // each bus can accomodate at most 50 passengers

    string departure;
    // bus either departs in the morning or in the evening

    vector<string> passengerList;
    // record the info of passenger that booked the bus

public:

    // constructor
    Bus(int busNumVal, int capacityVal, string departureVal);

    // get function
    // Note: get function should be const
    string getBusDeparture() const;
    vector<string> getPassengerList() const;

    // once a passenger book the bus,
    // add the info of passenger to the passengerList
    void addPassenger(string passengerName);
```

```
    // print the info of passenger that booked the bus  
    void printPassenger();  
};  
  
#endif
```

Bus.cpp

```
#include "Bus.h"
#include <iostream>
using namespace std;

// due to const data member,
// constructor can only be implemented by member initializer
Bus::Bus(int busNumVal, int capacityVal, string departureVal)
    : busNum(busNumVal),
      capacity(capacityVal),
      departure(departureVal)
{
    // empty body
}

string Bus::getBusDeparture() const {
    return departure;
}

void Bus::addPassenger(string passengerName){
    passengerList.push_back(passengerName);
}

void Bus::printPassenger(){
    cout << "Passengers of the " << departure << "-departed bus
are listed below: " << endl;

    for (auto passenger : passengerList){
        cout << passenger << " ";
    }
    cout << endl;
}

vector<string> Bus::getPassengerList() const{
    return passengerList;
};
```

Passenger.h

```
#ifndef PASSENGER_H
#define PASSENGER_H
#include <string>
#include <vector>
using namespace std;

class Passenger{
private:
    string name;
    vector<string> bookingList;
    // record what buses are booked by the passenger

public:

    // constructor
    Passenger(string nameVal);

    // get function
    string getName() const;

    // set function
    void setName(string nameVal);

    // once the passenger books a bus,
    // add the info of the booked bus to bookingList
    void bookBus(string busDeparture);

    // print buses booked by the passenger
    void printBookedBus();

};

#endif
```

Passenger.cpp

```
#include "Passenger.h"
#include <iostream>
#include <string>
using namespace std;

Passenger::Passenger(string nameVal){
    name = nameVal;
}

string Passenger::getName() const {
    return name;
}

void Passenger::setName(string nameVal){
    name = nameVal;
}

void Passenger::bookBus(string busDeparture){
    bookingList.push_back(busDeparture);
}

void Passenger::printBookedBus(){
    // if the passenger has booked a bus
    if (bookingList.size() != 0){
        cout << name << " has booked the following bus:" << endl;
        for (auto bus : bookingList){
            cout << bus << " ";
        }
        cout << endl;
    }
    // the passenger hasn't booked any bus
    else{
        cout << name << " currently has no booking" << endl;
    }
}
```

Booking.h

```
#ifndef BOOKING_H
#define BOOKING_H
#include <vector>
#include <string>
#include "Bus.h"
#include "Passenger.h"
using namespace std;

// class Booking is the booking system,
// which serves as the interface for passenger to book bus ticket
class Booking{

private:
    vector<Bus> busList;
    // record how many Bus objects have been created

    vector<Passenger> passengerList;
    // record how many Passenger objects have been created

public:

    // how many buses currently are there
    int totalBusNum() const;

    // creat a new Bus object
    void createBus(string departureVal);

    // create a new Passenger object
    void createPassenger(string nameVal);

    // passenger books a bus
    void passengerBooksBus(string passengerName, string
busDeparture);

    // print passenger's bus
    void printPassengerBus(string passengerName);
```



```
// // print those passengers who book 2 buses
void printPassengerBookBothBus();

// print bus's passenger
void printBusPassenger(string departure);
};

#endif
```

Booking.cpp

```
#include "Booking.h"
#include "Bus.h"
#include "Passenger.h"
#include <iostream>
#include <algorithm>
using namespace std;

int Booking::totalBusNum()const {
    return static_cast<int>(busList.size());
}

void Booking::createBus(string departureVal){
    int busNumVal {static_cast<int>(busList.size())};
    Bus bus(busNumVal, 50, departureVal);
    busList.push_back(bus);
}

void Booking::createPassenger(string nameVal){
    Passenger passenger(nameVal);
    passengerList.push_back(passenger);
}

void Booking::passengerBooksBus(string passengerName, string
busDeparture){
    for (auto & passenger : passengerList){
        if (passenger.getName() == passengerName){
            for (auto & bus : busList){
                if (bus.getBusDeparture() == busDeparture){
                    passenger.bookBus(bus.getBusDeparture());
                    bus.addPassenger(passenger.getName());
                }
            }
        }
    }
}
```

