CHAPTER 1
INTRODUCTION TO OBJECT ORIENTED PROGRAMMING
1.1 Object Oriented and E-learning

The name object-oriented has been used as an essential in modern years. Terminology invoking "objects" and "oriented" in object-oriented programming made its first advent at MIT in the late 1950s. The concept of E-learning is very large. It was originated in late 1990’s as the technological augmented learning appliances through internet. There are learning environments can based on the learning objective, aim audience, access as physical, virtual or both and type of content. The different techniques are evolves in E-learning. We are living in the globalized era and the age of World Wide Web being broadly connected the world. They have taken off the railing of age, place, time and socio-economic nature and E-learning first move have connected the whole world.

In present research we are combining assistive tools with computing for student have face the difficulties in OOP learning. Assistive tools are more expensive and student in rural area are not possible to purchase this software and application, internet computing solve this problem it can be provide all assistive tools free of cost for student. The technological revolt has created a new dimension in entire education scheme. The field of education has attempted to accomplishment web as communication channel to connect distance learners with their learning materials. The flexible nature of E-learning gives correct information and the students more familiar comfort in the latest education system. E-learning every time may involve the use of internet, software other media and telecommunications and refers to structured and managed learning experiences [1].

1.2 What actual understanding and teaching Problem occurred

There have been a large amount of difficulties faced by teachers, teaching Object Oriented Programming to college students. There have been searched students discover it very problematic to understand object oriented concepts like constructor calling, polymorphism, classes and other Object Oriented concepts. The student who have been revealed to procedural programming, discover it a minute difficult to shift towards Object Oriented Programming. The students taking some time for them to understand object oriented concepts. Student centered learning for programming classes are very important. As a teaching instrument, programming assignments are constructed to encourage student’s development of analytical programming writing knowledge. In case of programming language, this approach focuses on a teaching
concept to solve the given problem. Learning can be made possible within the lecture by making student do things which are called active learning. Understanding object oriented concept is always a critical work for students. There is near about similarly challenging for lecturers to teach these concepts. Over the years teachers have used different methods to teach the concepts. Object Oriented Programming has become popular over recent years and has completely replaced the Procedural Oriented Programming. Object-Oriented Programming provides software reuse the designs and language constructs for, polymorphism, encapsulation, inheritance and modularity. But main problem is object oriented programming difficult to understand for students and first time. Interactive programming and modular programming help us to solving our problems. Object-oriented programming characteristics objects, classes and inheritance and is manageable for many real world applications. This would allow students to concentrate in the important issues of object-oriented programming. We simply analyses the overall object oriented understanding difficulties for all types of users like students, teachers [10].

1.3 Procedural versus OO Programming

Firstly, we bottomless into the benefits of Object Oriented development consider about question arises: what is object? These both are a complicated and a easy question. Software development method is non-trivial that’s why it is complex. It is simple because people already think in terms of objects. Suppose we consider about example of person, consider you see the object as person. And it is defined by two terms: attributes and behaviors by object. The attributes of person, such as eye color, height, age etc. The behaviors of person, such as talking, walking, breathing, and so on. The simple definition of object is, existing both data and behavior.

There are a number of main reasons why in the past two decades object oriented programming has become the dominant programming. The ratio of scaling Object oriented programming is well, from two sided of problems to the most complex tasks. The object oriented programming provides a form of abstraction that used the people and use to solve their daily life problem. Now a day’s most of object oriented programming provides large number of libraries to develop large domains. The Object-oriented programming is used mostly when problem of software crisis, at that time large solutions available. The software crisis simply means that, the tasks we would like to solve with the help of computers and our imaginations and almost
always checking our abilities. Sometimes object oriented programming is not solution for all difficult situations, while in complex software systems. The software development system also utilizes the basic ideas of object oriented programming technology for better implementation using module terminology. As we know

**Old Programming Technique:**

**Structures**

```c
struct queue{
    int a[5];
    int head;
    int tail;
};

struct stack{
    int a[5];
    int top;
}

void main()
{
    struct queue Q;
    struct stack S;
    print( Q.a[2] );
    print( S.a[2] );
    add(S, 3);
    add(Q, 2);
}

void add(queue Z, int x){
    <codes for adding queue elements here>
}

void remove(queue Z, int x){
    <codes for removing queue elements here>
}

void add(stack Z, int x){
    <codes for adding stack elements here>
}

void remove(stack Z, int x){
    <codes for removing stack elements here>
}
```

Fig-1: Old programming technique

programming language very well, still one of the most difficult tasks ever undertaken by humans; becoming prominent in programming requires extra talent, logic, creativity, intelligence, the ability to develop and use abstractions, and back performance. Object-oriented programming is a new way of thinking about what it means to compute and find, from that we can communicate with machine to solve our
critical problems. If reevaluation of object oriented techniques the need change in traditional software development [12].

New Programming Technique: Object-Oriented

1.3.1 Difference between OO and Procedural
The attributes and behaviors are stored within a only one object in object oriented design, the attributes and behaviors are normally separated in procedural or structured design. For the period OO design grows in popularity, but realities that slowed its acceptance. The fact is that there is lot of non-OO systems preferred. At that condition lack of acceptance of OO database. But one point relational database replace by OO database seemed to development purpose. When all of the costs and risks of converting systems from relational to OO databases became apparent, there was no compelling reason to switch. Business find, much of the software development practices today have several development methodologies such as OO and structured.

