

**PROGRAMMING TECHNIQUE II**

(SCSJ1023)

**SEMESTER 2 2017/2018**

GROUP PROJECT

ONLINE BUS TICKET RESERVATION

SITI HAJAR ASYIKIN BINTI ISHAK ( B17CS3029 )

NADIA ASYILAH BINTI ALI ( B17CS3017 )

TASSVINI A/P GUNASEHARAN ( B17CS3030 )

NURALIEA BINTI ZAINUN (A17CS0141)

2 SCSR

**SECTION 13**

**Lecturer:**

**Dr. Mohamad Ashari Bin Alias**

CONTENT

|  |  |  |
| --- | --- | --- |
| **NO.** |  | **PAGE** |
|  | CONTENT…………………………………………………………… | **2** |
|  | ACKNOWLEDGEMENT……………………………………………. | **3** |
| **1.0** | PROJECT DESCRIPTION………………………………………… | **4** |
|  | OBJECTIVE…………………………………………………………. | **5** |
|  | PROBLEM STATEMENT………………………………………….. | **5-6** |
|  | FLOWCHART……………………………………………………….. | **7** |
|  | 1.0.1 OBJECT ORIENTED CONCEPTS…………………………. | **8-15** |
| **2.0** | CLASS DIAGRAMS AND CLASS DISCRIPTIONS……………… | **16-17** |
|  | 2.0.1 CLASS DIAGRAMS…………………………………………... | **16** |
|  | 2.0.2 CLASS DISCRIPTIONS……………………………………… | **17** |
| **3.0** | USER MANUAL AND SOURCE CODE.………………………….. | **18-30** |
|  | 3.0.1 USER MANUAL.……………………………………………… | **18-21** |
|  | 3.0.1 SOURCE CODE………………………………………………. | **22-30** |

# **ACKNOWLEDGEMENT**

*First of all, we would like to say thank you to our lecturer, Dr. Mohamad Ashari bin Alias for providing guidance for us to complete this project. Besides providing guidance, she also reminding us the deadline of this project so that we will always be aware of the deadline and complete the project on time. Besides that, she also advises us to complete the project as soon as possible so that we can concentrate on our revision for our final exam.*

*Secondly, we would like to thank our family members especially our parents for supporting us all the time. Whenever we are in trouble, they would give us support no matter in physical form or mental form. We could have not completed this project if there is no any support from them. Although they do not have any knowledge in digital logic, but their caring words and advice is the key for us to keep moving forward when we are facing difficulties.*

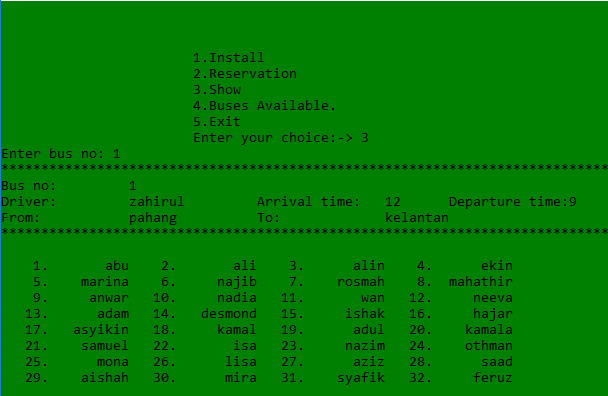
*Lastly, we would like to express our gratitude towards our friends which are also our course mates. They are so helpful and caring that they would spend their time to explain anything we do not understand while completing this project. Furthermore, they also suggest some ideas for us to complete the project. They always guide us when we are having trouble during the survey and the project. They would pin point our errors and guide us to correct it.*

# 1.0 : PROJECT DESCRIPTION

## 

## SYNOPSIS

We are given a few months to prepare this project and the background for this project is about Online Bus Ticket Reservation. Our groups member which is Siti Hajar Asyikin Binti Ishak, Nadia Asyilah Binti Ali, Tassvini A/P Gunaseharan and Noraliea Binti Zainun has create the online bus ticket reservation using application Dev C++ . The purpose for this online bus ticket reservation is to computerize traveling company to manage many data, that all transactions become fast and easy, there should not be any error in transaction calculation mistakes, bill, seat and bus available and other things. It replaces paper work. It keeps records of all bills also, giving to ensure 100% successful implementation of the computerized Bus Ticket Online Reservation.For this project,we used some object-oriented concept such as classes and object, multidimensional arrays, data hiding, aggregation and composition and encapsulation.



