Library Copy

DESIGN AND IMPLEMENTATION OF AN OBJECT-ORIENTED GRAPHICAL USER INTERFACE FOR A HOSPITAL ORGANISATION

Dissertation submitted to the Jawaharlal Nehru University in partial fulfilment of the requirements for the award of the Degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE & TECHNOLOGY

by

K.MURALI KRISHNA

[illegible]

[illegible]

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI 110067
INDIA

JANUARY 1995

CERTIFICATE

This is to certify that the dissertation entitled **DESIGN AND IMPLEMENTATION OF AN OBJECT-ORIENTED GRAPHICAL USER INTERFACE FOR HOSPITAL ORGANISATION"** , being submitted by **K.MURALI KRISHNA** to Jawaharlal Nehru University, New Delhi in partial fulfilment of the requirements for the award of the degree of **Master of Technology** in Computer Science and Technology is a record of the original work done by him under the supervision of **PROF K.K.NAMBIAR** , School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi during the year 1994, monsoon semester.

The results reported in this dissertation have not been submitted in part or full to any other university or Institution for the award of any degree or diploma.

[illegible] 16-1-95

Prof. K.K. Bharadwaj
Dean,
School Of Computer &
Systems Scinces
Jawaharlal Nehru University
New Delhi - (INDIA)
Pin 110067.

[illegible]

Prof. K.K. Nambiar,
School Of Computer&
Systems Scinces,
J.N.U,NewDelhi.

ACKNOWLEDGEMENTS

I wish to express my sincere thanks and gratitude to Prof. K.K. Nambiar for his invaluable guidance and the enthusiasm with which he helped me during the course of this project work.

I would like to thank Prof. K.K. Bharadwaj, Dean of School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, for making the various facilities available.

I am grateful to other members of teaching and non-teaching staff who helped in all measures to make this project a success.

(K.MURALI KRISHNA)

<u>**CONTENTS**</u>

CHAPTER 1

INTRODUCTION

The development of a small prototype of an object oriented graphical user interface for a hospital organisation is discussed in detail in this thesis. As the state of the art of computing matured, users are preferring programs, which are user friendly and simple to use. As a result of this graphical user interfaces are becoming popular.

Windows environment is called graphical user interface because windows uses graphics to organize workspace and present user with intuitive ways to accomplish tasks.

Window interfacing has many advantages over character based interfacing in which user gives command and system responds accordingly. In character interfacing user has to memorize sequences of keystrokes and inputs to accomplish certain task. Usage of pointing devices like mouse simplified activities further.

Windows environment provides mechanism not policy. By providing in built functions to create and manipulate windows objects (like menus, dialog boxes, scroll bars, list boxes, combo boxes, radio buttons, group boxes) it helps designer in creation of intuitive, consistent, user friendly user interface.

There are several windowing systems such as Microsoft windows, sun Micro Systems News, Dec's Dec windows and X windows. But Microsoft windows which is compatible for MS DOS operating system changed landscape of user interfaces for personal computers.

GUI's allow user, to interact not only with keyboard but also with pointing device like mouse. GUI's provide an application programming interface that allows user to

(i) Create screen objects

(ii) Draw screen objects

(iii) Monitor mouse activations

The Advantages of Windows

(i) User need not set system for each and every application. Once user sets the system then every application can take advantage of it.

(ii) More than one applications can run simultaneously.

(iii) There can be Data exchange between different application.

(iv) For user interface designers the availability of built in graphical functions enable them to create more intuitive user interface with less effort.

IN MS WINDOWS Window is the primary input and output device. Windows environment provides facilities and application uses them to design GUI.

Windows objects like dialog boxes, menus, list boxes, scroll bars, comboboxes, radio buttons, check boxes, group boxes to change look and feel of window intuitively for particular task. Window appearance is the result of windows environment and application programs collaborative effort.

By using object oriented approach, the burden further reduces for software developer of GUI. Because to create a simple window which does not contain any other screen objects requires several pages of code.

The advantage of object oriented programming in this situation is that, by modifying parts of a object or adding few features new object can be created. There is a natural fit between windows programming and object oriented programming. Windows defines everything on the screen as a window. Each window inherits some of its features from other windows. The data that goes with a window is encapsulated into the window object. Window objects are polymorphic. They can all receive common messages and take action that is appropriate for that window[4].

Another way to make windows programming easier is to use a commercial library of routines that handle windows. Developer can create window objects extending the routines in OWL.

The prototype for hospital organisation is implemented in Microsoft Windows using Borland's object windows eases windows applications development by providing.

(i) A consistent intuitive and simplified interface to windows.

(ii) Supplied behaviour for window management and message processing.

(iii) Basic frame work for structuring a windows application.

In this project

I took a small prototype for Real World Hospital Organisation. The functions included in this

(i) ADMISSION OF PATIENTS.

(ii) DISCHARGING AND BILLING OF PATIENTS.

(iii) VIEWING DETAILS OF WARD AND BED NUMBER OF PATIENTS.

(iv) VIEWING ADDRESS OF A PATIENT.

Due to limitation of time I was forced to take up implementation of prototype for real world system rather than real world system.

In Chapter 2 the importance of user interfaces in present situation is discussed.

In Chapter 3 the significance of hospital automation system and its advantages are discussed.

Chapter 4 gives introduction of object oriented approach and its concepts.

Chapter 5 gives discusses about object windows

Chapter 6 contain design and implementation of My Project.

CHAPTER 2

USER INTERFACE

2.1 INTRODUCTION

User interface can be defined as user's interaction with computer for accomplishment of tasks. It comprises user's commands and system's responses in accordance with user's commands. User interface becomes inevitable, where system needs commands from user to accomplish tasks. As the state of the art of computer technology matured, quality of user interfaces is increasing.

Earlier users were considered as a monolithic, homogeneous group, differentiated primarily, by discipline or task. As the number and varieties of users is increasing and as devices becoming more elaborate in functions and more voluminous in data, the demand for more informative and visually appealing user interfaces is growing. As a result of it quality of user interfacing became one of the factors to decide market value of product. In some cases code devoted for user interface is exceeding the code devoted for core program. At present we have user interfaces in which user can work on application by selecting and moving icons with pointing device such as mouse.

But there should be a trade off between quality of user interface and effort and time to be devoted for creation of user interface. Otherwise the effort and time spared on user interface creation may adversely affect cost effectivity. should be able to create an intuitive, visually appealing and user friendly user interface with least possible effort.

2.2 Characteristics of Good user interface

SIMPLE TO USE :- User interface should be understood by users and they should be able to use it with ease. By using pointing device user can select option easily than by using key board. Menu driven user interface makes easy to user to understand it rather than character based user interface.

USER CONTROL :- An user interface which makes user feel like controlling events will be preferred by him. This feature commonly found window interfacing. In windows user can work on application by clicking on icons and moving them.

CONSISTENCY :- User interfaces should be consistent. That means all user interfaces, should have same look and feel. It enables the user to switch between applications. After using one application it should not be difficult for using another application.

