



**MVPS's
RAJARSHI SHAHU MAHARAJ POLYTECHNIC,
NASHIK.**

DEPARTMENT OF COMPUTER TECHNOLOGY.

ACADEMIC YEAR 2020-21

OBJECT ORIENTED PROGRAMMING (22316)

**MICRO-PROJECT
ON**

“ Develop Hospital Management Application”

SUBMITTED BY

SR. NO	ENROLLMENT NO	EXAM SEAT NO	STUDENT NAME
1	1910020362	240630	Savant Omkar Vitthal
2	1910020360	240628	Raut Atharva Satish
3	1610020163	240586	Wani Pushpak Shrikant



MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION

CERTIFICATE

This is to certify that Ms./Mr. Savant Omkar Vitthal

Roll No. 45 of 3rd semester of Diploma in Computer Technology of
Institute MVPS's Rajarshi Shahu Maharaj Polytechnic, Nasik (Code: 1002)
has successfully completed micro-project in OBJECT ORIENTED
PROGRAMMING(22316) for academic year 2020-21 as prescribed in
curriculum of MSBTE, Mumbai.

Place: Nashik

Enrollment no.: 1910020362

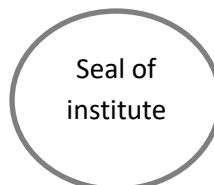
Date:.....

Exam seat no: 240630

Mrs.S.V.Sarode
Course Teacher/Guide

Mr. P. D. Boraste
H.O.D

Prof. Dr. D. B. Uphade
Principal





**MVPS's RAJARSHI SHAHU MAHARAJ
POLYTECHNIC, NASIK**
Institute Code: 1002
COMPUTER TECHNOLOGY DEPARTMENT
Log Book for Micro Project

Academic Year : 2020-21

Semester : III

Name of Course : Object Oriented Programming (OOP)

Scheme: I

Class: SYCM

Course Code: 22316

Title of the project: Develop Hospital Management Application

Group Members:

Sr. No.	Roll No.	Enrollment Number	Exam Seat No.	Name of the Student	Signature of student
1	45	1910020362	240630	Savant Omkar Vitthal	
2	43	1910020360	240628	Raut Atharva Satish	
3	1	1610020163	240586	Wani Pushpak Shrikant	

Project Reporting:

Sr. No.	Date	Discussion & details	Group members present	Teacher's comment/remark	Signature of teacher
1		Formation of groups			
2		Discussion on concept of Micro Project			
3		Topic selection for the Micro Project			
4		Preliminary discussion with guide			
5		Submission of Micro Project proposal			
6		Information Gathered			
7		Literature survey (Introduction)			
8		Discussion with guide			
9		Design of GUI and output			
10		Code generation of modules			
11		Error Evaluation			
12		Merging of all individual modules into one single module			
13		Draft copy of report.			
14		Final report writing			
15		Presentation & oral			
16		Final submission			



**MVPS's RAJARSHI SHAHU MAHARAJ
POLYTECHNIC, NASIK**
Institute Code: 1002
COMPUTER TECHNOLOGY DEPARTMENT

Rubrics for Evaluation of Micro Project

Academic Year: 2020-21

Scheme: I

Semester: III

Class: SYCM

Name of Course: Object Oriented Programming (OOP)

Course Code: 22316

Group Members:

Sr. No.	Roll No.	Enrollment Number	Exam Seat No.	Name of the Student	Signature of student
1	45	1910020362	240630	Savant Omkar Vitthal	
2	43	1910020360	240628	Raut Atharva Satish	
3	1	1610020163	240586	Wani Pushpak Shrikant	

Sr. No.	Criteria	Indicators of different levels of performance					Marks obtained
		Poor (01)	Satisfactory (02)	Good (03)	Very Good (04)	Excellent (05)	
1	Selection Of Application (Programming Elements)						
2	Concept/ Content/ Function Descriptions						
3	Coding						
4	Error solving						
5	Timely Submission						
Total marks Out of 25							
Marks Out of 6							

Mrs. S.V.Sarode
Name & Signature of Course Teacher/Guide



**MVPS's RAJARSHI SHAHU MAHARAJ
POLYTECHNIC, NASIK**

Institute Code: 1002

**COMPUTER TECHNOLOGY DEPARTMENT
ANNEXTURE II**

Academic Year: 2020-21

Name of Faculty: Mrs. S.V.Sarode

Course: Object Oriented Programming (OOP)

Course code: 22319

Semester: III

Title of the project: Develop Hospital Management Application

COs addressed by the Micro Project:

CO-316.1 Develop C++ program to solve problem using Procedure Oriented Approach.

CO-316.2 Develop C++ program using classes and objects

CO-316.3 Implement Inheritance in C++ program.

CO-316.4 Use Polymorphism in C++ program.

CO-316.5 Develop C++ program to perform file operations

Major learning Outcomes achieved by students by doing the Project:

a) Practical Outcomes

.....
.....

b) Unit outcomes in cognitive domain

.....
.....

c) Outcomes in affective domain

.....
.....

Comment/Suggestions about team work/leadership/inter-personal communication (if any)

.....
.....

Roll. No.	Enrollment no	Exam seat no	Student Name	Marks out of 6 for performance in group activity	Marks out of 4 for performance oral/ presentation	Total out of 10
45	1910020362	240630	Savant Omkar Vitthal			
43	1910020360	240628	Raut Atharva Satish			
1	1610020163	240586	Wani Pushpak Shrikant			

**Mrs. S.V. Sarode
Course Teacher/Guide**

**Mr. P. D. Boraste
H.O.D**

ABSTRACT

This paper is scrutinizes the use of different concepts of Object Oriented Programming, enabling viewer to get the complete concept of different aspects of Object Oriented Programming. OOP languages allows you to break down your software into bite-sized problems that you then can solve — one object at a time. To satisfy this we created a simple menu-driven program displaying various Hospital Management System. With use of Inheritance and Polymorphism is also used as a reference to the output, satisfying every need of a perfect OOP program.

Table of Content

1. Introduction.....	3
2. History.....	4
3. Design / Implementation.....	5-9
a) Flowchart.....	5
b) Code.....	6
4. Output And Analysis.....	10-15
a) Output.....	10
b) Analysis.....	15
5. Conclusion And Future Enhancement.....	16
a) Conclusion.....	16
b) Future Enhancement.....	16
6. References.....	17

Introduction

1. Object Oriented Programming:

OOP stands for Object-Oriented Programming. Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions. Object-oriented programming has several advantages over procedural programming. OOP is faster and easier to execute. OOP provides a clear structure for the programs. OOP helps to keep the C++ code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug. OOP makes it possible to create full reusable applications with less code and shorter development time.

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods). A feature of objects is that an object's own procedures can access and often modify the data fields of itself (objects have a notion of this or self). In OOP, computer programs are designed by making them out of objects that interact with one another. OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

The image shows the text "{ O O P }" in a large, bold, black serif font. The letters are widely spaced, and the curly braces are also large and stylized, matching the font's aesthetic.