**Figure 1 Example of the output of our Project**

## OBJECTIVE

## Objective:-

* To ensure the Customers don’t have to leave the confines of their comfort to book a ticket, and to help them get a ticket when they need it the most.
* The internet was being voted as a medium people couldn’t do without. PC and net penetration was increasing not only in urban areas, but also in rural Malaysia. Also, people were getting used to booking tickets for travel using private airlines websites. So why not buses can be like private airlines websites.

## PROBLEM STATMENT

System that are using by the staff at the counter currently is an internal system and just used to sell the bus ticket at the counter. Customer has to go to the counter to buy the ticket or ask for bus schedule. Furthermore, customers need to pay cash when they buy the bus ticket and sometimes needs to queue up long time to get the bus ticket. Besides that, customer also not allowed to buy bus ticket through phone calls and the bus company’s telephone always-busy line.

Bus Reservation Systems that were suggested till now, are not up to the desired level. There is no single system which automates all the process. In order to build the system, all the processes in the business should be studied, System study helps us under the problem and needs of the application. System study aims at establishing requests for the system to be acquired, development and installed. It involves studying and analysing the ways of an organization currently processing the data to produce information. Analysing the problem thoroughly forms the vital part of the system study. In system analysis, prevailing situation of problem is carefully examined by breaking them into sub problems. Problematic areas are identified and information is collected. Data gathering is essential to any analysis of requests. It is necessary that this analysis familiarizes the designer with objectives, activities and the function of the organization in which the system is to be implemented.

1.Existing system

-Existing system is totally on book and thus a great amount of manual work has to be done. The amount of manual work increases exponentially with increase in services.

-Needs a lot of working staff and extra attention on all the records.

-In existing system, there are various problems like keeping records of items, seats available, prices of per/seat and fixing bill generation on each bill.

-Finding out details regarding any information is very difficult, as the user has to go through all the books manually.

-Major problem was lack of security.

2.Proposed system

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features:

-Needs a lot of working staff and extra attention on all the records.

-Ensure data accuracy.

-Records are efficiently maintained by DBMS.

-DBMS also provides security for the information.

-Any person across the world, having internet can access this service.

-Availability of seats can be enquired very easily.

-Passengers can also cancel their tickets easily.

-Minimum time needed for the various processing

-Better Service

-Minimum time required

-This would help the corporation prepare and organize its schedules more efficiently on the basis of traffic demand.

## FLOWCHART

Exit

Show

Reservation

Bus Available

Insert Data

Install

NO

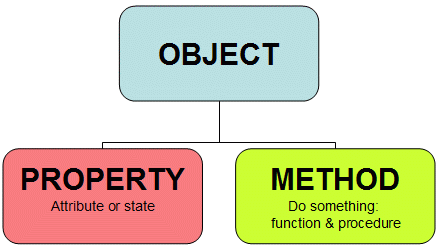
YES

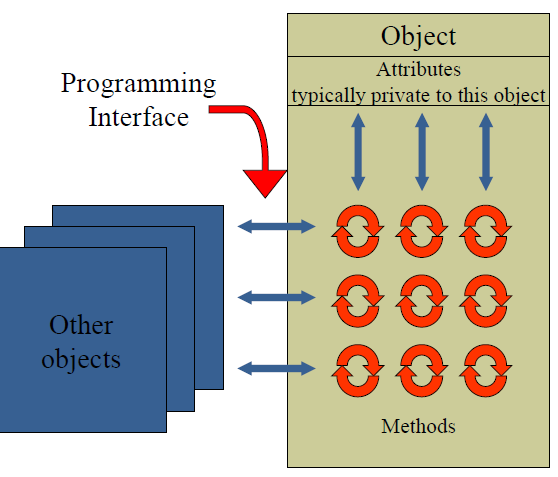
**1.0.1** **OBJECT ORIENTED CONCEPTS**

### OBJECT ORIENTED PROGRAMMING

Object-oriented programming(OOP) is centered on objects rather than procedures functions. Objects are a melding of data and procedure that manipulate that data. Data in an object are known as properties or attributes.

Procedures/functions in an object are known as methods. Object-oriented programming combines data and methods via encapsulation. Data hiding is the ability of an object to hide data from other objects in the program. Only object’s methods should be able to directly manipulate is attributes. Other objects are allowed to manipulate object’s attributes via object’s methods. This indirect access is known as a programming interface.