COST EFFECTIVENESS:- Creation to user interface should not effect cost adversely, that is possible when user interface can be created with least effort. To make it possible we need to take help of in built graphics routines.

We can satisfy above constraints by creating an environment which supports basic graphic routines. Using them we can develop an attractive user interface. MS windows which is compatible for PC's is such an environment.

By extending object oriented approach to windows we can further ease the burden to develop an user interface. Inheritance is an feature of object oriented approach which allows us to create new classes from existing classes. In object windows library of In built routines, which define standard window objects, from those objects we can create our own objects using inheritance. By manipulating them we can create user interface according to our application.

CHAPTER 3

HOSPITAL AUTOMATION SYSTEM

3.1 INTRODUCTION

The task to manage a hospital becomes complex when the number of patients become large hence most of the hospitals are not able to provide efficient and timely services. To meet the demands of health care there is a need to improve the efficiency of all their functions. Modern techniques should be introduced to overcome shortcomings in hospitals. There could be various forms of shortcomings such as

(i) Bed availability not known accurately at an instant of time.

(ii) Loss of revenue as department bills are not tallied with patient bills daily.

(iii) Absence of upto date information.

(iv) Delay in preparing patient bills causing in convenience to patients.

(v) Drug getting time barred.

(vi) Shortage and overstocking of medicines, reagents and other spares.

(vii) Consumption of lot of time and effort to collect and prepare statistical data.

(viii) lack of information to answer queries.

(ix) dependency on very few persons.

For organizing a hospital without the shortcomings above given, we have to manage complex flow of patient and administrative information carefully. Quick and easy access to such information is very important. Improvement of information has to be regarded an integral part of a comprehensive strategy, commitment for better management of the hospital and enhancement of quality of service to the patient.

Any good hospital always works under pressure, providing services to more patients with minimum increase in resources. This makes it necessary that worker productivity is increased. A clinical person be, it a nurse or a doctor or a technician should spend time in clinical activity and not in maintaining registers, making workday summaries and running around for an appointment. This can be done with the help of a computer.

Today computerization is a software driven process, where state of art software is chosen and it can be used for our application.

This allows us to save the investment in man hours and training, planning and recognizing activities.

For this objective there is an option

To provide a basic integrated system and providing visually appealing user interface.

By using object oriented approach the effort needed for implementation of this application reduces.

So I implemented this application using object windows library.

3.2 ABOUT MY PROJECT:

As prototype for real world system it covers important functions

ADMISSION, DISCHARGING AND BILLING OF PATIENTS Adding to these

o VIEW functions (Queries)

Queries about patient's WARD, BEDNO, ADDRESS are included.

3.3 Adantages Of Automation System:-

(i) This model suits any Hospital not with respect to its size. It can be used for small dispensary to big multi-speciality hospital having many beds.

(ii) It is user friendly by system: This allows us to save many man hours to train operator before operating. Help functions will contribute for it.

(iii) It prepares bill for patient while he has to be discharged he will be billed for stay in Hospital, Medicines, and miscellaneous charges. Thus there wont be any loss of revenues and no delay in preparing the patient bills.

(iv) It is reliable and dependable system. It is capable of 24 hours non stop operation. This enforces discipline, removes the operator errors and improves efficiency of the hospital management.

CHAPTER 4

OBJECT ORIENTED APPROACH

Object oriented approach offers the potential for significant improvements in the software development process. It is a way of organizing programs. Unlike structured programming approach (which visualizes program as a group of modules each of which will be assigned certain task) object oriented programming approach visualizes program as a group of objects which may resemble real world objects.In object oriented approach data and function which operate on that data encapsulated into a unit called object.

4.1 Object:-

Object is a well defined abstraction of real world entity. Object is a unit which encapsulates data and member functions which operate on the data. Object may be as small as character strings and as large and complex as databases.

Every object contains

Object	Data which is hidden with the object
data	Member function which operate on data
Functions	Message (functional calls) to member functions

Every object will contain its own data although it may share it with other objects. Object's implementation will decide the data it contains and how member functions modify the data. The only way to access the data an object is through calling member functions which operate on that data. Thus data is protected from unwanted variations. It is possible to modify data and member functions without affecting other objects. Thus hiding physical implementation details is called *data encapsulation*.

For single message there can be multiple implementations which is called *overloading*.
(Constructor is example for this)

Selection of code to perform certain task is called *binding*. Sometimes compiler may not recognise code to perform a service. In that case selection of code is done at run time. This is called *late binding*.

This is done by selecting the object to perform a service. If the code is selected in compile time that is called *early binding*.

4.2 CLASS:- Class is template for object.Object is Member of class. Though all the objects in class share executable code and member functions each object will have its own data no other function which is not declared in class cannot use data in class. Thus class can resemble one class of real word objects. Class definition may include a function with same name as class. That function is called *constructor*. It will be executed when an object of that class declared constructor can take arguments but returns no value. Constructor can be overloaded so an object can be created in different ways[7].

Destructor is also member functions which takes no arguments and has no return type. This is executed when object destroyed.

There can be a data item restricted to single instance for all objects of class. This is the case when the data item declared as static[7].

In my module classes TAdmitdlg, TDischargeDlg are examples.

Example program in c++

```
const float MTF=3.291
class distance
{
private:
      int feet;
      float inches;            //data//
public:
```

```
distance()
{feat=0; inches=0.0;}//constructor with no arguments//

distance (float meters)    //constructor with one argument/
distance (float meters)
{float fltfeet=MTF*meters;
feet = int(fltfeet);
inches=12*(fltfeet-feet);

}

void showdist() // MEMBER FUNCTION//

{ Cont<<feet<< :\" <<inches<<:/";}

};

void main()
distance dist1=2.35; //initialization dist')

cout<< "indist1" = ";
dist.showdist();              //message to member function//
```

[7]

4.3 INHERITANCE: Inheritance is the process of creating new classes, called derived classes, from existing or base classes. The derived class inherits all the capabilities of base class and have additional features its own. Base class unchanged in this process.

Data or functions in the base class that are prefaced by the key word protected can be accessed from derrived classes but not by objects of derrived classes. Classes may be publicly or privately derrived from base classes. Objects of publicly derived class can access public members of base class while objects of privitely derrived classes cannot[7].

Example(In Figure 4.1)

Explanation:- In Figure4.1 person is the base class for employee, student classes they inherit all the properties of person class but parts of person class which are prefixed by reserved word private. They can inherit public and protected parts of person class but objects of employee, student classes have access to public parts of person class only.

Again employee class is base class for worker, professor, clerk classes they are inherited from employee and they has their own characteristics too.
Class can inherit from more than one base class that is called *multiple inheritance*.

Example for Multiple inheritance in FIgure4.2

Explanation

Class scientist derrived from class student and class Manager derrived from class student and class employee class worker derrived from class employee.

There can be classes within class this is possible when a class is defined and an object of it is declared in another class.