Fig. Object Oriented Programming

History

Terminology invoking "objects" and "oriented" in the modern sense of object-oriented programming made its first appearance at MIT in the late 1950s and early 1960s. In the environment of the artificial intelligence group, as early as 1960, "object" could refer to identified items (LISP atoms) with properties (attributes) Alan Kay was later to cite a detailed understanding of LISP internals as a strong influence on his thinking in 1966. He thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages (so messaging came at the very beginning – it took a while to see how to do messaging in a programming language efficiently enough to be useful). Another early MIT example was Sketchpad created by Ivan Sutherland in 1960–61; In 1962, Kristen Nygaard initiated a project for a simulation language at the Norwegian Computing Center, based on his previous use of the Monte Carlo simulation and his work to conceptualise real-world systems. Ole-Johan Dahl formally joined the project and the Simula programming language was designed to run on the Universal Automatic Computer (UNIVAC) 1107. Simula introduced important concepts that are today an essential part of object-oriented programming, such as class and object, inheritance, and dynamic binding. But although the idea of data objects had already been established by 1965, data encapsulation through levels of scope for variables, such as private (-) and public (+), were not implemented in Simula because it would have required the accessing procedures to be also hidden.

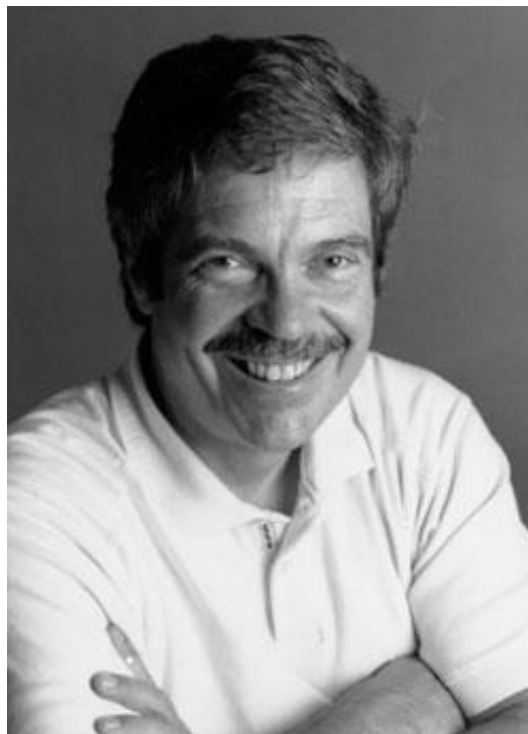
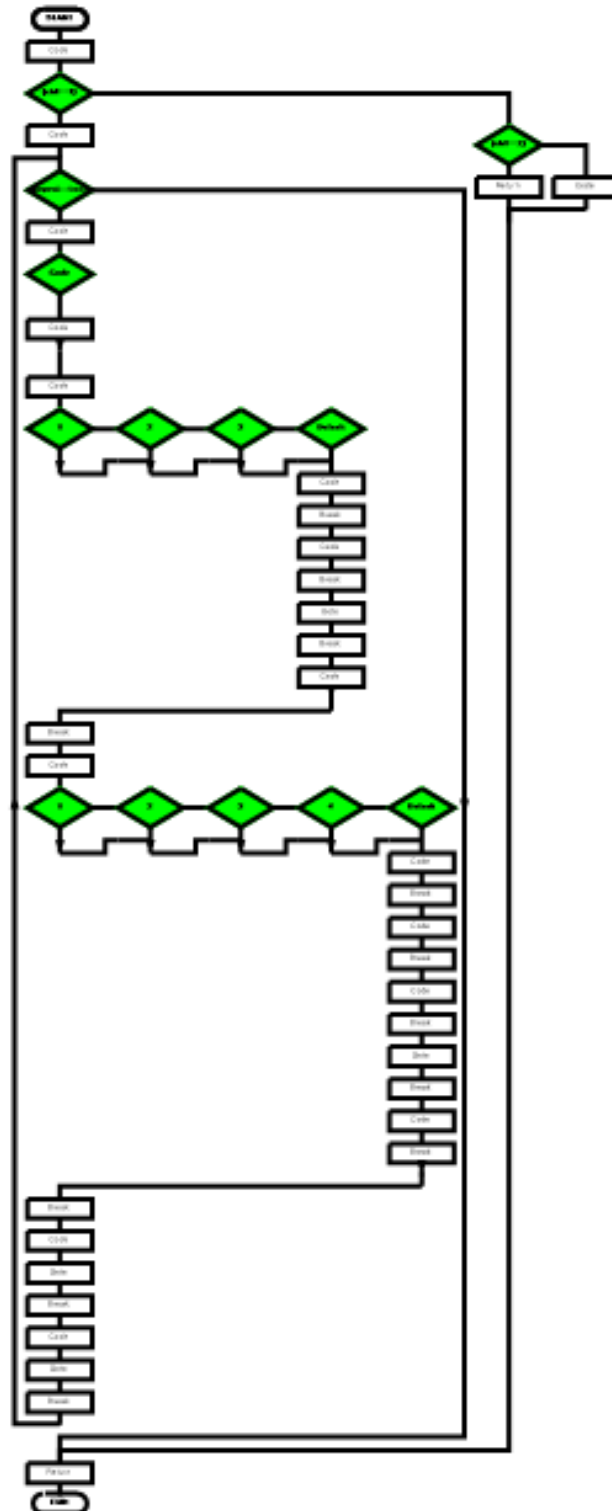