1. OBJECT

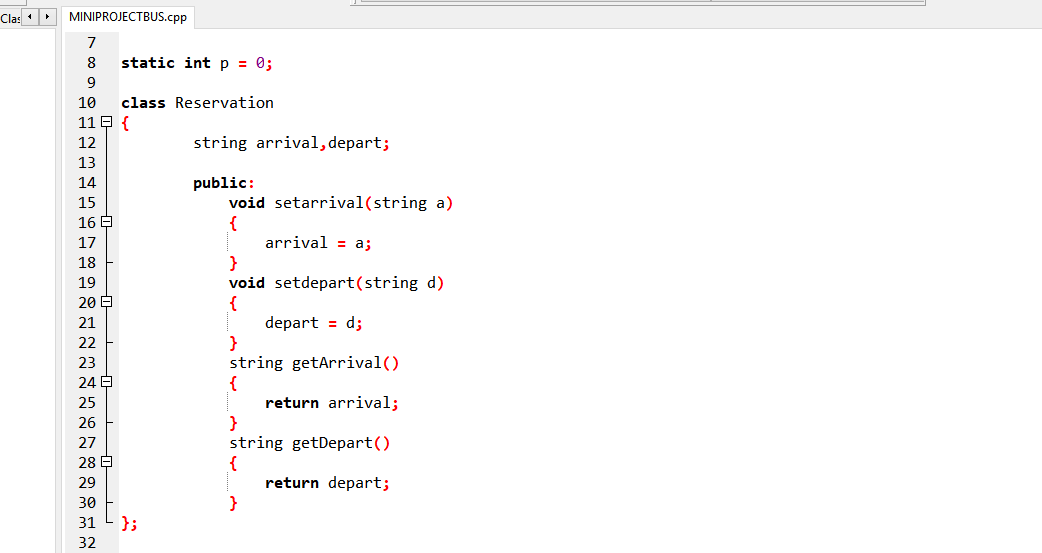
An object can be defined a thing that can a perform a set of related activities. The set of activities is the objects that performs the object’s behaviour. Such as example ,the objects is car which it is moving .

1. CLASS

A class is the template, mould or blueprint from which objects are actually made. It describe the details of an objects. Class are containing a name, attributes and method/operations.

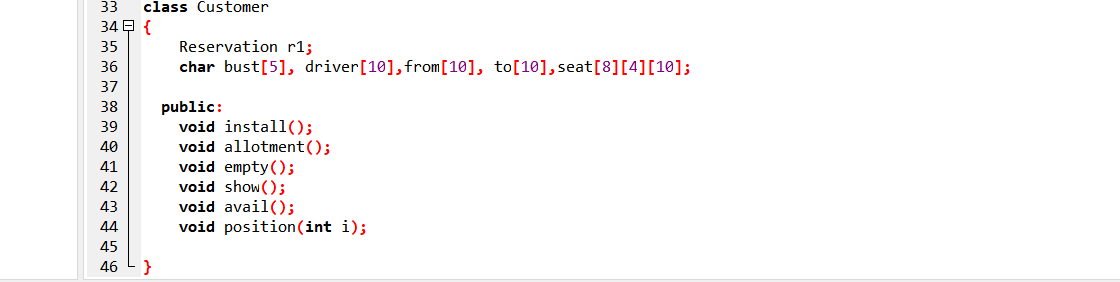


Defining classes and object.

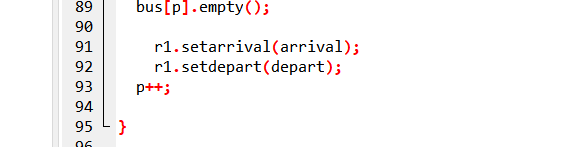


**Object r1**

**Classes**



**Classes**



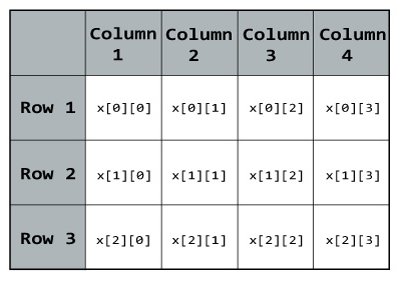
**From Reservation r1, we declare the object r1 to call arrival and depart.**

A program consists of 2 classes. When you have many classes, it needs to be managed. Some technique can be applied to manage classes of your program. In order to manage the classes in the program, we use several techniques, which can be grouped under four main concepts named.