Inheritance is one of important aspects of object oriented approach. Because of inheritance allows developer create a new class with little modification by making existing class public for new class and adding extra features to it. It is called reusability of code reusing existing code saves money and time and increases programs reliability.

Inheritance also help in original conceptualization of overall design of the problem.

4.4 OPERATOR OVERLOADING: Operator overloading is an important feature of object oriented approach. It can transform complex, obscure program listings into intuitively simple one. In other words it is a type conversion user defined type to basic type, basic type to user defined type etc[7].

4.5 POLYMORPHISM: Is one of major advances of object oriented programming, polymorphism allows the user of an object to invoke standard behavior, without knowing what type of object is used. This allows developer to create families of objects that share some part and has their own features. For polymorphic member functions late binding is done. For non polymorphic functions early binding or static binding is used. Virtual key word is used to specify that a member functions is polymorphic. Polymorphism is used when compiler can't determine the types in advance[11].

For polymorphism to be exhibited these are the conditions to prevail.

(i) base class and derived cases should have member function with same name.

(ii) member function in base class should be declared as virtual function.

A virtual function is called pure virtual function when it has no body.

...

CHAPTER 5

OBJECT WINDOWS

5.1 INTRODUCTION

Object Windows provides an easy alternative way to develop windows applications. Before we develop our application we have to choose required API functions from 600 odd AP1 functions.This makes developing an application in windows environment complicated and complex job. It affects cost of product adversely.

5.2 Advantages Of Object Windows:

Object windows lets the software developer ignore internal details of windows. Using inheritance we can create screen objects according to our intuition from the standard objects which are readily available with object windows library. By manipulating them we can create user interface which suits our applications with much lesser effort. Object windows allows default message processing.

In windows every window was assigned handle (an integer type). Many functions in windows require that handle to do any processing with a window. Software developer has to keep track of each window's handle for doing any processing with window object windows takes away that trouble from software developer. In object windows each window is an object. What ever the data it has is encapsulated in it. For each window its handle is stored as parameter HWindow using that window functions recognize on which window processing to be done.

Object windows allows you to do certain task with fewer function calls. It results in reduction in complexity and effort to do certain task.

Object windows allows developer to ignore default message processing. If a control or a window or a dialogbox ignores certain message by not defining response to it, object windows invokes default message processing by calling def wnd proc function.

5.3 ABOUT OBJECT WINDOWS PROGRAM:-

5.3.1 TApplication CLASS:-

TApplication member function is the prime requisite for object windows program. It includes data members such as the application instance (HInstnace) a pointer to main window object (Main Window), and a pointer to accelerator table (HAccTable). The member functions for TApplication class handle the details of initializing the application and relieving messages from windows. Your program must call the constructor for TApplication to set up the data members and run member function to process windows message[4]. Most applications make a descendent class of TApplication so that they can overload certain member functions to suit the specific need of the application. For example almost every program overloads InitMainWindow function with one that creates TWindowsObject.

Object windows calls TApplication member functions, InitApplication and InitInstance during the call to run function to initialize the application. If that is the first instance of the application, object windows calls InitApplication .

In windows program any function which goes long time should never be executed without checking for messages from windows[4]. In object windows program fast computations can be made in TApplication member function called IdleAction. Object windows calls the function where there are no messages for application.

Just before the exit from application object windows calls the TApplication memberfunction canclose to see it is OK to stop application. If application can close safely, canclose should return true.

5.3.2 TWindow Class:- The other object class found in most object windows programs is a descendent of TWindow. This class represents a window on the screen. Typically a program creates a TWindowObject in the InitMainWindow function the constructor for this class sets up the object variable such as Attr that contain the position and size of the window, the window style, the control ID for control windows and a far pointer that you can use any way you want[4].

5.3.3 Important conventions in object windows:-

The type (classes and instances of classes) prefixed by letter T

Pointers to types by letters PT

and references to types by letters RT

5.4 OBJECT WINDOWS HIERARCHY :-

Object Windows is comprehensive set of classes that simplifies he development of windows programs with C++. You can derive new class from standard classes using inheritance. The object windows hierarchy is shown in figure 5.1.

Object is the base class for all object windows derived classes.

TApplication which defines behaviour of all object windows applications is derived from *TModule* which it self is derived from object[6]. TModule provide support for window memory management and error processing.

TWindowsObject is an abstract class, derived from *Object* and *TStreamable*, that defines the fundamental behaviour shared by all object windows interface objects. Objects representing windows, dialog boxes and controls are called user interface objects or simply interface objects. *TWindow* object provides member functions to handle creation, message processing and destruction of window objects.

TWindow is a general purpose window class which can represent main, popup, or child windows of an application. Usually in object windows applications main window class is derived from TWindow[6].

TDialog serves as a base class for derived classes that mange windows dialog boxes. Dialog objects serve to facilitate interactive groups of controls such as buttons list boxes and scroll bars. They associated with dialog resources. They can be run as model or modeless dialog boxes. Member functions are provided to handle communication between a dialog and its controls.

TEditWindow defines a class that allows text editing in a window. *TFileWindow* derived from *TEditWindow*, defines a class that allows loading and saving text files in addition to text editing in a window. *TFileDialog* defines a dialog that allows the user to choose a file for any purpose, such as opening, editing or saving[6].

TInputDialog derived from *TDialog* defines a dialog box for user input of a single data item.

Control objects such as list boxes, buttons and edit controls defined by windows. Tcontrol is the base class for all control objects and it defines member functions to handle creation and message processing for all control objects[6].

TButton, TCheckBox, TRadioButton, TListBox, TComboBox TStatic, TGroupBox are derived from TControl. TButton Class represents push button interface element in windows. TCheckBox and TRadioButton classes handle creation and state management of check boxes and radio buttons respectively, TListBox handles creation of and selection from windows, list boxes and gives member functions to manipulate items in a list. TGroup Box provides member functions to handle a group of selection boxes (check boxes and radio buttons) or other controls[6]. TStatic provides, member functions that set, query and clear the text of a static control. TEdit is derived from TStatic and it provides extensive text processing capabilities for a windows edit control.

CHAPTER 6

DESIGN AND IMPLEMENTATION

6.1 Hardware and Software Requirements

The Hospital Automation System application has been implemented in MS windows environment using Borland C++ for Windows. The basic hardware requirements for the Hospital Automation system application are the same as those of Object Windows Application.

1. A hard disk
2. 2 MB of memory or more
3. Windows-Compatible graphics display
4. Windows 3.0 or later in 386 enhanced mode

6.2 Functions Included

1. Admission of Patients
2. Discharging and Billing Patients
3. Viewing ward and Bed member of patient
4. Viewing address of patient

6.3 Database Design and Implementation

:Database used for implementation of the user interface is a relational database with relations Patient and Address. Attributes for *patient* relation.