The structured programming data is differentiating from the procedures, and sometimes the globalized data, that’s why easily outside scope. Many functions can have access to the globalized data easily. Sometimes there is no control over who has access to the data; testing and debugging are much more critical. Here, Objects notifies these problems by concatenating data and behavior into a one package.
Structured programs have complex data structures, such as arrays, and so on. Many characteristics of objects present in C++ structures like classes. Objects are available more than data structures and integer string are primitive data types. Suppose, object contain entities such as integer and string, which are used to giving different parts, and occupy functions and which act behaviors. Methods are used to perform action in object data. But more essentially, The access to members of an object is controlled by us, both attributes and methods. Sometimes this means that some members, both attributes and methods, cannot be shown from other objects.

1.4 Moving from Procedural to Object-Oriented Development

The procedural and object-oriented technologies general differences can be understand as deep as follows.

1.4.1 Procedural Programming

Data of a system is separates from operations that work on data by procedural programming. Suppose we consider about example, if you want the information across a network is send, particular data is send across network (see Figure1.1) that work that, the program at other end network pipe knows what information with it. In this model only data transmit request signal deliver and receive take place between client and server. In this model, the actual code is not sent over the wire (actual data).

![Fig-3: Data transmitted over a wire](image)

1.4.2 OO Programming

The most benefit of OO (object oriented programming) is that data and with its operations that work on data that is code, are both combine together into the single object. Suppose we consider a example, when an object is send across a network, the entire object includes, the data and behavior, goes with it. In Figure 1.2, the Student object is sent over the network.
1. 5 What Is an Object?

Objects are the basic construct blocks of an Object Oriented program. The program which uses OO technology is consisting group of objects. To explain, let’s consider a system consisting objects that represent Student of that university. Each of these objects has two fields compulsory the data and behavior explain in following parts.

Fig-5: Encapsulated Objects

1.5.1 Object Data

The state of object shown by data stored within object. The data could be considered as attribute in Object Oriented programming terminology. In our example, emp attributes could be name, PRN number, date of birth, gender, phone number, and so on. The various objects differentiate attributes containing the knowledge. We take
lots of examples like student object data having stud name, rollno, address etc attributes of objects.

1.5.2 Object Behaviors

   The object can do what? is the behavior of object. Procedures, subroutines and functions are defined the behavior of procedural language. The methods can be used behaviors in object oriented programming and we can call the methods by sending message to it. The student example shows one of the behaviors needful of a student object is to set and return values of different attributes. Each attribute have would have methods, as setGen() and getGen(). Suppose, someone want to know information i.e another object then it can send message to an student object and ask what is gender is. As we know, Portable data and objects and persistent objects: serialization and relational databases, data are build in Object Oriented manner. The most important thing is that, the data is not separate part of code is the part of package [11].

1.5.3 Advantages of object technology

   The group of classes related to well define data abstractions in building of software systems is object oriented software construction. The client and inheritance are structured using two inner class relations in group. Which communicate through feature calls at run time is represented by a collection of objects. The number of quality factors of object technology and its underlying principles allowed designing software. The factors are as follows.

1] Extendibility:

   The feasibility to simply replace and grow existing components and structure. It can improve by minimizing the coupling between software modules, and maximizing the cohesion of single modules. Two techniques are essential for improving extendibility observed by Meyer,

2] Modularity:

   The feasibility to construct a software structure from diminutive parts and to reason about its properties depends on the possessions of these components.

3] Ease of reuse:
The utility as building blocks for new software structure, and the applicability of already available software components to various contexts.

4] Think of simplicity:
A easy architecture will always be easier to make suitable to replacement than a complicated one.

5] Decentralization:
Object technology allows uplifting all these essential factors in successive software when properly applied. The more autonomous the modules, the biggest the likelihood that a easy replacement will cause just one module, or a tiny number of modules, rather than triggering off a chain reaction of replacement over the whole structure.

6] Abstraction and information hiding
Information squirrel away is an essential means to getting high levels of abstraction. A class to decide that a characteristic is available to all clients by the author or to specified clients only. It is also feasible to write a class as deferred, i.e. specified but not fully build. How a given feature is implemented and abstraction allows stripping away irrelevant details, An abstract concept may have large amount of various implementations; To use a class through a clearly defined interface then clients should be able, without the essence to see the implementation particulars.

7] Inheritance
A class can become heir from other classes, thus it uses the others’ characteristics in addition to its own. Many are variants of others; Software development usually contains a various number of classes. Inheritance is a commodious tool for conceptualization and specialization; it helps explore co-relations between the concepts modeled by classes, and it results in a clearly and compact software structure.

8] Static typing
A run time type safety can be occurred by well defined type safety and the not present of run time errors of certain kinds, regular rule and compulsion a number of type declarations. The motivation in the correctness of a structure before the system is accomplished by increasing static typing.
9] Polymorphism and dynamic binding

The objects of various types can be attached with entities. Polymorphism is not arbitrary in typed language and it is controlled by the type rules. Which may varying in different executions of the call, the corresponding object calling a feature on an entity triggers the feature corresponding to the actual run-time type. Dynamic binding is adequate version of a feature is referred by run time choice. The forward stage in the implementation of software engineering techniques is carrying the benefits of object technology to a substituting context. Providing a minimum support for advanced techniques such as generosity, polymorphism, and inheritance, and they cannot take full benefit of modularity, extendibility, and the potential for reuse provided by object technology in existing concurrent object-oriented languages. Developing within the same framework both sequential and concurrent programs and the ending part between parallel and concurrency be no impediment to reuse is important for practical software engineering. To minimize the syntactic and semantic separated of sequential and concurrent code by integration of concurrency and object oriented techniques [35].