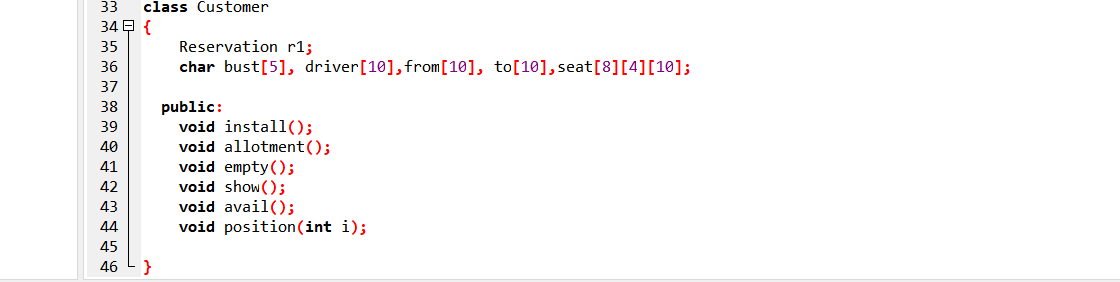
1. Multidimensional Arrays
2. Encapsulation
3. Aggregation and Composition
4. Data Hiding

**MULTIDIMENSIONAL ARRAYS**

Dimensional arrays allows for arrays of two or more dimensions. A two-dimensional (2D) array is an array of arrays. A three-dimensional (3D) array is an array of arrays of arrays.Dimensional programming an array can have two, three, or even ten or more dimensions. The maximum dimensions can have depends on which compiler is being used. More dimensions in an array means more data be held, but also means greater difficulty in managing and understanding arrays. We used (3D) three-dimensional array for seat number. We used (3D) because we should separate seat for the system.



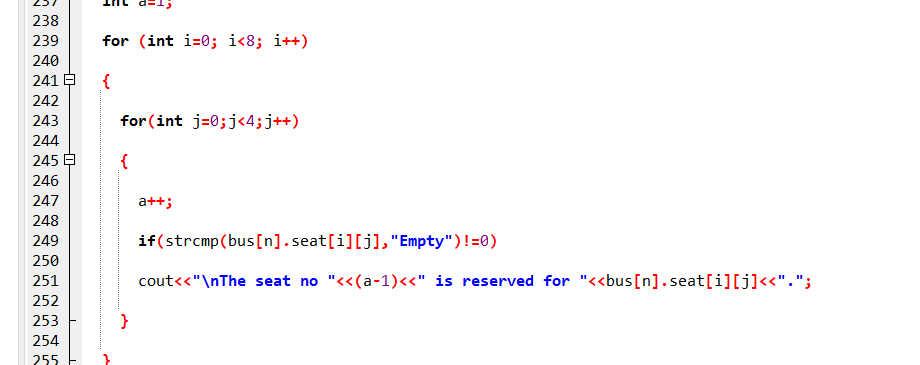
**Use this (3D) three-dimensional array for guide this code: seat[8][4][10]**