Fig. Alan Kay

Design/Implementation

1. Flowchart



2. Code:

```

1 | #include<iostream>
2 | #include<conio.h>
3 | #include<string.h>
4 | #include<stdlib.h>
5 | #include<stdio.h>
6 |
7 | struct doctor1
8 | {
9 |     int id;
10 |     char name[20],Q[20],age[20],exp[20],city[20],special[20];
11 | };
12 | struct patient1
13 | {
14 |     int id1;
15 |     char name[20],age[20],city[20],dis[20],room[20],sym[20],con[20],date[20],charg[20],bill[20];
16 | };
17 |
18 | using namespace std;
19 | class doctor;
20 | class patient;
21 | class hospital
22 | {
23 |     public:
24 |     // int id=0,id1=0,
25 |     int counter=0;
26 | };
27 |
28 | class doctor:public hospital
29 | {
30 |     public:
31 |     int docid=0;
32 |     struct doctor1 arr[10];
33 |     void add_doc_info()
34 |     {
35 |         int i,en;
36 |         cout<<"How Many Entries you want to add :";
37 |         cin>>en;
38 |         for(i=1;i<=en;i++)
39 |         {
40 |             cout<<"Enter Doctor's ID          :";
41 |             cin>>arr[docid].id;
42 |             cout<<"Enter Doctor's Name          :";
43 |             cin>>arr[docid].name;
44 |             cout<<"Enter Doctor's Age          :";
45 |
46 |             cout<<"Enter Doctor's city          :";
47 |             cin>>arr[docid].city;
48 |             docid++;
49 |             counter++;
50 |             cout<<"\n";
51 |             cout<<"You filled all Entries of "<<i><<i><<" doctor successfully"<<"\n";
52 |             cout<<"enter value for "<<i><<i><<" "<<i><<i><<" "<<"doctor"<<"\n";
53 |         }
54 |     }
55 |     void Display()
56 |     {
57 |         system("cls");
58 |         int n,i;
59 |         cout<<"\n Enter the doctor's ID  to display Record :";
60 |         cin>>n;
61 |         if(n==0)
62 |         {
63 |             cout<<"\n\n                      OOPS!!!!                      "<<"\n\n";
64 |             cout<<"Note:-  No Record To Display  Plz Go Back And Enter some Entries..... "<<"\n";
65 |         }
66 |         else
67 |         {
68 |             int status=0;
69 |             for(i=0;i<docid;i++)
70 |             {
71 |                 if(arr[i].id==n)
72 |                 {
73 |                     status=1;
74 |                     break;
75 |                 }
76 |             }
77 |             if(status)
78 |             {
79 |                 cout<<"\n\n";
80 |                 cout<<"1.Doctor's ID          : "<<arr[i].id<<"\n";
81 |                 cout<<"2.Doctor's Name          : "<<arr[i].name<<"\n";
82 |                 cout<<"3.Doctor's Age          : "<<arr[i].age<<"\n";
83 |                 cout<<"4.Doctor's city          : "<<arr[i].city<<"\n";
84 |                 cout<<"\n Press Any KEY To choose another Option.... ";
85 |             }
86 |             else
87 |             {
88 |                 cout<<"\n\n No such ID in database "<<endl;
89 |                 cout<<"\n Press Any KEY To choose another Option.... ";

```

```

89         cout<<" \n Press Any KEY To choose another Option.... ";
90     }
91 }
92 getch();
93 }
94 void tot_no_of_doc()
95 {
96     system("cls");
97     int i=counter;
98     cout<<"Total Doctor's in Hospital : "<<i<<"\n";
99     cout<<" \n Press Any Button To choose another Option.... ";
100    getch();
101 }
102 };
103
104 class patient:public hospital
105 {
106 public:
107     int docidl=0;
108     struct patientl arr[10];
109     void add_pat_info();
110     void Display();
111     void patient_report();
112     void patient_detail();
113     void tot_no_of_pat();
114     void gen_pat_report();
115 };
116
117 void patient :: add_pat_info()
118 {
119     int i,en;
120     cout<<"How Many Entries you want to add : ";
121     cin>>en;
122     for(i=1;i<=en;i++)
123     {
124         cout<<" 1.Enter Patient's ID                : ";
125         cin>>arr[docidl].idl;
126         cout<<" 2. Enter patient's Name              : ";
127         cin>>arr[docidl].name;
128         cout<<"3. Enter patient's Age                  : ";
129         cin>>arr[docidl].age;
130         cout<<" 4. Enter patient's Disease             : ";
131         cin>>arr[docidl].dis;
132         cout<<" 5. Enter Patient's Room Charge          : ";

```

```

177         else{
178             cout<<" \n\n No such ID in database "<<endl;
179             cout<<" \n Press Any KEY To choose another Option.... ";
180         }
181     }
182     getch();
183 }
184
185 void patient :: patient_report()
186 {
187     system("cls");
188     int i,n;
189     cout<<"\n Enter the Patient's ID to Display Report :";
190     cin>>n;
191     int status=0;
192     for(i=0;i<docid;i++)
193     {
194         if(arr[i].idl==n)
195         {
196             status=1;
197             break;
198         }
199     }
200     if(status)
201     {
202         cout<<"\n\n    *** Patient's Report ***    "<<"\n\n";
203         cout<<"1. Patient's Name" <<arr[i].name<<"\n";
204         cout<<"2. Patient's Age" <<arr[i].age<<"\n";
205         cout<<"3. Patient symptoms" <<arr[i].sym<<"\n";
206         cout<<"4. Patient Disease" <<arr[i].dis<<"\n";
207         cout<<"5. Patient Admit Date" <<arr[i].date<<"\n";
208         cout<<"6. Patient condition At The Time Of Discharge" <<arr[i].con<<"\n";
209         cout<<"Press Any Key To Go Back....";
210     }
211     else{
212         cout<<" \n\n No such ID in database "<<endl;
213         cout<<" \n Press Any KEY To choose another Option.... ";
214     }
215
216     getch();
217 }
218
219
220 void patient :: tot_no_of_pat()
221 {
222     system("cls");
223     int i=counter;
224     cout<<"Total Patients in Hospital : "<<i<<"\n";
225     cout<<" \n Press Any KEY To choose another Option.... ";
226     getch();
227 }
228
229 void patient :: gen_pat_report()
230 {
231     system("cls");
232     int i,n;
233     cout<<"\n Enter the Patient's ID to Display Bill :";
234     cin>>n;
235     int status=0;
236     for(i=0;i<docid;i++)
237     {
238         if(arr[i].idl==n)
239         {
240             status=1;
241             break;
242         }
243     }
244     if(status)
245     {
246         cout<<"\n\n    *** Patient's Report ***    "<<"\n\n";
247         cout<<"1. Patient's Medicine Charge" <<arr[i].bill<<"\n";
248         cout<<"2. Patient's Room Charge" <<arr[i].charg<<"\n";
249         cout<<"Press Any Key To Go Back....";
250     }
251
252     else{
253         cout<<" \n\n No such ID in database "<<endl;
254         cout<<" \n Press Any KEY To choose another Option.... ";
255     }
256     getch();
257 }
258
259
260
261 int main()
262 {
263     system("color BQ");
264

```

```

265 bool repeat= true;
266 int ch1,ch2,ch3,ch4;
267 doctor d;
268 patient p;
269 xyz:
270 system("cls");
271 cout<<"\n\n";
272 cout<<"          *** Welcome to the Hospital Management System ***          "\n\n\n";
273 cout<<"          1. Menu :      "<<"\n\n";
274 cout<<"          2. Exit :      "<<"\n\n\n";
275 cout<<"Enter Your Choice  :";
276 cin>>ch1;
277 cout<<"\n\n\n";
278
279 if(ch1==1)
280 {
281     xyz2:
282     system("cls");
283     cout<<"\n\n";
284     cout<<"  1.  Enter into Doctor's  DataBase  "<<endl;
285     cout<<"  2.  Enter into Patient's  DataBase  "<<endl;
286     cout<<"  3.  Generate Patient's  Bill  "<<endl;
287     cout<<"  4.  EXIT  "<<"\n";
288     cout<<"Please Enter Your choice  :"<<" ";
289     cin>>ch2;
290     while(repeat==true)
291     {
292         system("cls");
293         switch(ch2)
294         case 1:
295         {
296             cout<<"\n\n";
297             cout<<"          *** Welcome To Doctor's DataBase ***          "\n\n\n";
298
299             cout<<"  \t  1. Add New Doctor's Information  "<<endl;
300             cout<<"  \t  2. Display Doctor's Information  "<<endl;
301             cout<<"  \t  3. EXIT  "<<"\n";
302
303             cout<<"Please Enter your choice  :"<<" ";
304             cin>>ch3;
305             switch(ch3)
306             {
307                 case 1:
308                     system("cls");