Software as a set or group of programs and solitary data in object-oriented structure development methods deviate from regular development techniques in that the regular or traditional techniques view. It can be defined the program as,

\[ \text{Programs} = \text{Algorithms} + \text{Data Structures} \]

“A set of mechanisms for executing certain action on certain data is called as a software system”. The important discrimination between newer object-oriented methodologies and traditional system methodologies, on their primary focus dependent.

- Traditional approach
- The main focuses on the methods of the system
- Object-oriented systems development
- The concentration on the object, which joins data and functionality.

1.6 What Is a Class?

A class is a copy for an object that is class is binding of data and function into single unit. When instantiate an object, we use a class as the basis for the object is how built. It is most critical to describe a class without using the term object and
without class, the object can’t be instantiated. Suppose an example, a specific individual student is an object. However, someone had to have created the blueprints (that is, the class) to create student class. Here, we can say that class is chicken and egg dilemma.

![Class example of person object](image)

For objects such as points and rectangles, the field names, method dictionary, and protocol will be the same for many objects. In an implementation, then, it is reasonable to group these together into a class. Further, rather than having multiple methods in a given dictionary, the methods could be partitioned among sub- and super classes, as in Smalltalk [11].

### 1.6.1 Object oriented principles

The main two principles are Abstraction and encapsulation that are most useful in Oriented Object approach to software development. Abstraction guves
permission to us, consider complicated things that are essential ideas while not select not essential detail that would confuse as encapsulation allows us to only focus on what something needed without considering the confliction of how it works. The use inheritance and polymorphism can be done with the help of two basic fundamental principles of object orientation are specialization and generalization. General divisions of objects which have same properties is consider in generalization and then define specialized sub classes that inherit the properties of the general divisions.

E.g.: Class

![Class diagram](image)

**Fig-7: Class object**

E.g.: Abstraction

![Abstraction diagram](image)

**Fig-8: Abstraction model**

The concept active learning which tells students to do things on his own way. Suppose, we can taking a small example, while teaching the concept of classes as follows,

class dist
```cpp
private:
int ft;
float inch;
public:
void get();
void display();
};

void dist:: get()
{
    cout<<" To Enter value the ft "<<endl;
    cin>>ft;
    cout<<"Enter the inch"<<endl;
    cin>>inch;
}

void dist::display()
{
    cout<<ft<<"\""<<inch<<"\""<<endl;
}

void main()
{
    dist d;
    d.get(); //calling member functions of the class with
    d.display();//member access operator
    return 0;
}
```

Here, the encapsulation concept is elaborate. The consideration is that get() and display() are methods of the class and these functions could invoke all members easily and the students could have lots of questions on his mind but fact of data encapsulation and the fact that the methods of a class can access the data members of that class, the students cannot understand easily while they writing the given code proper manner. Giving home works and assignments on class based concept to help them to known easily these concepts.

1.6.2 Bottom-up Design
The making of independent program modules this method can be used. The starting first phase of development overall planning process bottom-up approach involves to develop these program modules. The development of program is depending upon the known features of these separate modules. There is exact coherence of bottom-up design and top-down design i.e. - both are different processes. The specific to general process involves writing a modular program. The modules are building upward direction until reach to general solution obtained. The given methodology in modular programming is not acceptable so far. When the task at hand involves just the changes and updating of already available program to calculate the needed result at that time given method adopt.

1.7 Object Oriented Programming

Sometimes, language design depending on the size of the programs, if program were generally small and if program size become large, the attention have been changed. The most regular statement in small programs is regularly assignment statement. In major programs procedure call to function or subprogram is most common statement. It is major issue, when passing correct parameters to the correct subprogram. All classes that are closed to the difficulties to be found solution is define by object oriented analysis. The hierarchical structures can be handled the small programs. The institution is much of network systems in large programs. Differentiation a network of subsystems from program, object oriented approach is more realistic. In object oriented programming language state is central or local to the object itself. Local variables declared as area of the object and cannot accessible to components outside the object by the state of an object is represented. The local state can be accessed and changed with every object contains a set of functions and procedures. They are automatically access the objects data, ordinary functions and methods are called methods. Sending the object message is also known as object method calling.

Objects are declared, to creating a structure for the regular state and functions. Class is known as given pattern and it is most important as data type. A class is building in standard ways of type system of the language and a class is a type in object oriented programming. The variables are declared in language like C or any other programming and also particular classes the object is declared. All we know that objects are the instance of the class. Which is a division of module providing data and subroutine, the main concept of object-oriented programming is object. The entity
which represents internal state i.e. – objects and that can attend to message i.e. – calls to its sub functions. Consider an example stud records object, it consisting of the details of all registered studs [21].

It will change its state to reflect change, when message sent to it tell adding the new student details. It prints all registered student list when message is sent to print itself when message arrives. Software engineering is to start by identify the objects considered in a problem and detect the message that objects should reply in object oriented approach. Here, conclusion is that group of objects and each object with its self data and also own working responsibility. The objects are working with each other by sending message to each other.

**1.7.1 Object Oriented Programming Properties**

Following properties are show by Object oriented programming: Abstraction data, encapsulation, polymorphism and inheritance.