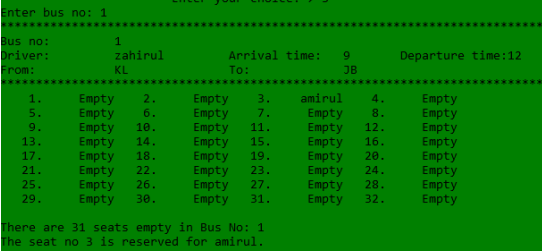


**3D(THREE-**

**DIMENSIONAL ARRAY)**

**Define three-dimensional array and loop.**

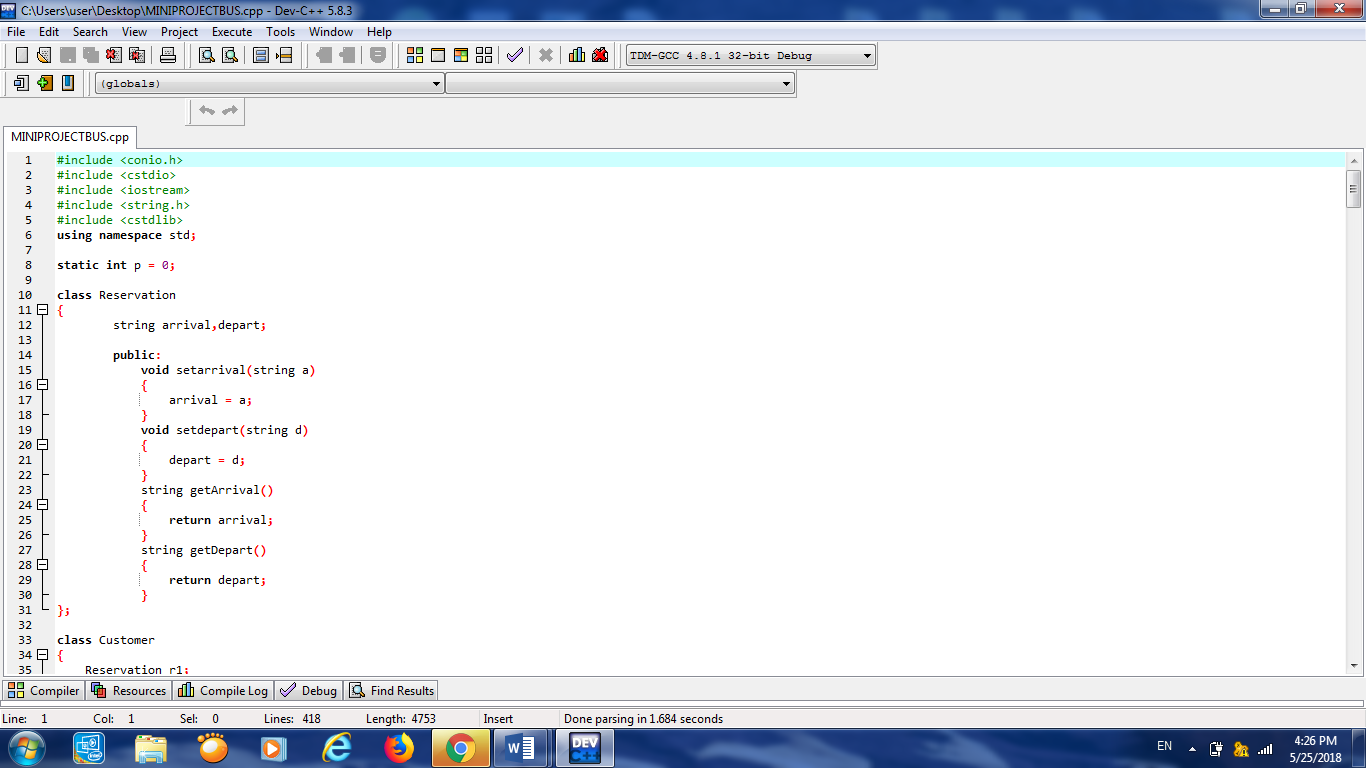




Example output (3D)three-dimensional arrays.

**ENCAPSULATION**

Encapsulation is an Object Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse. Data encapsulation led to the important OOP concept of **data hiding**. **Data encapsulation** is a mechanism of bundling the data, and the functions that use them and data abstraction is a mechanism of exposing only the interfaces and hiding the implementation details from the user. C++ supports the properties of encapsulation and data hiding through the creation of user-defined types, called **classes**. We already have studied that a class can contain **private, protected**and **public** members. By default, all items defined in a class are private.



The variables arrival and depart in Line 12 are **private**. This means that they can be accessed only by other members of the **Reservation** class, and not by any other part of this program. This is one way encapsulation is achieved. To make parts of a class **public**, we must declare them after the **public** keyword. All variables or functions defined after the public specifier are accessible by all other functions in this program.

**COMPOSITION**

Composition is a special form of aggregation, which represents an exclusive ownership relationship between two classes. It models a whole or a part relationship. The part object will be created inside the composed or whole class. Besides that, in composition when an object is destroyed , other objects belonging to it will be destroyed as well and finally The UML Diagram uses a solid diamond attached to the end of association line to indicate composition.

Line 12 in Program Bus Reservation System, the **Customer** class has a **Reservation** object as member variables. The object is used as attributes of the **Customer** class. In this program, the Customer class is a compose class because an instance of it is made of constituent object (line 60,97,176,195,267,331,366 and 386).

While object composition models has-a type relationships (a body has-a heart, a function depart), we can be more precise and say that composition model “part-of” relationships (a heart is part-of a body, a arrival is part of a Reservation). Composition is often used to model physical relationships, where one object is physically contained inside another.

The parts of a composition can be singular or multiplicative. For example, a heart is a singular part of the body, but a body contains 10 fingers (which could be modelled as an array).