Attribute	Range	
Name	Character string	
iv	Character	(status of patient)
age	Character string	
sex	Character string	
problem	Character string	
splty	Character string	speciality under which problem comes
xray	Character string	for which part of the body
tests	Character string	which tests to be done (blood, urine etc.)
result	Character string	result of the tests
operation	Character string	need of operation
condition	Character string	whether normal or serious
advice	Character string	advice to the patient
mrdno	Character string	medical reference No.

iv = 'i' if he is old patient (once admitted and discharged)

= 'j' if he is present in hospital.

In this relation mrdno forms the key because it alone distinguishes each patient.

Every other attribute depends on mrdno. No other partial of transitive dependencies present.

Address relation:

Attributes of Address relation.

attribute	range	
hno	character string	house number
street	character string	street number
viltwn	character string	village or town
dist	character string	district
state	character string	state
pin	character string	pin code
mrdno	character string	medical reference number

In this relation also mrdno forms the key. Every other attribute fully depends on mrdno.

Old patients details stored in old p.d

Bed status stored in bstatus.d

Patient relation stored in patient.dat file

address relation stored in address.dat file

I used arrays of structures to access the database. Each structure is a node, which contains all the information about patient. (natural join of relations patient and address).

Using index we traverse one node to another in database.

By using index we can retrieve any information about a patient.

6.4 DESIGN AND IMPLEMENTATION OF WINDOW INTERFACE

In window interfacing user selects menus for doing specific tasks and gives input through dialog boxes . This input passes to application program which uses this data and interacts with database and will do appropriate processing and return user output in window or in message box.

The Pseudo code for the message processing and appli cation flow given below. Code takes care of messages related to application only. Other messages will be done by default message processing in object windows.

The input from dialog box is transferred to a structure using transferbuffer member function. That input passed to application program it processes that and interacts with database and output is transferred to output window.

Example:-

DoAdmit function receives data from struct tr of type GroupData.That is used to calculate bed number,mrdno etc.
All the information stored in a node. Patient details (of illness) are stored in patient.dat, address in address.Dat.

No of patients incremented by one. Bed status is updated. Then output window is created. MRDNO, BEDNO will be shown on output window.

Thus this function works.

6.4.1 Pseudo code for TMainWindow Constructor

```
Start
load bed status from bstatus.d
open and read Patient.Dat and Address.Dat
load the information about the patients in array of
 structures.
End.
```

6.4.2 MESSAGE PROCESSING FOR MAIN WINDOW

```
Case selection
menuitem Admit        =>  Execute TAdmitDlg
menuitem Discharge    =>  Execute TDischargeDlg
menuitem Viewbed      =>  Execute TViewbedDlg
menuitem Viewaddress  =>  Execute TViewaddressDIg
menuitem Quit         =>   process close
Default               =>  Default windows processing
End case
End
```

All these message processing can be done only when TMainWindow object is active otherwise no action will be taken.

6.5 Application flow:

```
Start
   . Initialize TApplication object

   . if it is first instance perform InitApplication
     else perform  InitInstance
   . process InitMainwindow
Repeat
      Process Messages for the application until
message=WM_QUIT)
Destroy TMain Window and TApplication objects return status
END.
```

Execute TAdmitDlg

```
Start
Construct TAdmitDlg child objects.
Get input from user for Admiting patient.
case selection
OK Button: if (Valid Input)
              Increment number of patients by 1.
              Assign bed.
              Assign mrdno.
```

```
                Store patient details(of illness) in patient
                file.
               Store address in Address file.
               Update bed status.
               Create output window.
                   Give mrdno, bedno on output window.
case(message)
WM_PAINT:Close TAdmitDlg and return to TMainWindow.
WM_QUIT : Close TAdmit Window
           Destroy Window //MAIN WINDOW
DEFAULT: Default Windows  Processing
End case
else
give error message
End if
End
```

Execute DischargeDlg

```
Start
Execute TDischargeDlg to get mrdno from user
   if (valid mrdno)
if (Patient is old patient)
give error message
else
get patient details  from database
```

```
Deassign bed
Change status of patient (to old patient) in patient.dat
Update bed status.
Calculate days of stay
Calculate total bill (for stay, medical, others)

Create a output window
Give name, age, mrdno, days of stay, bill for stay, medical
bill, others in output window
case message
WM_PAINT :Close TDichargeDlg and return to TMainWindow
WM_QUIT Close TDischargeDlg and return to TMainWindow
DEFAULT:  Default windows processing
End case
End if
else
give error message
End if
End
```

Execute Viewbed

```
start

Execute TViewbedDlg to get mrdno from user

   if (valid mrdno)
if (patient is old patient)
give error message
else
Get bedno, speciality, Name of the patient  from database
Create output window
Output name, bedno, speciality
Case Message
WM_PAINT :Close TViewbedDlg

WM_QUIT  close TViewbedDlg
destroy window
default: Default windows processing
End case
End if
else give error message
End if
End
```

Execute TViewaddressDlg

```
Start
Execute TViewaddressDlg to get mrdno from user
```

```
    if (Valid mrdno)
Get address from address.dat
Create output window
Output address of patient in  output window
case (message)
  WM_PAINT:Close TViewaddressDlg
return to TMainWindow
WM_QUIT: Close TViewaddressDlg
destroy window (Main window)

Default: Default windows processing else give error message.

End if

End
```

Process close

```
start

do
read all  patient's information from array of structure
write patient details in patient.dat address in address.dat
if (patient is old patient)
write all information of that structure in oldp.d
End if
While (i<no of patients)
End.
```

CHAPTER 7

SAMPLE SESSION

The user can run the hospital application from program manager's File\Run menuitem by typing application's path name.The application's Main window shows(in figure 7.1) main menu of the application. The MAIN menuitem provides important functions in hospital organisation(ADMIT,DISCHARGE etc).The view menuitem provides quiries about patients.The QUIT menuitem is used for updating database before closing application.

Figure 7.2 shows pulldown menu of *MAIN* menuitem.it can be seen by user by clicking MAIN menuitem.User can select eiether ADMIT or DISCHARGE by clicking with mouse or by using Up/Down Arrow keys.

Figure 7.3 shows *VIEW* menuitem's pulldown menu.User can select the quiery he want to make.

When user selects *ADMIT* menuitem, dialogbox appears on the screen (shown in figure7.4)which accepts user input(patient details who has to be admitted) if the input is valid.If it is not valid input it gives error message.

If user input is valid, application responds by creating an output window and medical reference number,bed number alloted to the patient be given out.This is shown in figure7.5.

When user selects *DISCHARGE* menuitem, input dialog appears on screen which accepts medical reference number of the patient to be discharged.In case there is no patient exists in hospital with that mrdno, application gives error message .Otherwise it accepts the input from user.Input dialogbox shown in figure 7.6.

As a response for user input, application creates an output window and gives out patient details and bill to be paid by him in output window.this is shown in fig 7.7.

When user selects *VIEWBED* menuitem input dialogbox appears on screen as shown in fig7.6.If input is valid an output window created and patient's bed number,name,spciality given out in output window.This is shown in fig7.8.