**a. Data Abstraction**

The isolation of the details of how it is constructed from primitive data objects compound data object is used. The differentiation between the essential properties of concrete details and a data type is data abstraction. It model classes appropriate to the problem for simplifying complex reality and take exact level of inheritance to the aspect of difficulty. Suppose we consider the example of class car having achieved through composition i.e. created of Engine, Steering objects, Gearbox and more components. One doesn’t need to know how the every separated components work together internally at the time of build car class, but to check how to interface them and sending receiving messages from them and how the different objects communicate with each other in class.

**a. i. Problems with Data Abstraction**

A sort of black box defines by an abstract data type. It doesn’t really communicate with the remaining of the program, once it has been defined. Modifying its definition, there is no way of taking it to new uses except. Now we consider a type figure for use in a system of graphics. The system has to stand by triang, sqr and cir is consider.

That have some classes is assumed:

```cpp
class pt{ /* ... */ }

class colour{ /* ... */ }

defining a figure like this:
enum kd { cir, triang, sqr };

class figure
{
    pt centre;
    colour coll;
    kd kr;
    // representation of figure
    public:

Fig-9: Interaction of different objects
pt where() { return centre; }
void movee(pt to) { centre = to; draww(); }
void draww();
void rotatee(int);
// vast operations
);
The “type field” kr is needed to give the functions as draww() and rotatee() to creating what type of figure they are fixing with (in a Pascal like language, one might use a different record with tag kr). The function draww() might be defined like this:
void figure::draww()
{
switch (kr)
{
  case cir:
    // a circle draw
    break;
  case triang:
    // a triangle draw
    break;
  case sqr:
    // a square draw
}
}

draww () function must “know about” all the kinds of figures there are. The code for any different function maximize every time a new figure is added to the system. When, define a new figure, every operation on a figure must be examined and modified if necessary. Unless you have access to the source code for every operation, you are unable to adding a new figure to a system. The code of every essential operation on figures, while adding a new figure, it requires great knowledge and potentially generated errors into the code handling other figureshapes [25].

b. Encapsulation

Encapsulation is the binding data and function into single one separate unit. It is the capacity to collect codes and data combine in a place and hides that data from Outside world. It through the associated code forcing anyone who wants to access.
The code everywhere to deal directly within data structures in structured programming.

c. Inheritance

Inheritance is the technique of an existing class to create new classes. Thus the new created classes are called derived class and existing class is referred to as a base class. The Sub class inherits all the features inherited in the base class. It is perhaps one of the most essential characteristic of object-oriented programming paradigm. Inheritance can support base class modification, reusability, reliability.

A single inheritance is when every derived class can inherit from only one base class, where as a multiple inheritance is a case in which a class may inherit from two or more base classes. C++ and java provide multiple inheritances in newer object oriented languages. Its graphs can be acyclic instead of a tree in a language with multiple inheritances. Multiple inheritances can be important but its approach can be complicated.

Reusability is an essential characteristic of OOP’s. It is always good practice to reuse something rather than time to create the same thing all over again there use of a class that already been implemented, tested and debugged used many times can save the efforts of testing and implementing the same again.

Fortunately C++ strongly supports the concept of reusability this means that there is we can plus additional characteristic to a present class without changing it. Deriving a new class from the existing one is possible. The new class will have the all combine characteristics of both the classes.

The inheritance is the mechanism of deriving a new class from old one. The old class is referred to as base class and new one is called derived class can inherit same or all the properties of the base class. The C++ classes can be reused different ways once a class has been written and tested it can be adopted by other programmers to suitable their requirement. This is done be creating new classes reusing the features of existing one classes. In another word the inheritance can be defined as the mechanism of deriving a child class or derived class from its parent or base class is called inheritance.
Defining Derived Classes

Syntax:

```cpp
class derived class_name : visibility label Base_Class_name;
{
    -----------------------------;
    Member of derived class;
    -----------------------------;
    -----------------------------;
}
```

A derived class is defined by specified its relationship with a base class in addition to its self details, the colon indicates that the visibility mode is optional not essential and its present as to be either private or public. The by default visibility label is private visibility mode is specified whether features of base class are privately derived or public derived.
Fig-11: Tree of Object data

Eg. :-
Class KBM : private abc  //private declaration
{
    Member of KBM
}
Class KBM : public abc  //public declaration
{
    Member of KBM
}
Class KBM : abc
{
    Member of KBM     //private declatation
};
When a base class is inherited privately by the derived class. The public members of base class becomes private member of class and therefore the public member of base class can only the access by member function of derived class. They are in accessible to object of derived class; the result is that there is no member of base class is inviolable or accessible to object of derived class.

When base class is publically inherited, public properties of base class becomes public properties of the derived class and therefore they are accessible to the object of derived class. The private members are not inherited and therefore private properties of base class will not becomes the properties of the derived class cannot be inherited either in public or private mode in both cases.

Eg. :-

Public derivation

Class K
{
    int k;
    public:
    int r;
    void getab();
    void showab();
};

Class R : public K
{
    int p;
    public:
    void mul1();
    void display1();
};

Eg:- Private Derivation

Class R
{
    int k;
    public:
    int r;
Fig-12: Class digram for class K & R

```cpp
class M : private R
{
    int p;
    public:
    void mul1();
    void display1();
};
```

Fig-13: Class diagram for class R & P

```cpp
fig-13: Class diagram for class R & P
```

```cpp
Class K
private Section
K
public Section
R
getab()
showab()
```
#Access control and Inheritance:

All the non-private members of its base class can be accessed by the derived class. Thus, base class member that should not be approachable to the member function of derived classes could be declared base class as private. Following table shows the various access types who can access them in the following way according to.

<table>
<thead>
<tr>
<th>Access</th>
<th>Public</th>
<th>protected</th>
<th>private</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same class</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Derived class</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Outside class</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table-1: Access control in classes

1.7.2 Types of Inheritance:-

There are five inheritance types available in C++ programming language.

- a. Single Inheritance
- b. Multiple Inheritance
- c. Multilevel Inheritance
- d. Hierarchical Inheritance
- e. Hybrid Inheritance

**a. Single Inheritance:**

A class is derived from single base class is known as single inheritance.