```

```

309         d.add_doc_info();
310         break;
311         case 2:
312             d.Display();
313             cout<<"\n";
314             break;
315
316         case 3:
317             goto xyz2;
318             break;
319
320         default:
321             cout<<"invalid";
322     }
323     break;
324 case 2:
325     cout<<"\n\n";
326     cout<<"          *** Welcome To Patient's DataBase ***          "\n\n\n";
327
328     cout<<"  1. Add New Patient's Information  "<<endl;
329     cout<<"  2. Display Patient's Information  "<<endl;
330     cout<<"  3. Total Number of Patient's in Hospital  "<<endl;
331     cout<<"  4. EXIT  "<<"\n";
332
333     cout<<"Please Enter your choice  :"<<" ";
334     cin>>ch4;
335     switch(ch4)
336     {
337         case 1:
338             system("cls");
339             p.add_pat_info();
340             break;
341         case 2:
342             p.Display();
343             cout<<"\n";
344             break;
345
346         case 3:
347             p.tot_no_of_pat();
348             break;
349
350         case 4:
351             goto xyz2;
352             break;

```

Output/Analysis

1. Output

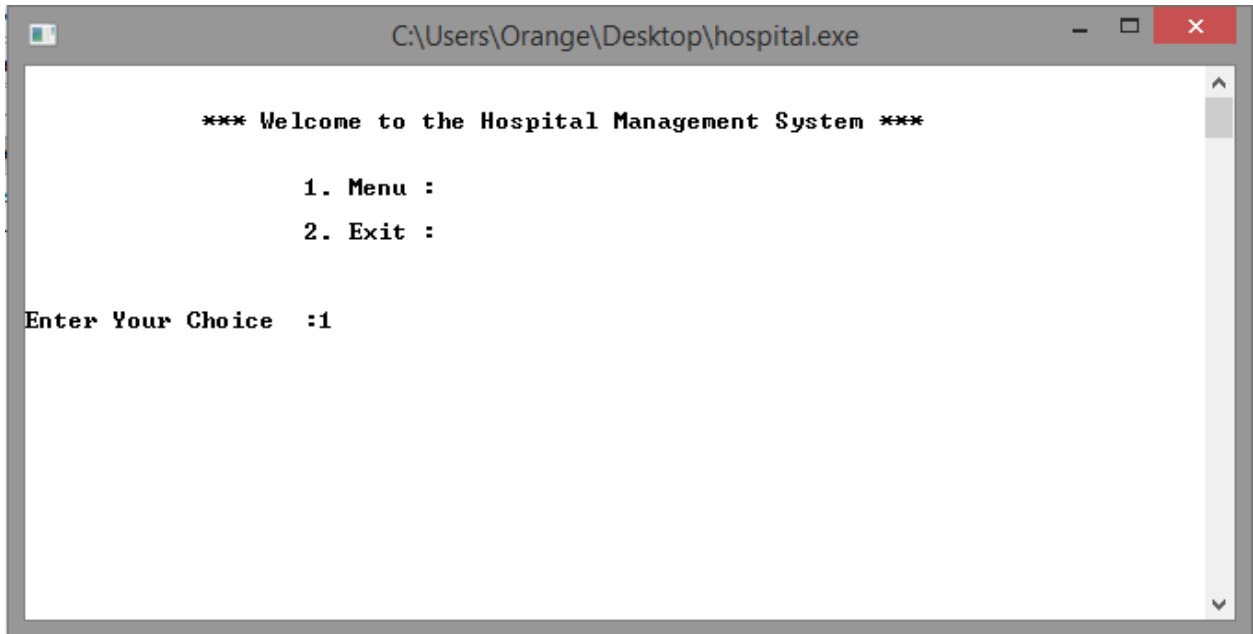


Fig. 1. Main Menu

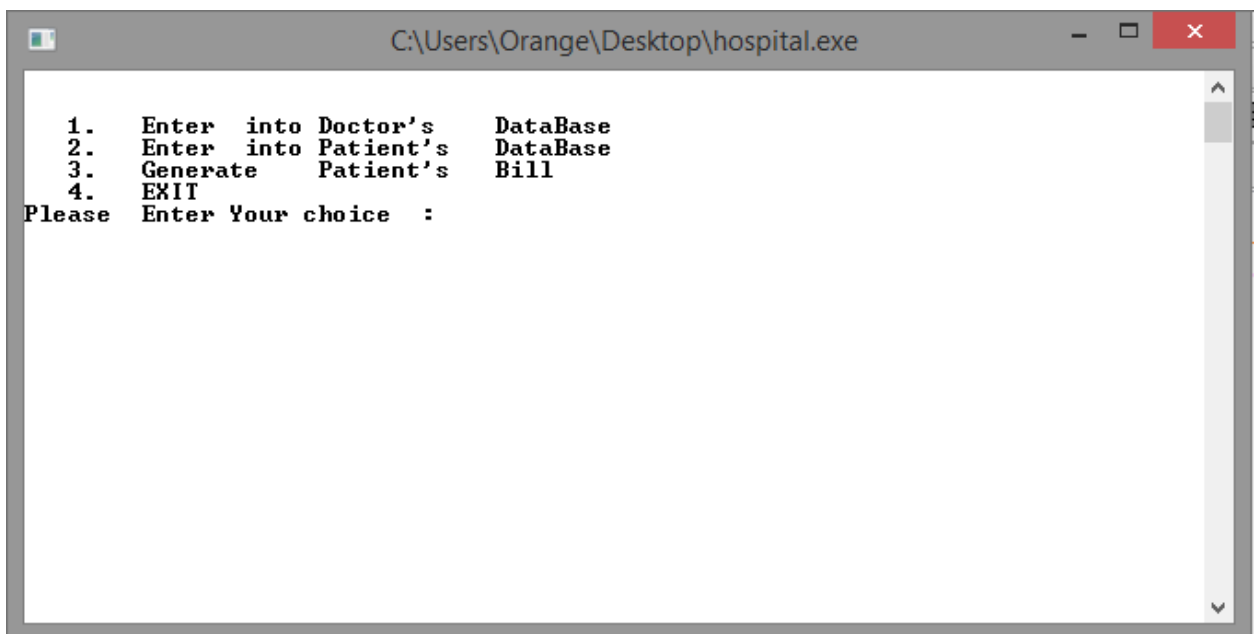


Fig. 2. Menu