When user selects *VIEWADDRESS* menuitem Input dialogbox appears on screen which accepts input from user. As response to it an output window created and patient address will be given out on that. This is shown in fig7.9.

CHAPTER 8

CONCLUSION

AND

SUGGESTIONS FOR FUTURE EXTENTIONS

In this thesis the design and implementation of Object oriented graphical user interface for hospital organisation is discussed. My module as a prototype for real world system, has some limitations. To make it a real world system some improvements are necessary.

Those are:

(1) My module can be used by single uses only. For a big hospital there need to be number of terminals using them, operators coordinating the daily operations in Hospital. For that purpose these is need of a network which connects all the terminals to make smooth operations possible.

(2) In real world system protection from unauthorized users is one of the essential features. Password to be provided for user so that others cannot access the information.

(3) My module support important functions in hospital organisation but they wont be sufficient for real world system. There is a need of extending functions and database extention inaccordance with functions.

(4) For big hospitals there will be lot of data to be searched to do one function. There should be efficient way of searching to minimize search time. This can be an improvement in case of big hospitals.

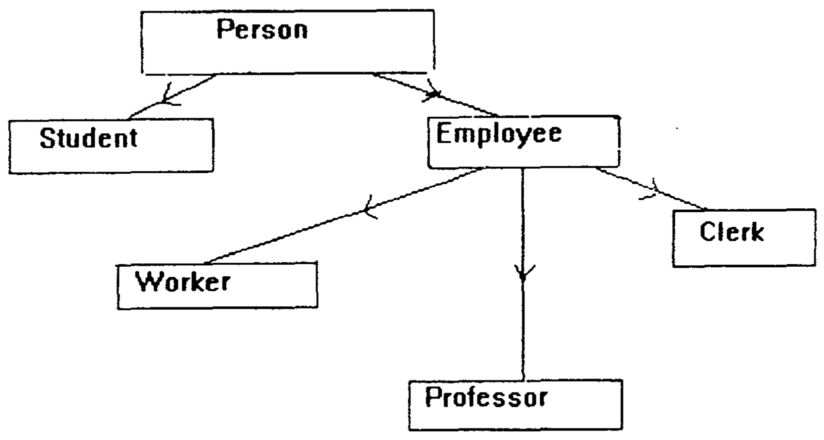
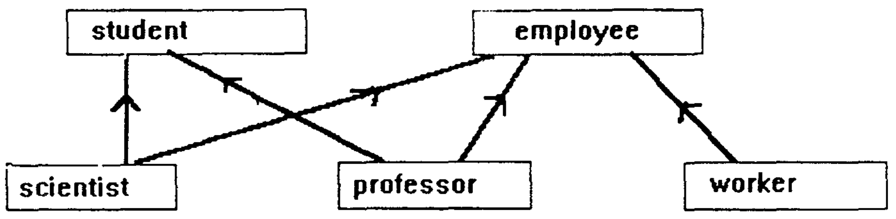
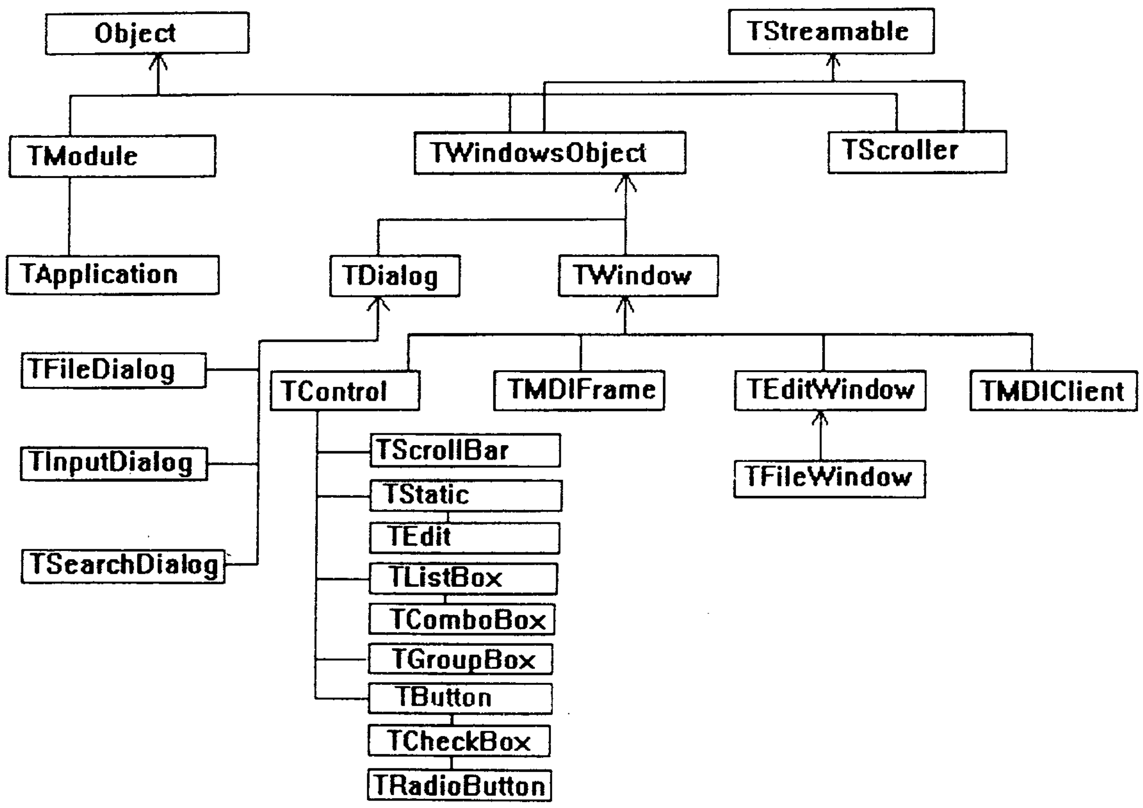


Fig 4.1 Class Hierarchy for a college



From ref 7 Example for Multiple inheritence

Fig 4.2



(FROM OBJECT WINDOWS FOR C++ USER'S GUIDE p 224)