```
A                      Base Class/Parent Class/Super Class

B                      Derived Class/Child Class/Sub Class
```
In above figure derived class B with only one base class A present this structure is known as single inheritance in another way, if one derived class acquired the properties of only one base class.

Now we think about a simple example to demonstrate single inheritance. The below program shows a class B as base and a class D as derive the class B content data member private, one public data member and three public member functions. The class D content one private data members and two public members function [26].

class B
{
    int x;
    public:
        int y;
        void getxy();
        void getx();
        void showx();
}
Class D : public B
{
    int z;
    public:
        void mul1();
        void display1();
};
void B : : getxy()
{
    x=5;
    y=10;
}
int B : : getx()
{
    return(x);
}
void B::showx()
{
    cout << "x" << x;
}

void D::mul1()
{
    z = y * getx();
}

void D::display1()
{
    cout << "x" << getx() << "n";
    cout << "y" << y << "n";
    cout << "z" << z << "n";
}

void main()
{
    D q;
    q.getxy();
    q.mul1();
    q.showx();
    q.display1();
    q.y = 20;
    q.mul1();
    q.display1();
    getch();
}

The Class D is a public derivation class B. Therefore all public members of B inherits D and retains their visibility. Thus public member of the derived class D also a public member of a class B. The D can not inherit private member of B class.

The program illustrate that the object of class D have obtainable to all public members of B. For example void showx(), getxy(), void getx(), although the data member x is private in B and cannot be inherited, object of D are able to
obtainable it inherited through an member function of B. Let us now private derivation.

Eg. :-

Class B
{
    int x;
    public:
    int y;
    void getxy( );
    void getx( );
    void showx( );
};

Class D : private B
{
    int x;
    public:
    void mul1( );
    void display1( );
};

In private section derive the public member of the base class B becomes private variables or elements of the derived class D. Therefore the object of D cannot have directly approachable to the public member function of B. The private member of B cannot be inherited by D.

The statement such as

D d;
    d.getxy( );
    d.getx( );  \
    This are private
    d.show( );

The code will not work however these function can be used inside mul1() ,display1( ), Display function(mul1( ),display1( )) like the normal function

void D :: display1( )
{

Hence, the private member of base class cannot be inherited either in the publicly or in private mode.

b. **Multiple Inheritances**

A derived class can fall to attribute of two or more base classes is called as multiple inheritance. It allows combining feature of different present available classes into new class. The derived class with multiple base classes syntax of as follows.

Syntax:

```
Class derived_class_name : visibility mode class R-1, visibility mode class R-2,…….. R-n.
{
    Body of Class
}
```

Eg. :-

```
Class M
{
    ............
    ............
    ............
}
```

```
Class H
```
Class S: public M, public H
{

};

Eg. :-
Class stud
{
int rno;
char n[20];
public:
void get( )
{
    cout<<"n enter roll no";
    cin>>rno;
    cout<<"n enter name;"
    cin>>n;
}
void put( )
{
    cout<<"n\n roll no”<<rno;
    cout<<"n\n name”<<name;
}
};
class marks
{
public:
int m1,m2,m3;
void getmarks( )
cout<<"n enter marks of 3 subjects";
cin>>m1>>m2>>m3;
}

void purmarks( )
{
    cout<<"n Marks in m1="<<m1;
    cout<<"n Marks in m2="<<m2;
    cout<<"n Marks in m3="<<m3;
}
}

class result : public stud, public marks
{
    int tot;
    float per;
public:
    void cal( )
    {
        get();
        getmarks( );
        tot=m1+m2+m3;
        per=(float)tot/3;
    }
    void putresult( )
    {
        put( );
        putmarks( );
        cout<<"n total="
            <<tot;
        cout<<"n percentage="<<per;
    }
};

void main( )
{
c. Multilevel Inheritance:

The appliance of deriving class from another derived class is called as multilevel inheritance. As shown in above figure class A has serve as a base class for the derived class B, the class B in term served as a base class for the derived class C. Since it provides a linked for inheritance between A and C. The chain A-B-C is called as path of inheritance.

Syntax:

Class A
{
    .......
    .......
};
Class B : visibility mode A
{
    .......
    .......
};
Class C : visibility mode B
Consider that the test result of a batch student stored in the three separate classes the class student stored the roll no, class test stored the marks getting in the test and class result can inherit the details of the marks obtain in the test and roll no of student through in multilevel inheritance.

d. **Hierarchical Inheritance:**

![Fig-16: Hierarchical inheritance](image-url)
When the properties of single base class may be inherited by more than one derived classes this process is called as hierarchical inheritance. It support hierarchical design of problem i.e. many program problem can be coast into hierarchical. Where certain feature of single level are used by many other classes below than level.

Hierarchical classification of student in university as follows.

![Hierarchical Inheritance Diagram]

The base class can contain all the characteristic that are same to subclasses. A subclass can be created by inheriting the functionality of base class. A subclass can be treated as a base class for the classes at lower level.

Eg. :