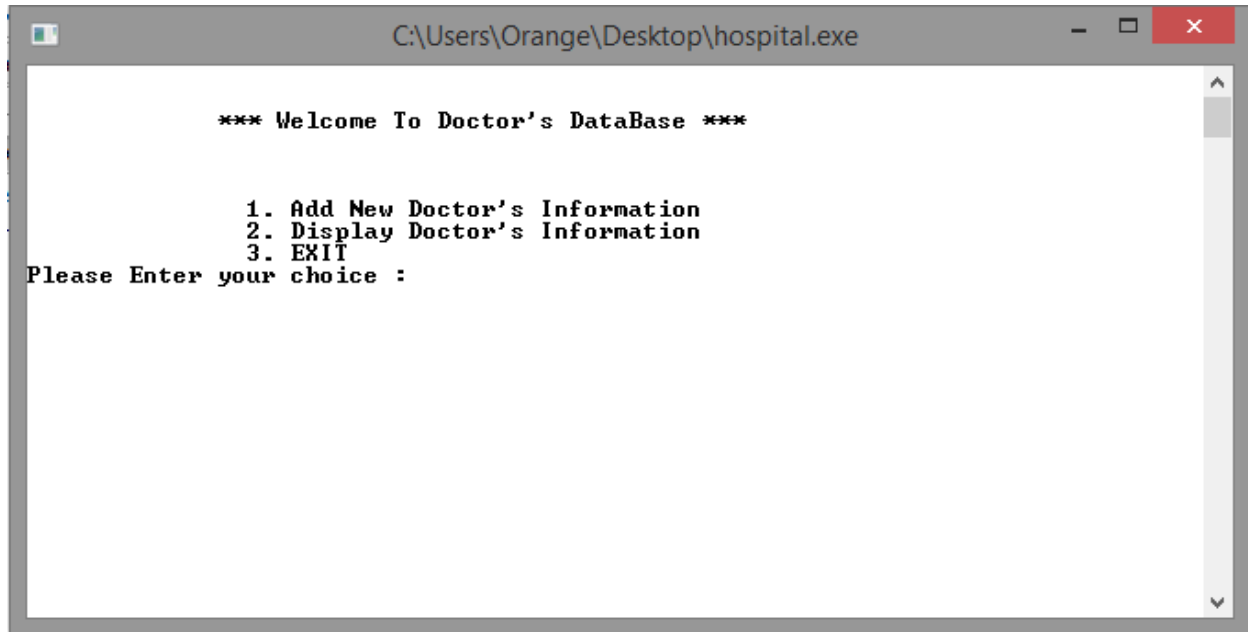


Fig. 3. Doctor's Database

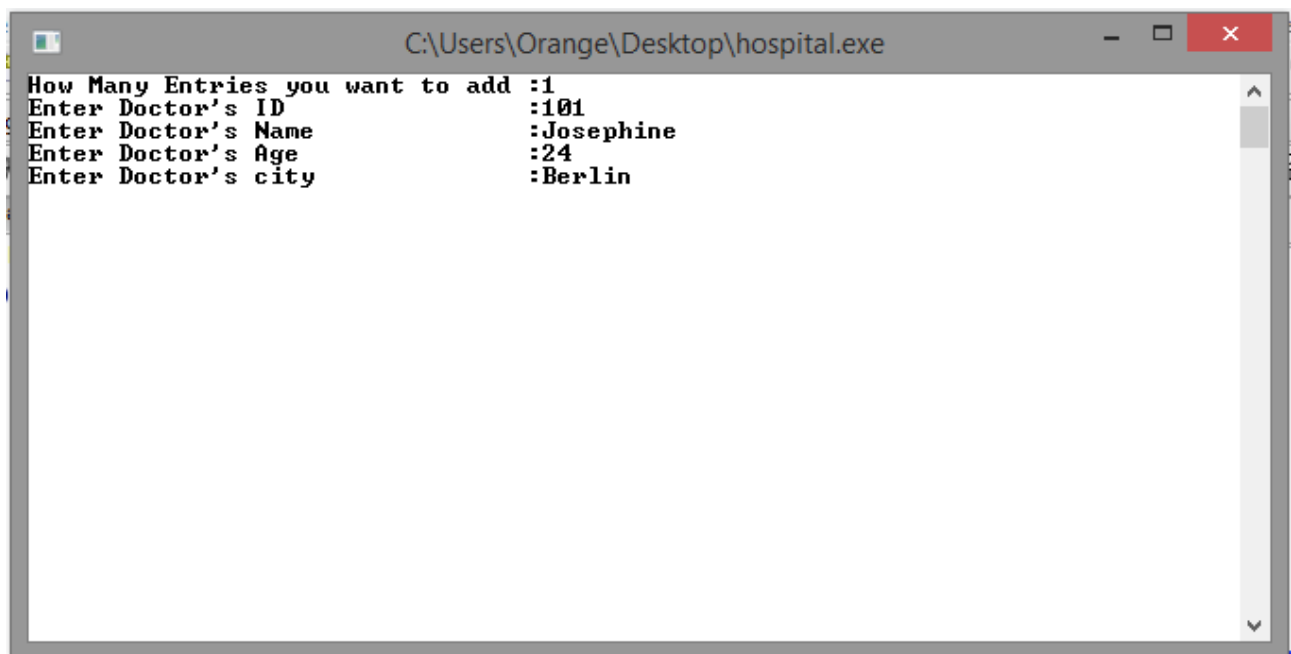


Fig .4. Add Doctor Information

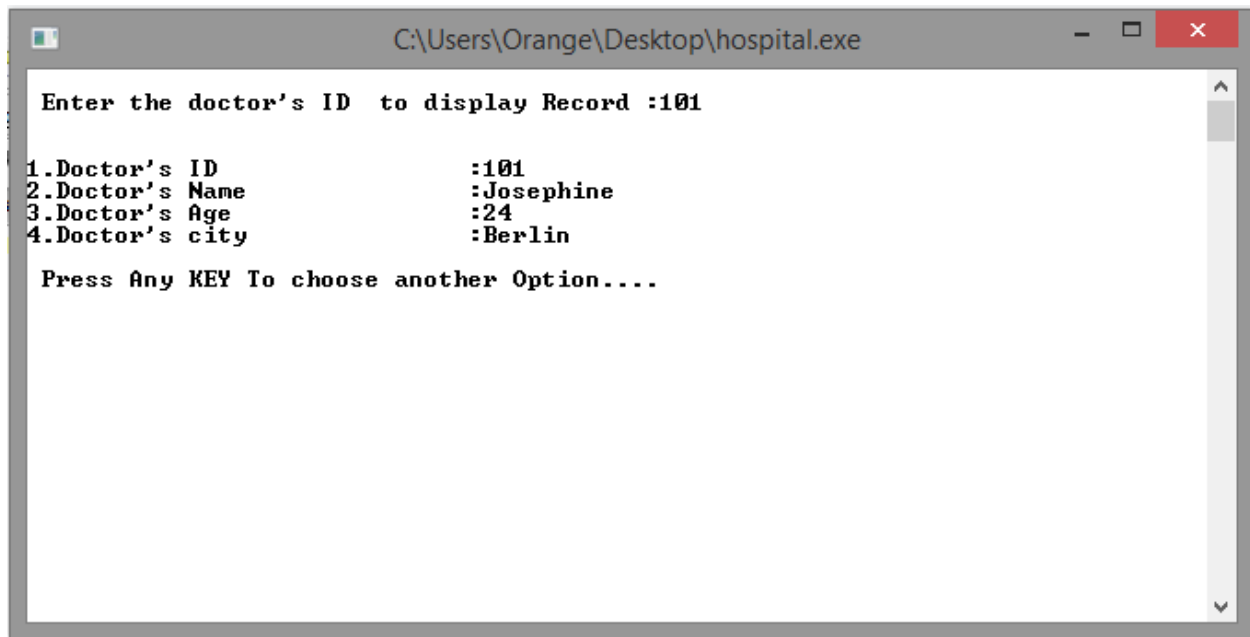


Fig. 5. Display Doctor Detail

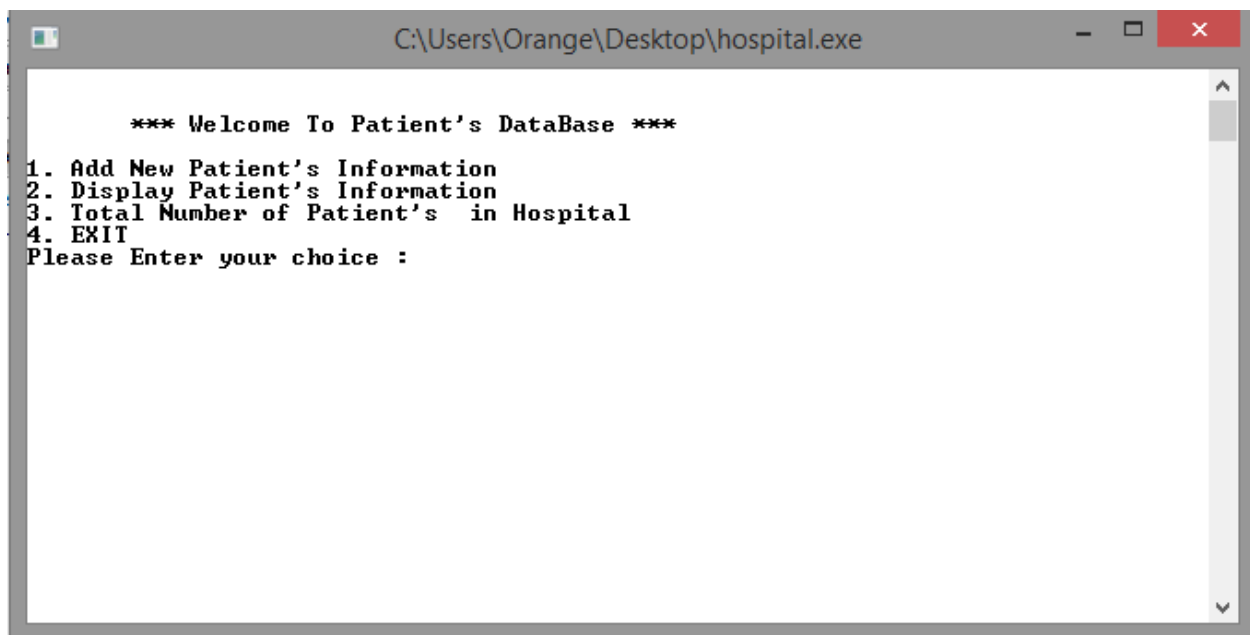


Fig. 6. Patient Database

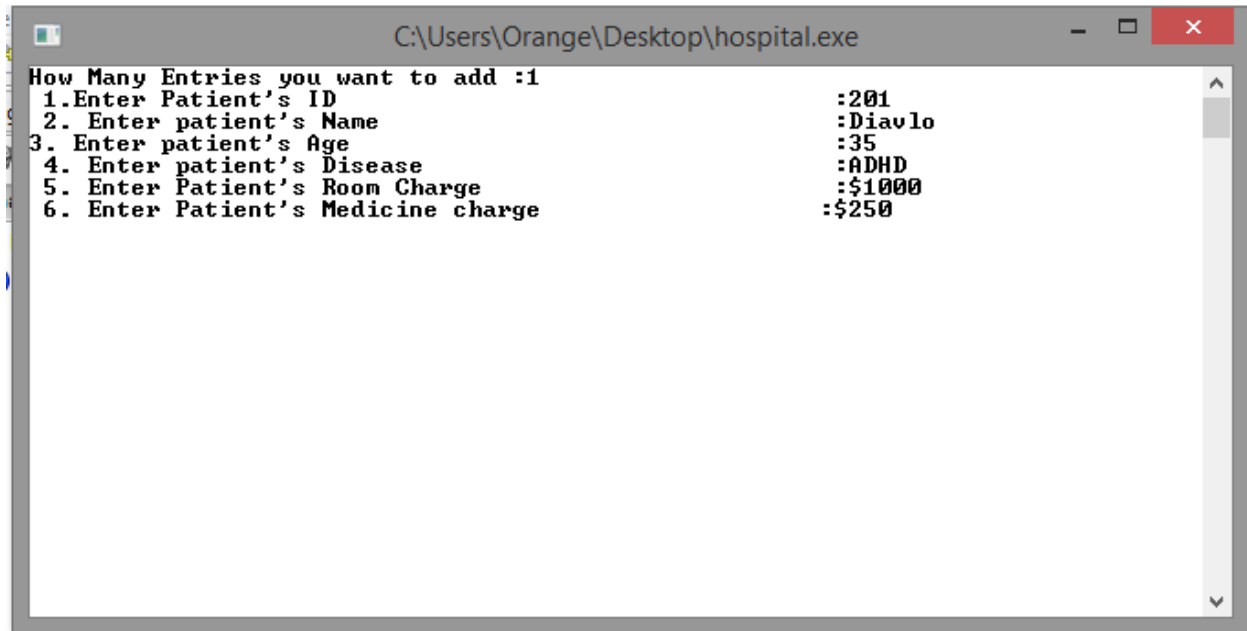


Fig. 6. Add New Patient

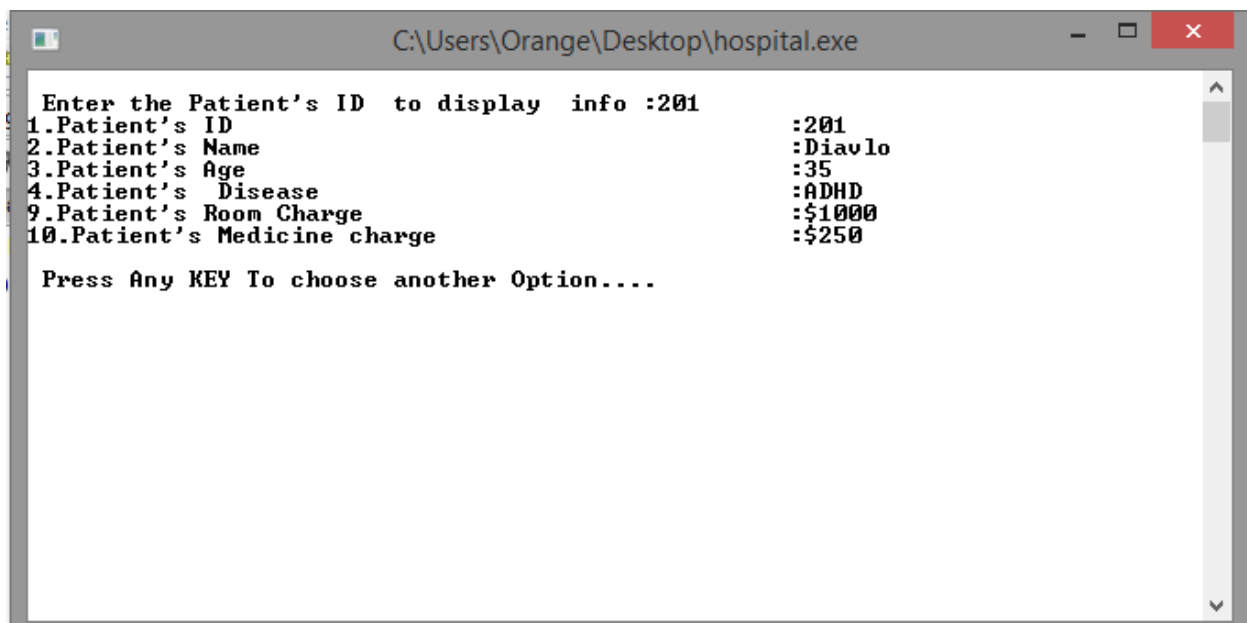


Fig. 7. Display Patient

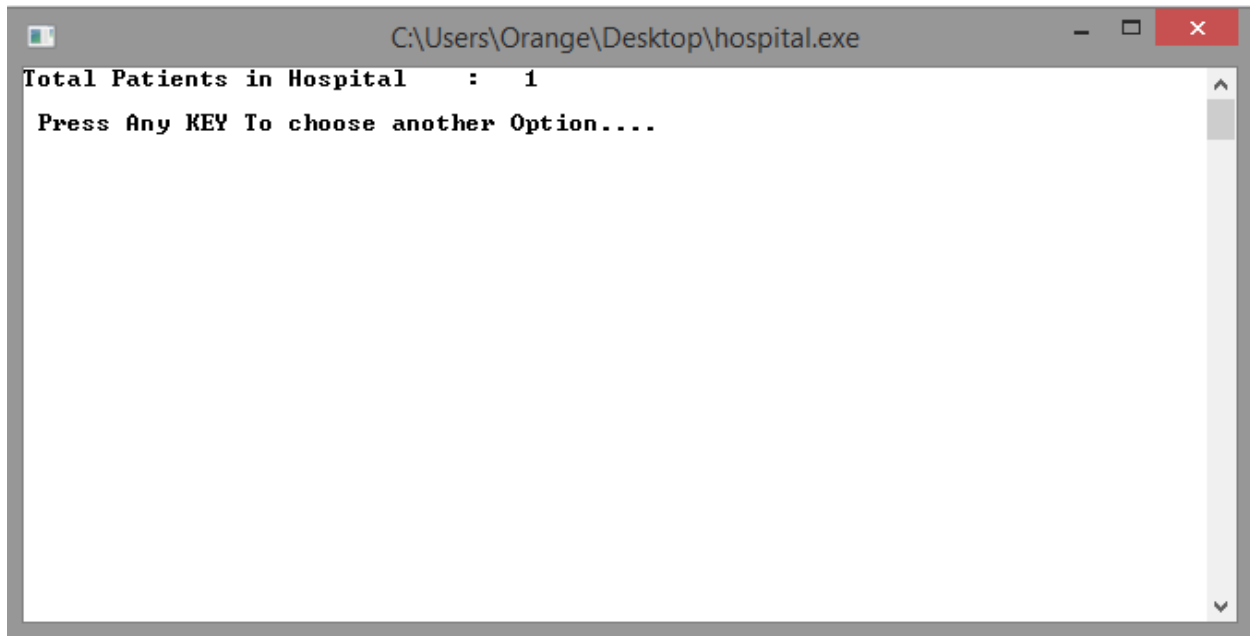


Fig. 8. Total Patients

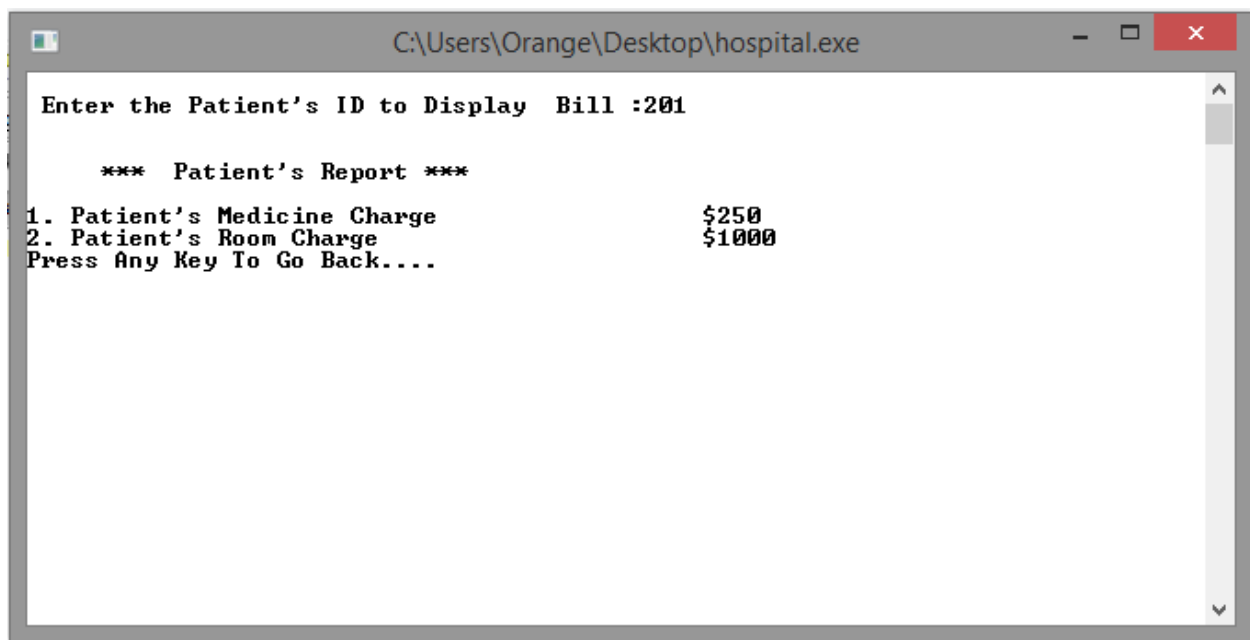


Fig. 9. Patient Bill

Analysis

Many of the most widely used programming languages (such as C++, Java, Python, etc.) are multi-paradigm and they support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include: (list order based on TIOBE index) Java, C++, C#, Python, R, PHP, Visual Basic.NET, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Kotlin, Common Lisp, MATLAB, and Smalltalk. Object-oriented programming uses objects, but not all of the associated techniques and structures are supported directly in languages that claim to support OOP. The features listed below are common among languages considered to be strongly class- and object-oriented (or multi-paradigm with OOP support), with notable exceptions mentioned.