OBJECT WINDOWS HIERARCHY FIG 5.1

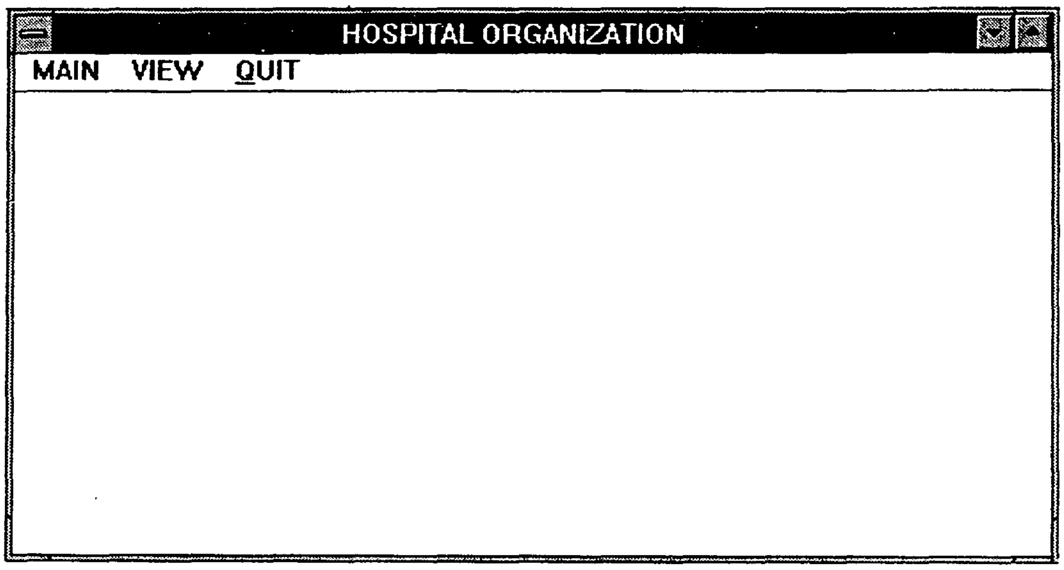


FIGURE 7.1
MAIN WINDOW

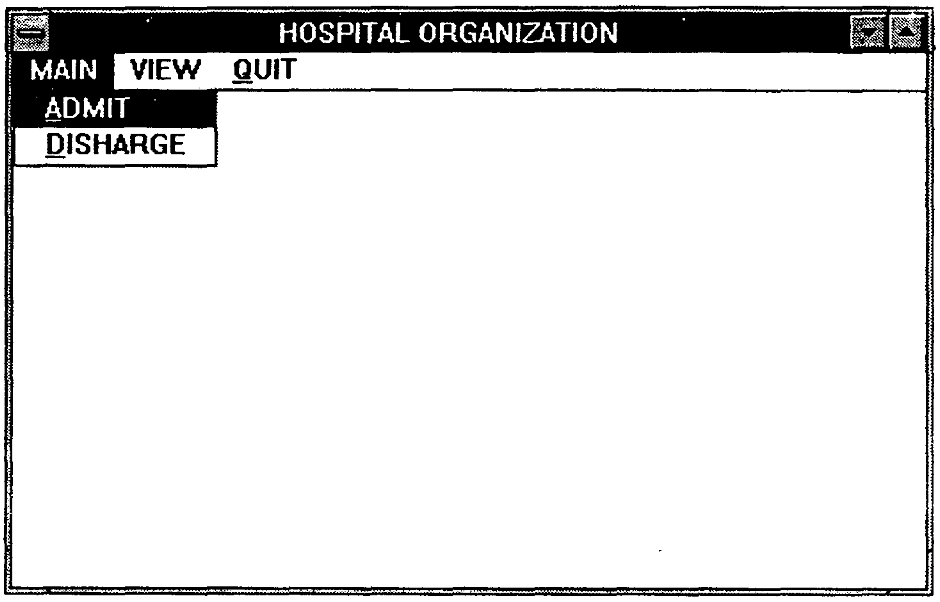


FIGURE 7.2 popup menu MAIN

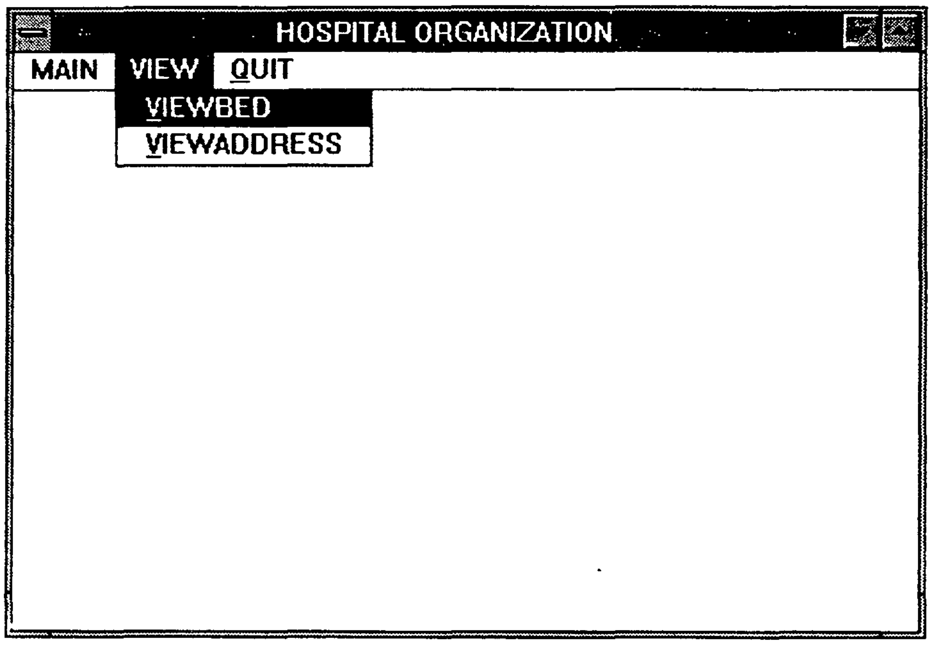
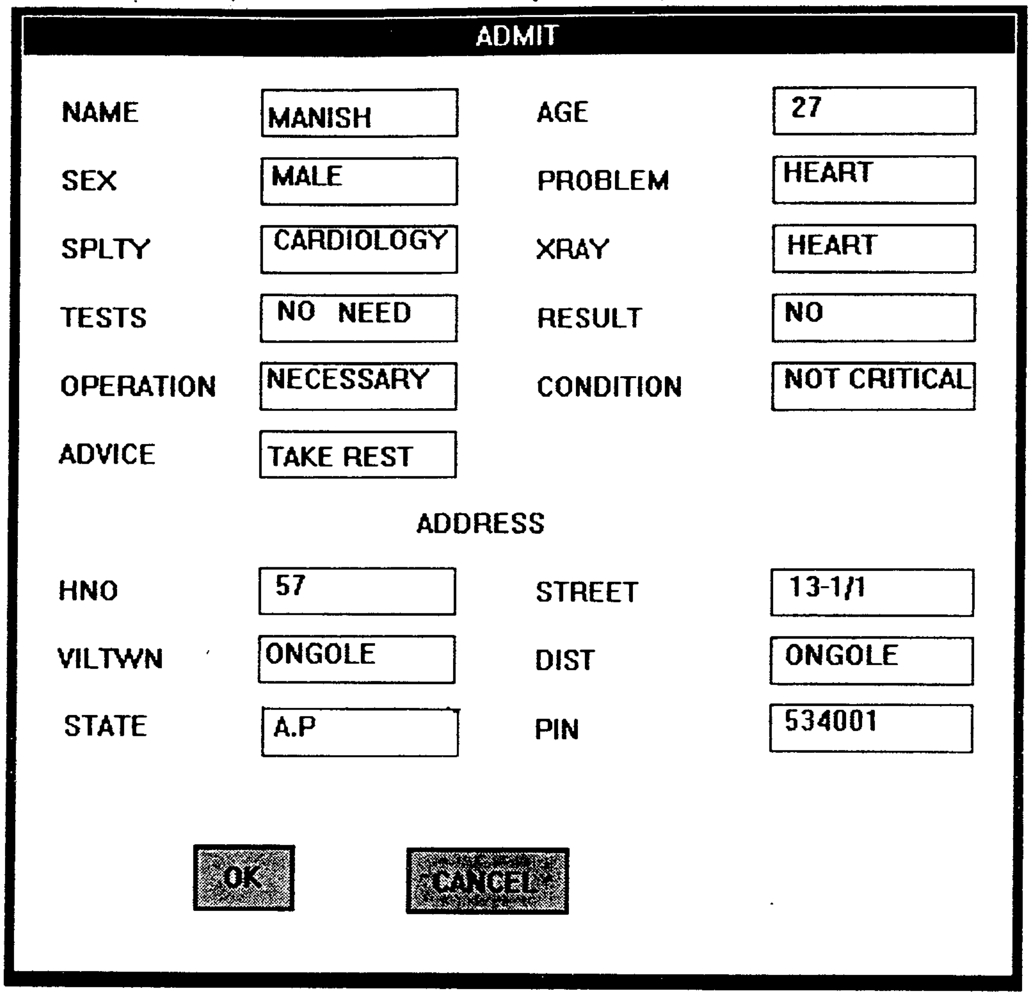