```
void Booking::printPassengerBus(string passengerName){
    bool found {false};
    for (auto passenger : passengerList){
        if (passenger.getName() == passengerName){
            found = true;
            passenger.printBookedBus();
        }
    }
    if (found == false){
        cout << "There's no such passenger called " <<
passengerName << endl;
    }
    cout << endl;
}

void Booking::printBusPassenger(string departure){
    for (auto bus : busList){
        if (bus.getBusDeparture() == departure){
            bus.printPassenger();
        }
    }
    cout << endl;
}

void Booking::printPassengerBookBothBus(){
    vector<string> result;

    for (int i {0}; i < 2; ++i){
        if (i == 0){
            // copy the passengerList of the first bus to result
            result = busList.at(i).getPassengerList();
        }
        else{
            vector<string> tmp;
            // copy the passengerList of the second bus to result
            vector<string> passengerList =
busList.at(i).getPassengerList();

            for (auto name: passengerList){
```

```
        // if name in the second bus's passengerList is
found in result
        if (find(result.begin(), result.end(), name) !=
result.end()){
            // push_back the name to vector<string>tmp
            tmp.push_back(name);
        }
    }
    result = tmp;
}

cout << "Passengers who booked both buses are listed below:"
<< endl;
for (auto name: result){
    cout << name << " ";
}
cout << endl;
}
```

main.cpp

```
#include <string>
#include <vector>
#include <iostream>
#include "Bus.h"
#include "Booking.h"
#include "Passenger.h"
using namespace std;

int main(){

    // create a Booking object
    Booking booking;

    // create a Bus object that departs in the morning
    booking.createBus("morning");
    // create a Bus object that departs in the evening
    booking.createBus("evening");

    // create 5 Passenger objects with the names of David, Lisa,
    Amanda, John, and Joe, respectively
    booking.createPassenger("David");
    booking.createPassenger("Lisa");
    booking.createPassenger("Amanda");
    booking.createPassenger("John");
    booking.createPassenger("Joe");

    // Passengers book buses
    booking.passengerBooksBus("David", "morning");
    booking.passengerBooksBus("David", "evening");
    booking.passengerBooksBus("Lisa", "evening");
    booking.passengerBooksBus("Amanda", "morning");
    booking.passengerBooksBus("Amanda", "evening");
    booking.passengerBooksBus("John", "morning");