**Implementing compositions**

Compositions are one of the easiest relationship types to implement in C++. They are typically created as structs or classes with normal data members because there data members exist directly as part of the struct or class, their lifetimes are bound to that of the class instance itself.

Compositions that need to do dynamic allocation or deallocation may be implemented using pointer data members. In this case, the composition should be responsible for doing all necessary memory management itself.

In general, if you can design a class using composition, you should design a class using composition. Classes designed using composition are straightforward, flexible and robust (in that they clean up after themselves nicely).

**DATA HIDING**

**Definition - What does Data Hiding mean?**

Data hiding is a software development technique specifically used in object-oriented programming (OOP) to hide internal object details (data members). Data hiding ensures exclusive data access to class members and protects object integrity by preventing unintended or intended changes.

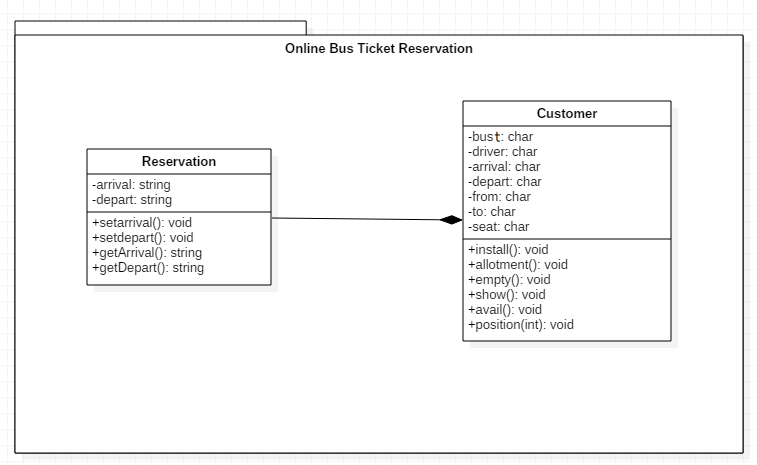
Data hiding also reduces system complexity for increased robustness by limiting interdependencies between software components.

Data hiding is also known as data encapsulation or information hiding. Data hiding was introduced as part of the OOP methodology, in which a program is segregated into objects with specific data and functions. This technique enhances a programmer’s ability to create classes with unique data sets and functions, avoiding unnecessary penetration from other program classes.

Because software architecture techniques rarely differ, there are few data hiding contradictions. Data hiding only hides class data components, whereas data encapsulation hides class data parts and private methods.

**2.0 CLASS DIAGRAMS AND CLASS DISCRIPTIONS**

2.0.1 CLASS DIAGRAMS



2.0.2 CLASS DESCRIPTION

Class Reservation:

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| arrival | The arrival time of the bus |
| depart | The departure time of the bus |
|  |  |
| **Method** | **Description** |
| Reservation | Constructor for class reservation |
| setarrival | To hold Arrival |
| setdepart | To hold Departure |
| getArrival | To access Arrival |
| getDepart | To access Departure |

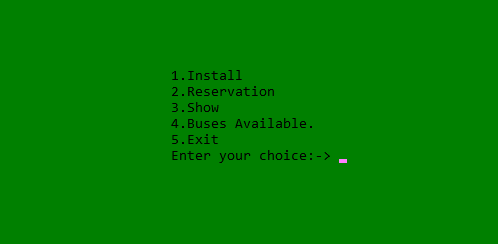
Class Customer:

|  |  |
| --- | --- |
| **Attributes** | **Description** |
| bus\_no | Bus number |
| driver | Driver name |
| arrival | The arrival time of the bus |
| depart | The departure time of the bus |
| from | Bus leaves departure platform |
| to | Bus reach arrival platform |
| seat |  |
|  |  |
| **Method** | **Description** |
| Customer | Constructor for class customer |
| install | To hold Install |
| allotment | To hold Allotment |
| empty | To access Empty |
| show | To access Show |
| avail | To access Avail |
| position | To access Position |