FIGURE 7.3 popup menu VIEW

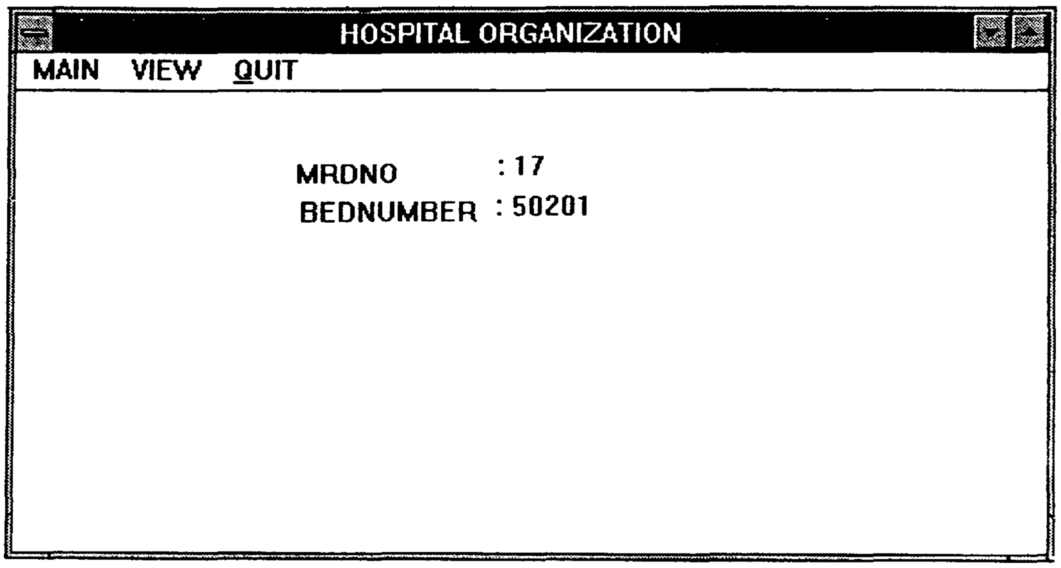

ADMIT

NAME	MANISH	AGE	27
SEX	MALE	PROBLEM	HEART
SPLTY	CARDIOLOGY	XRAY	HEART
TESTS	NO NEED	RESULT	NO
OPERATION	NECESSARY	CONDITION	NOT CRITICAL
ADVICE	TAKE REST		

ADDRESS

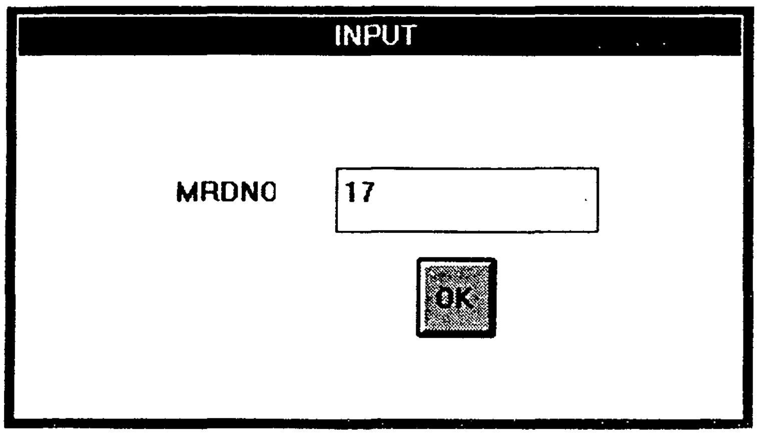
HNO	57	STREET	13-1/1
VILTWN	ONGOLE	DIST	ONGOLE
STATE	A.P	PIN	534001

FIG 7.4 DIALOG BOX WHICH APPEARS ON SCREEN WHEN USER PRESSES *ADMIT* MENUITEM



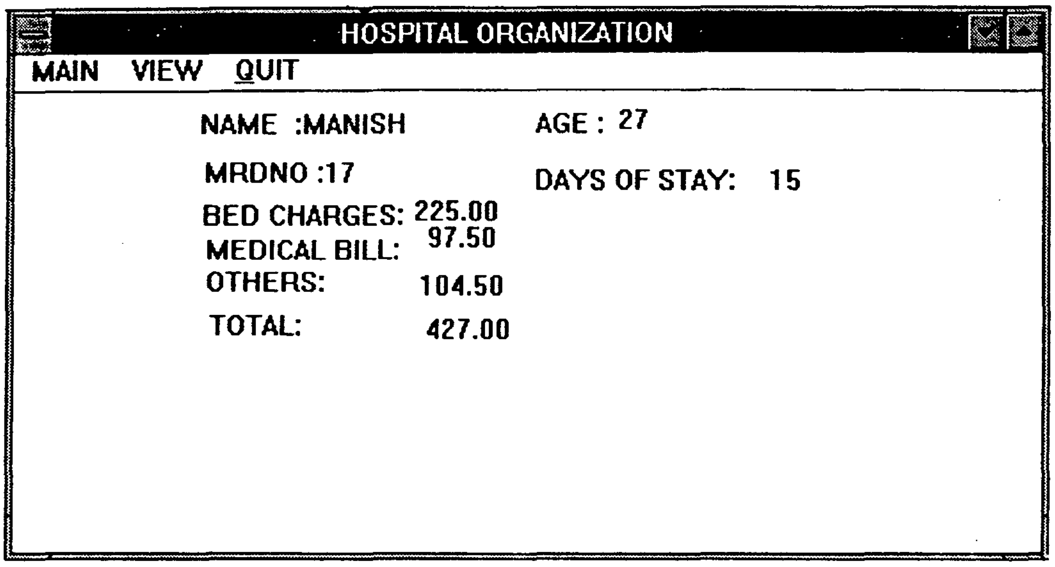
```
MRDNO       : 17
BEDNUMBER   : 50201
```