    // print the bus booking info of each passenger
```

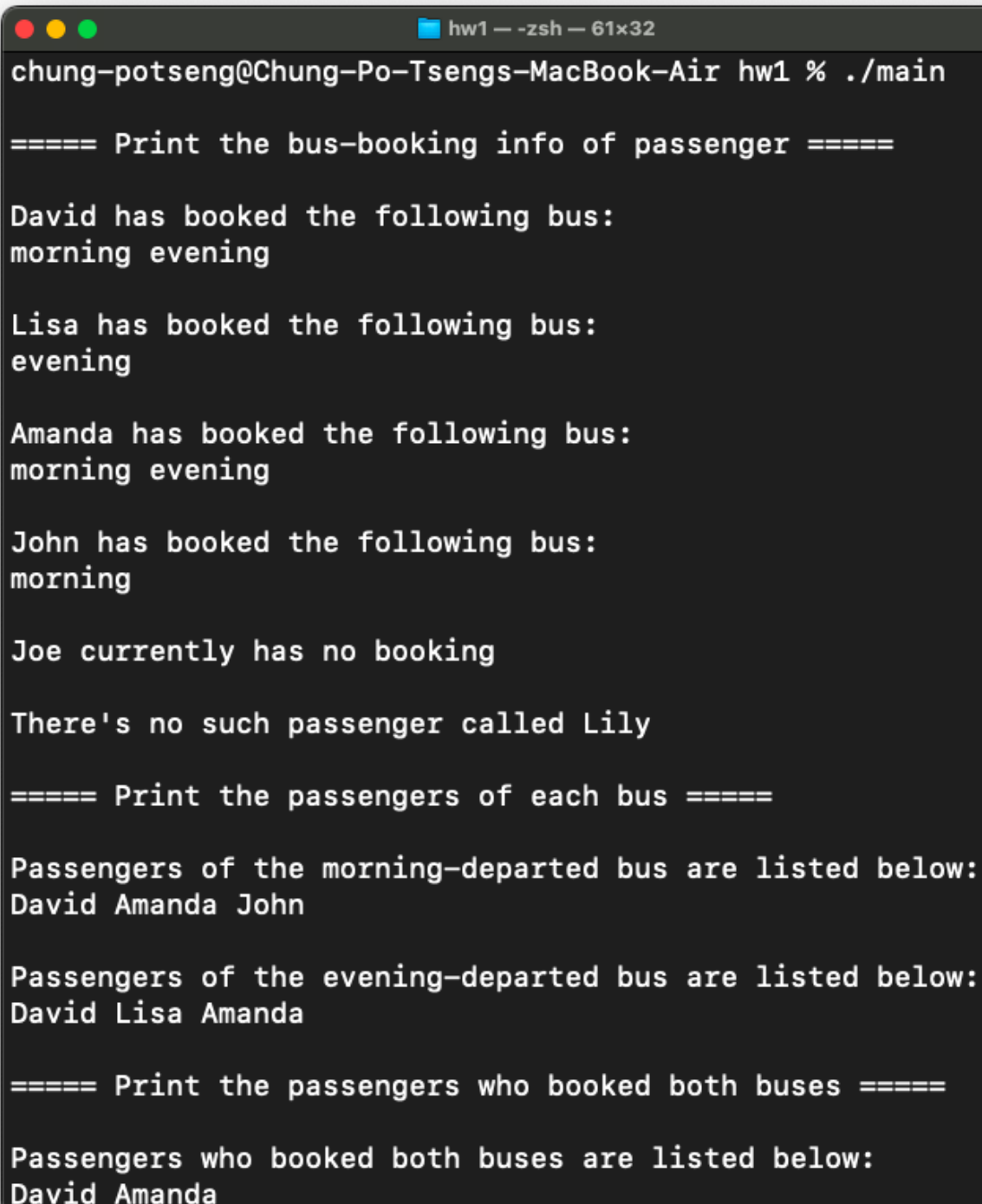
```
    cout << "\n==== Print the bus-booking info of passenger\n" << endl;
    booking.printPassengerBus("David");
    booking.printPassengerBus("Lisa");
    booking.printPassengerBus("Amanda");
    booking.printPassengerBus("John");
    booking.printPassengerBus("Joe");
    // Note: Joe hasn't booked any bus
    booking.printPassengerBus("Lily");
    // Note: there's no registered passenger called Lily

    // print the passengers of each bus
    cout << "==== Print the passengers of each bus\n" << endl;
    booking.printBusPassenger("morning");
    booking.printBusPassenger("evening");

    // print the passengers who booked both buses
    cout << "==== Print the passengers who booked both buses\n" << endl;
    booking.printPassengerBookBothBus();

    return 0;
}
```

Result of main.cpp



A terminal window titled "hw1 — zsh — 61x32" showing the execution of a program. The user runs `./main` and the program outputs the following information:

```
chung-potseng@Chung-Po-Tsengs-MacBook-Air hw1 % ./main

===== Print the bus-booking info of passenger =====

David has booked the following bus:
morning evening

Lisa has booked the following bus:
evening

Amanda has booked the following bus:
morning evening

John has booked the following bus:
morning

Joe currently has no booking

There's no such passenger called Lily

===== Print the passengers of each bus =====

Passengers of the morning-departed bus are listed below:
David Amanda John

Passengers of the evening-departed bus are listed below:
David Lisa Amanda

===== Print the passengers who booked both buses =====

Passengers who booked both buses are listed below:
David Amanda
```