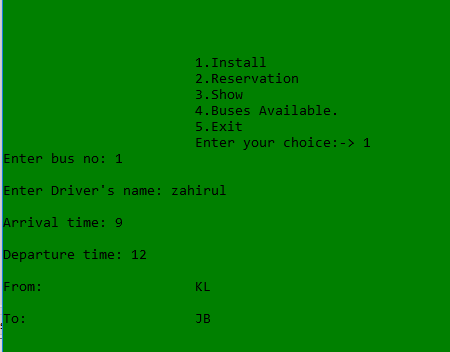
**3.0 USER MANUAL AND SOURCE CODE**

**3.0.1 User manual**

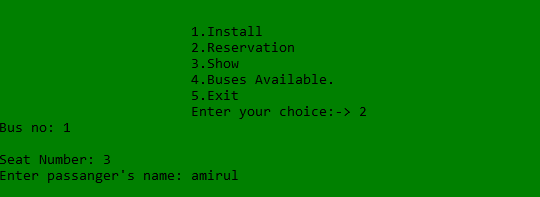
**Main screen:-**

****

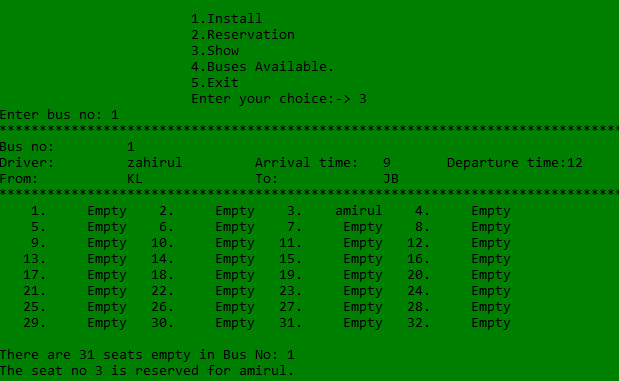
1. This screen is the main interface for our project. The purpose of this screen is to determine whether the user is a customer. The user has to enter ‘1’ to install.



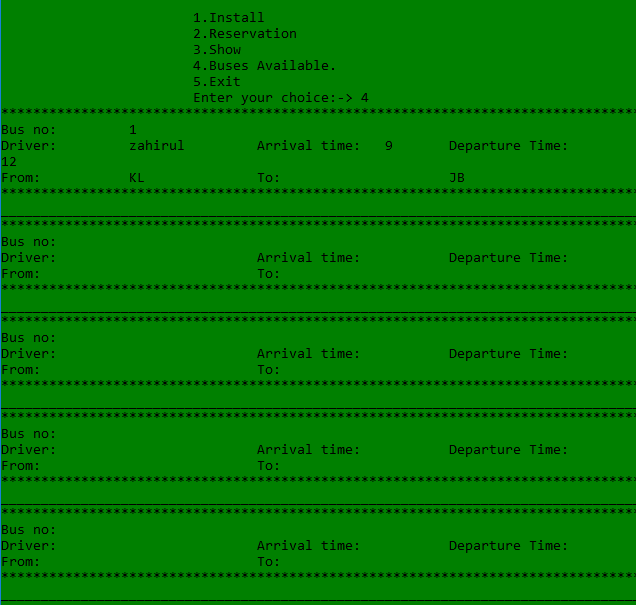
1. Customer need to enter bus number, driver’s name, arrival time, departure time, and destination. They can choose any bus on the system.



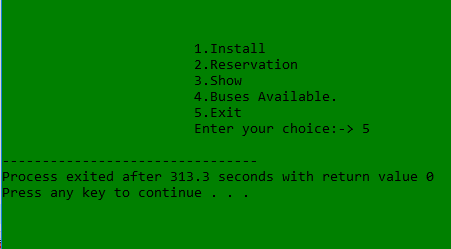
1. Next, customer need to enter ‘2’ for reservation. Customer should enter bus number which they choose by them, customer can choose seat number available and customer should enter their name.



1. After reserve the seat number, they can see results of the reservation. They should enter ‘3’show and enter bus number.