```
Class emp
{
    int emp_no;
    char name1[20];
    public:
    void get()
    {
        cout<<"n enter emp no";
        cin>>emp_no;
        cout<<"n enter emp name";
        cin>>name1;
    }
```

Fig-16.1: Hierarchical inheritance for student faculty
void put()
{
    cout<<"n emp no="<<emp_no;
    cout<<"n emp name="<<name1;
}
}
class manager : public emp
{
    int dept_no, sal;
public:
    void input( )
    {
        cout<<"n enter dept no";
        cin>>dept_no;
        cout<<"n enter salary";
        cin>>sal;
    }
    void output( )
    {
        cout<<"n Dept no="<<dept_no;
        cout<<"n Salary="<<sal;
    }
};
class worker : public emp
{
    int sal1;
public:
    void get1( )
    {
        cout<<"n enter worker salary";
        cin>>sal1;
    }
    void put1( )
    {
cout<"\n Worker Salary="<<sal;
}
}
void main( )
{
    manager m;
    worker w;
    m.get( );
    m.input( );
    m.put( );
    m.output( );
    w.get1( );
    w.put( );
    w.put1( );
}

e. Hybrid Inheritance:

![Diagram of Hybrid Inheritance]

Fig-17: Hybrid inheritance

The mixture of two or more inheritance together is called as hybrid inheritance. In some occasion we need to apply two or more type of inheritance to design a problem such inheritance is called as hybrid inheritance. In given figure we need two types of inheritance together as hierarchical inheritance and multiple inheritances.

Eg. :-
Suppose we consider the case of storing or adding student result. Then assuming that weightage for sports we have consider before finishing the result the weightage of sport is stored in sports class as shown in following figure.

```
Class stud
{
    int rno;
    public:
    void getdata( )
    {
        cout<"n enter roll no";
        cin>>rno;
    }
    void putdata( )
    {
        cout<"n Roll no="<<rno;
    }
};

class test : public stud
{
```
int marks;
char sub[10];
public:
void getdata1( )
{
    stud : : getdata( )
    cout<<"\n enter subject name";
    cin>>sub;
    cout<<"\n enter marks;"
cin>>marks;
}
void putdata1( )
{
    stud : : putdata( );
    cout<<"\n name of subject="<<sub;
    cout<<"\n subject marks ="<<marks;
}
};
class sport
{
    int marks;
    int id_no;
    public:
    void getdata2( )
    {
        cout<<"\n Enter Id_no"
        cin>>id_no;
        cout<<"\n enter the sport marks";
        cin>>marks;
    }
    void putdata2( )
    {
        cout<<"\n id_no="<<id_no;
        cout<<"\n marks="<<marks;
    }
class result : public test, public sports
{
    char grade;
    public;
    void get()
    {
        test :: getdata1();
        sport :: getdata2();
        cout <<"\n enter grade";
        cin >> grade;
    }
    void put()
    {
        test :: putdata1();
        sport :: putdata2();
        cout <<"\n grade=" << grade;
    }
};
void main()
{
    cout << "\n Details of stud"
    result r;
    r.get();
    r.put();
    getch();
}

1.7.3 Polymorphism
An object of several types to respond separately to the same method call
given mechanism is known as polymorphism. Polymorphism can be stated as, Poly
means "more" and morph means "form." The object oriented system context,
objects that can consider or suppose to be more separated forms. They behave
differently on different classes with similar operation is a polymorphism. The
author named Booch said Some common super class consider polymorphism as the relationship of objects of many several classes. Some common set of operations in a different way that any of the objects called by this name is able to replied. Assume an example, in a payroll structure, manager, office worker, and production worker objects.

The primitive polymorphisms are overloading and template because determination of calling a needed function is made at compile time rather than at run time. The specific detail cannot be finding the nature of some object sat compile time. Sometimes such objects to be delayed until runtime where determinations on which function to call will be present. The given technique that gives true polymorphism. Different data types have in ordinary and write code works similarly is the ability of polymorphism.

Polymorphism is a Greek letter that firmly means more shapes. It is tightly joined to inheritance is polymorphism; it is often cited differently as one of the most important benefit to object-oriented technologies.

Fig-18.1: Polymorphism attribute
At the time when a message is send to an object, the object respond to that message must have a method defined. The inheritance hierarchy contain, every subclasses inherit the interfaces from their super class. However, because every subclass is a different entity, everyone needs a different reply to the similar message. Suppose we consider for example, consider the shapes class and the behavior called Draws. When someone tells to draws shapes, the starting question asked is, “What shapes?” [30].

Suppose we consider an array of three shapes—Cir, Sqr, and Star. Even all they treated as Shapes objects, and sending a Draw message to each Shapes object, the end result is separate for every because Cir, Sqr, and Star supplied the current execution. In other words we say that every class is able to reply separately to the similar draw method and draw itself. This is known as polymorphism.

Following class shows Shapes class:

```java
public abstract class Shapes {
    private float ara;
    public abstract float getAra();
}
```

The Shapes class has an attribute called ara that stored the value for the area of the shape. The function getAra() contain an identifier called abstract. A subclass must supply the utensil for this method, when a method is defined as abstract; in that case, Shapes is needing subclasses to provide a getAra() appliances. Now let’s build a class called Circal that inherits from Shapes