Objects sometimes correspond to things found in the real world. For example, a graphics program may have objects such as "circle", "square", "menu". An online shopping system might have objects such as "shopping cart", "customer", and "product". Sometimes objects represent more abstract entities, like an object that represents an open file, or an object that provides the service of translating measurements from U.S. customary to metric.

Each object is said to be an instance of a particular class (for example, an object with its name field set to "Mary" might be an instance of class Employee). Procedures in object-oriented programming are known as methods; variables are also known as fields, members, attributes, or properties.

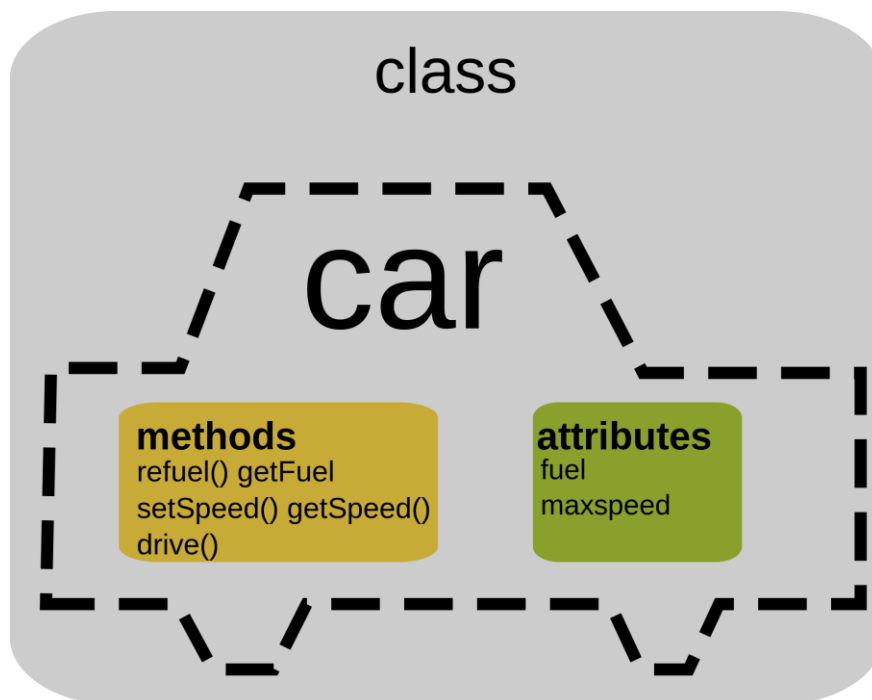


Fig. OOP Class and Object

Conclusion and Future Enhancement

1. Conclusion:

Objects are accessed somewhat like variables with complex internal structure, and in many languages are effectively pointers, serving as actual references to a single instance of said object in memory within a heap or stack. They provide a layer of abstraction which can be used to separate internal from external code. External code can use an object by calling a specific instance method with a certain set of input parameters, read an instance variable, or write to an instance variable. Objects are created by calling a special type of method in the class known as a constructor. A program may create many instances of the same class as it runs, which operate independently. This is an easy way for the same procedures to be used on different sets of data.

Object-oriented programming that uses classes is sometimes called class-based programming, while prototype-based programming does not typically use classes. As a result, significantly different yet analogous terminology is used to define the concepts of object and instance. In some languages classes and objects can be composed using other concepts like traits and mixins.

2. Future Enhancements:

In recent years, object-oriented programming has become especially popular in dynamic programming languages. Python, PowerShell, Ruby and Groovy are dynamic languages built on OOP principles, while Perl and PHP have been adding object-oriented features since Perl 5 and PHP 4, and ColdFusion since version 6.

The Document Object Model of HTML, XHTML, and XML documents on the Internet has bindings to the popular JavaScript/ECMAScript language. JavaScript is perhaps the best known prototype-based programming language, which employs cloning from prototypes rather than inheriting from a class (contrast to class-based programming). Another scripting language that takes this approach is Lua.

Object-oriented programming is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields, and code, in the form of procedures. A feature of objects is that an object's own procedures can access and often modify the data fields of itself.

References

Kindler, E.; Krivy, I. (2011). "Object-Oriented Simulation of systems with sophisticated control". *International Journal of General Systems*: 313–343.

Lewis, John; Loftus, William (2008). *Java Software Solutions Foundations of Programming Design* 6th ed. Pearson Education Inc. ISBN 978-0-321-53205-3., section 1.6 "Object-Oriented Programming"

Deborah J. Armstrong. *The Quarks of Object-Oriented Development*. A survey of nearly 40 years of computing literature which identified a number of fundamental concepts found in the large majority of definitions of OOP, in descending order of popularity: Inheritance, Object, Class, Encapsulation, Method, Message Passing, Polymorphism, and Abstraction.

John C. Mitchell, *Concepts in programming languages*, Cambridge University Press, 2003, ISBN 0-521-78098-5, p.278. Lists: Dynamic dispatch, abstraction, subtype polymorphism, and inheritance.

Michael Lee Scott, *Programming language pragmatics*, Edition 2, Morgan Kaufmann, 2006, ISBN 0-12-633951-1, p. 470. Lists encapsulation, inheritance, and dynamic dispatch.

Pierce, Benjamin (2002). *Types and Programming Languages*. MIT Press. ISBN 978-0-262-16209-8., section 18.1 "What is Object-Oriented Programming?" Lists: Dynamic dispatch, encapsulation or multi-methods (multiple dispatch), subtype polymorphism, inheritance or delegation, open recursion ("this"/"self")

Booch, Grady (1986). *Software Engineering with Ada*. Addison Wesley. p. 220. ISBN 978-0805306088. Perhaps the greatest strength of an object-oriented approach to development is that it offers a mechanism that captures a model of the real world.

Ali, Junade (28 September 2016). *Mastering PHP Design Patterns | PACKT Books* (1 ed.). Birmingham, England, UK: Packt Publishing Limited. p. 11. ISBN 978-1-78588-713-0. Retrieved 11 December 2017.

Jacobsen, Ivar; Magnus Christerson; Patrik Jonsson; Gunnar Overgaard (1992). *Object Oriented Software Engineering*. Addison-Wesley ACM Press. pp. 43–69. ISBN 978-0-201-54435-0.

McCarthy, J.; Brayton, R.; Edwards, D.; Fox, P.; Hodes, L.; Luckham, D.; Maling, K.; Park, D.; Russell, S. (March 1960). "LISP I Programmers Manual" (PDF). Boston, Massachusetts: Artificial Intelligence Group, M.I.T. Computation Center and Research Laboratory: 88f. Archived from the original (PDF) on 17 July 2010. In the local M.I.T. patois, association lists [of atomic symbols] are also referred to as "property lists", and atomic symbols are sometimes called "objects"