1. They can check buses available or not. Customer should enter ‘4’.



1. After done reserve, customer can exit from the system. Enter ‘5’ for exit.

**3.0.2 Source code**

|  |
| --- |
| #include <conio.h>  #include <cstdio>  #include <iostream>  #include <string.h>  #include <cstdlib>  using namespace std;  static int p = 0;  class Reservation  {  string arrival,depart;    public:  void setarrival(string a)  {  arrival = a;  }  void setdepart(string d)  {  depart = d;  }  string getArrival()  {  return arrival;  }  string getDepart()  {  return depart;  }  };  class Customer  {  Reservation r1;  char bust[5], driver[10],from[10], to[10],seat[8][4][10];  public:  void install();  void allotment();  void empty();  void show();  void avail();  void position(int i);  }  bus[10];  void vline(char ch)  {  for (int i=80;i>0;i--)  cout<<ch;  }  void Customer::install()  {  string arrival,depart;  cout<<"Enter bus no: ";  cin>>bus[p].bust;  cout<<"\nEnter Driver's name: ";  cin>>bus[p].driver;  cout<<"\nArrival time: ";  cin>>arrival;  cout<<"\nDeparture time: ";  cin>>depart;  cout<<"\nFrom: \t\t\t";  cin>>bus[p].from;  cout<<"\nTo: \t\t\t";  cin>>bus[p].to;  bus[p].empty();  r1.setarrival(arrival);  r1.setdepart(depart);  p++;  }  void Customer::allotment()  {  int seat;  char number[5];  top:  cout<<"Bus no: ";  cin>>number;  int n;  for(n=0;n<=p;n++)  {  if(strcmp(bus[n].bust, number)==0)  break;  }  while(n<=p)  {  cout<<"\nSeat Number: ";  cin>>seat;  if(seat>32)  {  cout<<"\nThere are only 32 seats available in this bus.";  }  else  {  if (strcmp(bus[n].seat[seat/4][(seat%4)-1], "Empty")==0)  {  cout<<"Enter passanger's name: ";  cin>>bus[n].seat[seat/4][(seat%4)-1];  break;  }  else  cout<<"The seat no. is already reserved.\n";  }  }  if(n>p)  {  cout<<"Enter correct bus no.\n";  goto top;  }  }  void Customer::empty()  {  for(int i=0; i<8;i++)  {  for(int j=0;j<4;j++)  {  strcpy(bus[p].seat[i][j], "Empty");  }  }  }  void Customer::show()  {  int n;  char number[5];  cout<<"Enter bus no: ";  cin>>number;  for(n=0;n<=p;n++)  {  if(strcmp(bus[n].bust, number)==0)  break;  }  while(n<=p)  {  vline('\*');  cout<<"Bus no: \t"<<bus[n].bust  <<"\nDriver: \t"<<bus[n].driver<<"\t\tArrival time: \t"  <<r1.getArrival()<<"\tDeparture time:"<<r1.getDepart()  <<"\nFrom: \t\t"<<bus[n].from<<"\t\tTo: \t\t"<<  bus[n].to<<"\n";  vline('\*');  bus[0].position(n);  int a=1;  for (int i=0; i<8; i++)  {  for(int j=0;j<4;j++)  {  a++;  if(strcmp(bus[n].seat[i][j],"Empty")!=0)  cout<<"\nThe seat no "<<(a-1)<<" is reserved for "<<bus[n].seat[i][j]<<".";  }  }  break;  }  if(n>p)  cout<<"Enter correct bus no: ";  }  void Customer::position(int l)  {  int s=0;p=0;  for (int i =0; i<8;i++)  {  cout<<"\n";  for (int j = 0;j<4; j++)  {  s++;  if(strcmp(bus[l].seat[i][j], "Empty")==0)  {  cout.width(5);  cout.fill(' ');  cout<<s<<".";  cout.width(10);  cout.fill(' ');  cout<<bus[l].seat[i][j];  p++;  }  else  {  cout.width(5);  cout.fill(' ');  cout<<s<<".";  cout.width(10);  cout.fill(' ');  cout<<bus[l].seat[i][j];  }  }  }  cout<<"\n\nThere are "<<p<<" seats empty in Bus No: "<<bus[l].bust;  }  void Customer::avail()  {  for(int n=0;n<p;n++)  {  vline('\*');  cout<<"Bus no: \t"<<bus[n].bust<<"\nDriver: \t"<<bus[n].driver  <<"\t\tArrival time: \t"<<r1.getArrival()<<"\tDeparture Time: \t"  <<r1.getDepart()<<"\nFrom: \t\t"<<bus[n].from<<"\t\tTo: \t\t\t"  <<bus[n].to<<"\n";  vline('\*');  vline('\_');  }  }  int main()  {  system("cls");  int w;  while(1)  {  cout<<"\n\n\n\n\n";  cout<<"\t\t\t1.Install\n\t\t\t"  <<"2.Reservation\n\t\t\t"  <<"3.Show\n\t\t\t"  <<"4.Buses Available. \n\t\t\t"  <<"5.Exit";  cout<<"\n\t\t\tEnter your choice:-> ";  cin>>w;  switch(w)  {  case 1: bus[p].install();  break;  case 2: bus[p].allotment();  break;  case 3: bus[0].show();  break;  case 4: bus[0].avail();  break;  case 5: exit(0);  }    }  return 0;  } |