FIG 7.5 OUTPUT WINDOW CREATED WHILE ADMITTING A PATIENT GIVES OUT ALLOTED MRDNO,BED NUMBER



MRDNO 17

FIG 7.6 INPUT DIALOG TAKES MRDNO OF PATIENT

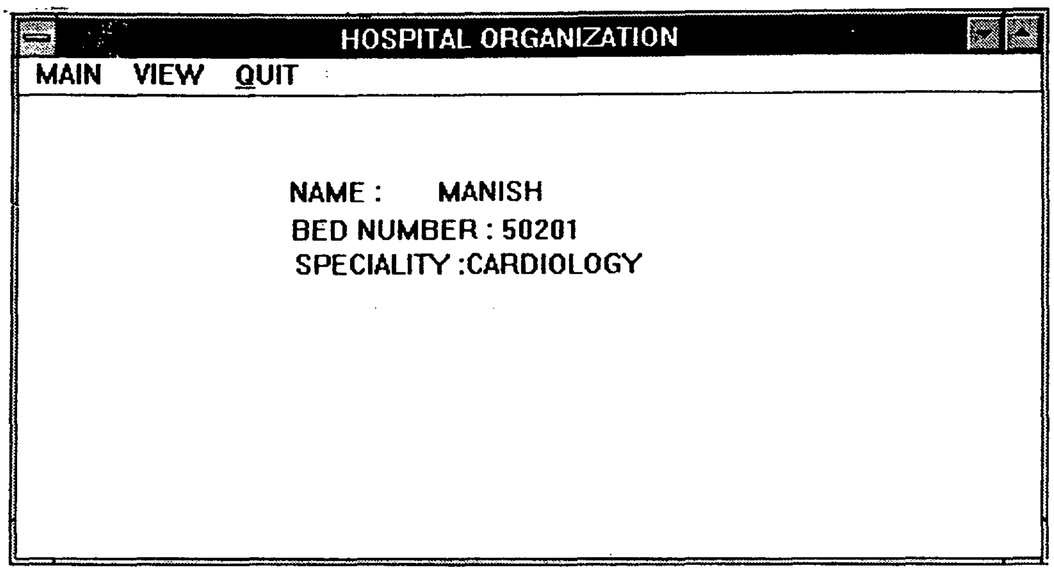


NAME :MANISH AGE : 27

MRDNO :17 DAYS OF STAY: 15

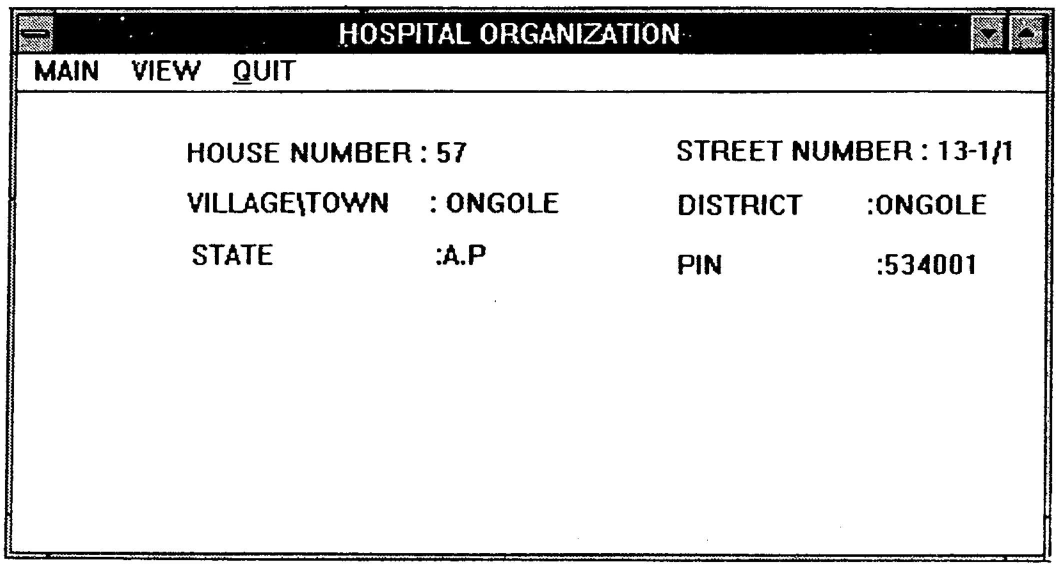
BED CHARGES: 225.00
MEDICAL BILL: 97.50
OTHERS: 104.50
TOTAL: 427.00

FIG 7.7 OUTPUT FOR DISCHAGE DIALOG



NAME : MANISH
BED NUMBER : 50201
SPECIALITY :CARDIOLOGY

FIG 7.8 OUTPUT WINDOW CREATED FOR *VIEW BED* MENUITEM
GIVES OUT BED NUMBER ,SPECIALITY AND NAME OF THE PATIENT



HOUSE NUMBER : 57 STREET NUMBER : 13-1/1

VILLAGE\TOWN : ONGOLE DISTRICT :ONGOLE

STATE :A.P PIN :534001

FIG 7.9 OUT PUT WINDOW APPEARS WHILE USER ASKS FOR A PATIENT'S ADDRESS

BIBILOGGRAPHY

1. Jim Conger, "Windows Programming Primer Plus", Galgotia publications private limited, 1994.

2. Louis Fernades, Yogesh Seth and Anish Kurup, "Borland c++ 3.0 for Windows 3.1 Programming with Object Windows", BPB publications, 1993.

3. Henry F. Korth and Abraham Silbeischatz, "Database System Concepts", Second edition McGraw-Hill Computer Science Series, 1991.

4. Gary Syck, "Object Windows How-To", Galgotia publication private limited, 1993.

5. Brain W. Kernighan and Dennis M. Ritchie, "The C programming Languague", Printice Hall of India private limited, 1993.

6. "Object Windows for C++ User's Guide", Borland International Inc

7. Robert Lafore, "Object Oriented Programming in Turbo C++", Galgotia Publication private limited, 1993.

8. James Conger "Windows API Bible", Galgotia Publications Private Limited,1993.

9. Bjarne Stroup Stroup, "The c++ Programming Language", Addision Wesley Publishing Company, 1994.

10. C.J. Date, "An Introduction to Database Systems" Vol. 1 Adsdison-Wesley narósa Indian Student Edition, 1987.

11. Kaare Christian,"Borland C++ Techniques and Utilities", ZBPress, 1994.

12. Chapman Hall "Object Oriented Programming Systems, Tools and Applications" London 1991.