```java
public class Circal extends Shapes {
    float rad;
    public Circal(float r) {
        rad = r;
    }
    public double getAra() {
        ara = 3.14*(rad*rad);
        return (ara);
    }
}
```
Constructor is new concept here we introduced. The Circal class has a method with the similar name, Circal. At that time when a function or method name is the same as the class name and no return type is supplied, then the method is a special known as a constructor. Here we consider a constructor as the starting point for the class, where the object is created and the constructor is a better point to perform initializations of variables and start-up tasks for elements. The Circle constructor accepts a one parameter, coinciding the radius, and apply it to the radius attribute of the Circal class. The Circal class also supplies the implementation for the getAra method or function, purely defined as abstract in the Shapes class. We can create a same class, called Rect:

```java
public class Rect extends Shapes {
    double leng;
    double wid;
    public Rect(double l, double w) {
        leng = l;
        wid = w;
    }
    public double getAra() {
        ara = leng*wid;
        return (ara);
    }
}
```

Now we can building any number of rects, circles, and so on and called their getAra() method. We know that all rectangles and circles inherit from Shapes, and all Shapes classes have a getAra() method.

Thus, we can create object of the Shapes classes in this way:

```java
Circal cir = new Circal(6);
Rect rec = new Rect(7,8);
```
1.7.4 What Is a Stack?

A stack is a structure of data that is a last-in, first-out system. The stack is a structure to show first element is inserted at the top of the stack and we consider an example of cylinder inserting a coin to the top of the cylinder, when you need the coin simply we take from the top element coin that we lastly inserted.
If we want to adding the elements or items to the stack, that is to adding element to the top of the stack. Popping or removing an item off the stack means that we are taking the last top item off the stack.

The stack we can empty the stack, and we don’t have to tension about what kind of Shapes classes are in it and we only knows there is shape present.

```java
while ( !stk.empty())
{
    Shapes shape = (Shapes) stk.pop();
    System.out.println("Area = “ + shape.getAra());
}
```

In reality, we are sending the same message to all the shapes:

```java
shape.getArea()
```

This approach is meant to provide standardization across classes, as well as applications. Consider an office suite that includes a word processing and a spreadsheet application. Let’s assume that both have a method called Print. This Print method can be part of the Office class as a requirement any class that inherits from it to implement a Print method. The interesting thing here is that although both the word processor and spreadsheet do different things when the Print method is invoked, one prints a processing document and the other a spreadsheet document [32]. In above figure the stack shows the elements as string type args array and employee objects. The stack inserted the elements differently in the form of heap as stack structure.

### 1.8 Methodology for object-oriented systems development

Contrast model from the customary software development approach, which is depend on functions and procedures offers by object oriented development. Building self-contained modules or objects that can be easily transfer and reused is object-oriented systems development way to develop software.

Object-oriented surroundings present as,

- A group of individual objects that combine their data as well as the utilitarian to real-world model objects is software.
- An object orientation contain essential advantages to the practice of software building.
- Attributes and functions are present in every object.
- In object-oriented terms; objects are grouped into classes, the classes specifies in the problem domain.
Each object is responsible for itself and everything is an object.

Suppose we consider an example, the Windows operation requires Windows instances. They responsible or things like opening and closing itself is a Windows object. Certainly when a window shows something and it is an object. Here we consider an chart object as example. A object chart is liable for things like preserving its labels and data. They enables to create group of instances that works combine to generate software that good model their problem domains than system created using traditional techniques and even better method is object oriented methods. Object-oriented growingness allows us to build functionality modules. It is easier to preserve, more robust, easier to taking to transferring requirements, good design and code reuse by the system.

1.8.1 Programming Styles

A particular style of programming is forces every programming languages. Programming paradigm can be defined as style of programming is mixing by organizing information. The languages used for numeric computations such as Fortran, ALGOL is considered in First Generation Programming languages. The variable or a constant in a program can be represent data for the purpose of use of data. The operator acts on the data to perform an action. The purpose to form an expression with the combination of operands and operator. Every instruction is create with help of expressions is written. The step by step sequence of statements creates a program. The 1st Generation Language Structure is shown in Fig.

![Fig-20: 1st Generation Language Structure](image)

Monolithic programming is programming that there is no support for subprograms. There is no occurrences of data hiding because of data is available globally. Data hiding is can be defined as to denying the access of data. The simple applications were used by first generation languages. To representing the functions in the programming language that can be performed in the computer that closer to the solution domain.
The subprogram that is functions or procedures mostly introduced in Second Generation Programming Language. The following figure shows Data and subprogram.

The subprograms will neglects the consistency of include coding such programming is known as procedural programming language. The applications that are required small sized programs that used to suit second generation language. COBOL, FORTRAN and ALGOL 60, are second generation languages. The possibility of information hiding should be provided in the second generation language. That hides the building details of a subprogram. The data hiding principle is broken by the use of sharing same data many subprograms. Where eye to the aim is on operations using functions is closer to the solution domain [35].

The languages like C and Pascal use local data, sequential code, global data and subprograms is considered in third generation programming languages.

The modular programming is supported by following structure programming. The number of modules the program can be divided. Every module existing in a several number of subprograms, represented by ellipse. The approach which develop algorithm is known as algorithmic oriented programming. Data are transfer between two devices across the network named as data transfer. A computer files can send as attachments in mail to clients, associates and friends. It can be easy download by users on website which can be uploaded on server.

The data and subprograms exist separately, in structural programming approach,

Programs=Algorithm + Data Structure

As shown in following fig.
At the situation occurs main program calls the subprograms. The following feature supports structured programming approach.
1. Its self local data and algorithm present in every procedure.
2. Scope and visibility of data are introduced.
3. Argument-supplying mechanisms are existing.
4. There is possibility to create or build user-defined data types.
5. The rich set of control structures is provided.
6. Every procedure is not dependent of other procedures.
7. It supported nesting of subprograms.
8. Function or method essentials are gain, including abstract operations.
9. Supporting modular programming that subprograms are the basic physical building blocks.

Data hiding should be possible by providing scope of variables. The sustentation of the program becomes very difficult task because consisting of different subprograms and global data.
1.8.2 Pseudocode:

It is programming analysis tools used for planning program logic. Pseudocode is a code which is used to give clear description of the algorithm. The code refers to the writing an instruction and false is pseudo. In ordinary natural language can be used to pseudo instructions are phrases written, like as English, French, German etc.

Pseudo is made of above that have been proved simple logic structures to be limiting for writing any computer program.

1) Sequence

2) Selection

3) Iteration

Advantages of pseudo Code:

i) Is easier as compared to flowchart than Converting pseudo to programming language.

ii) It’s easy to write pseudocode as compared to drawing flowchart. Flowchart require more time and efforts. Pseudo have few rules to follow.

Limitations of pseudocode:

i) There is not available graphic representation of program logic in pseudocode.

ii) As there are no standard rules for writing pseudocode, different programmers may use their own style, which may cause communication problem.

1.9 Object-Oriented Programming Paradigms and Software Development

It can be seen that in structured programming, the aim is on the subprograms and the better way of developing algorithms in location of computing time and to clarify the problem use computer memory. The programmer and program relationship is given priority first importance.

As shown in Fig.
Develops Fig-23: A Program and Programmer Relationship

It must be depending on the result domain and not on the difficulty that structured programming paradigms. Regarding access permission the data is not given importance. First the complicated problem is separated into tiny difficulties to solve a complicated problem using the approach top-down. Afterward these tiny problems are divided and a grouping of little problems is carried out left. Every problem clarified one at a time. The problem representing global functionality of structured programming starts with high-level descriptions. By dividing it into subprograms using lower level descriptions to repeatedly refines the global functionality. A data structure is refined at each step, either a control. The structured programming followed the top down approach. It will create difficulties only when there is a repetition of design phase is a fairly successful approach. Such repetitions may solution in extra changes in the program. Reuse of software modules is reduced possibly. Software industries were unable to meet the need in reality in software engineering and despite the invention of new programming languages. Only a few of them were used in software development in many programming languages evolved.

1.10 Objectives

1] To study on object oriented programming concepts.

3] To study and analyze difficulties in object oriented programming.

4] To study of understanding problems of object oriented programming to students, teachers.

5] Our research can take total 500 samples from 20 colleges for analysis purpose and divided these samples in two parts, 50% samples take in rural and take 50% samples in urban area.

6] To study and analyze effectiveness of E-learning in teaching methods.
7] To need of E-learning to solve the given problem facing by institutions, teachers and students.

8] Through our research we can describe in details all types of difficulties in OOP

1.11 Virtual classroom tools are implemented:

Aim to our propose research is to modify the education system in rural and urban education. The research is not only limited upto modify education system but also this education is helpful for difficulty occurs in OOP student in rural area. Our research is broadly categorise into two phase in existing research only find out the difficulties OOP of student in rural area and urban and many researcher identified the learning difficulties only. But in our research we identified and study some cases. Difficulties is not only neurologic base it is also infrastructure and educational infrastructure and educational environment base impact of educational infrastructure are improve the student intrest in education.

Virtual classroom enhance technology to improve education system. Virtual classroom are use to solve infrastructure based difficulties. This topic can be discuss in next chapter but when we introduce the role of internet computing in education, it is important to describe the features of virtual classroom. Virtual classroom is interactive method of learning VC provide the interactive environment for student and instructor and both are interact via computer. Student and trainer can access resource via mobile device that has internet access. We create virtual classroom using multimedia, this classroom is really different than regular classroom but in rural area student can learn in open space or slum type classroom. In today it is important to enhance education in rural and urban area to solve difficulties. In virtual classroom student learn via internet including learning content that is hosted on the internet E-learning is mixture of multimedia, computer machine and network technology in learning process.

❖ CEL architecture (Gagandeep Kaur et al, 2014)

CEL architecture is based on cloud computing and web 2.0. this architecture provide reliable and secure educational services for institution there are multiple new advances in computer based technology these days, so we utilize cloud computing and Web 2.0 to create, implement, test and use our E-Learning application for student, trainer and faculty. This present cloud computing architecture for E-learning consists
of three prime layers: Cloud Model Layer, Service Model Layer and Application Model layer.

1.12 Scope of research study

1] Scope of our research is very broadly useful for students, teachers and developers that have difficulties in learning OOP.
2] Scope of our research is very useful for professional and non-professional users in programming languages. Especially for those who totally unknown about Object Oriented Programming.
3] Our research is helpful for structured oriented programming background students.
4] Our research can solve the problems faced by colleges.
5] With the use of our research, students can learn all concepts of object oriented programming.
6] We will give all possible analysis of difficulties object oriented programming.

1.13 Limitations

1] We mentioned all over Object Oriented programming concepts but some of them not cover by our research.
2] Our research had some limitation because E-learning based on internet facilities.
3] Some problem arises to have no knowledge about internet resources.
4] Structured oriented programming background users facing problems.