

# Object Oriented Programming with C++

Introduction and Course details

Lecturer : Astha Sharma

# Objectives



To familiarize with  
Object Oriented Concept

To introduce the  
fundamentals of C++

To enable the students to  
solve the problems in  
Object Oriented tec

To cope with features of  
Object Oriented  
Programming

# CHAPTERS

1.

Thinking  
Object  
Oriented

2.

Classes and  
Methods

3.

Message,  
Instance and  
Initialization

4.

Object  
Inheritance  
and  
Reusability

5.

Polymorphism

6.

Template and  
generic  
programming

7.

Object  
oriented  
Design

# Revision

## Programming with C

## **Introduction**

History of computing and computers, programming,  
Block diagram of computer,  
Generation of computer,  
Types of computer, Hardware  
Software, Programming Languages, Traditional and  
Structured programming concept

## **Programming logic**

Problems solving(understanding of problems, feasibility and  
requirement analysis)  
Design(flow Chart & Algorithm),  
Program coding (execution, translator),  
Testing and Debugging,  
Implementation, evaluation and Maintenance of programs,  
Documentation

## **Variables and data types**

Constants and variables,  
Variable declaration,  
Variable Types,  
Simple input/output  
function, Operators

## **Control Structures**

Introduction,  
Types of control statements - sequential, branching, repetition  
- if, else, else-if and switch statements,  
Case, break and continue statements;  
Looping- for loop,  
While loop,  
Do—while loop,  
Nested loop,  
Goto statement



## **Arrays and Strings**

Introduction to arrays,  
Initialization of arrays,  
Multidimensional arrays,  
String,  
Function related to the strings

## **Functions**

Introduction,  
Returning a value from a function,  
Sending a value to a function,  
Arguments,  
Parsing arrays and structure,  
External variables,  
Storage classes,  
Pre-processor directives,  
C libraries, macros, header files and prototyping



## **Pointers**

Definition, pointers for arrays,  
returning multiple values from functions using pointers,  
Pointer arithmetic,  
Pointer for strings,  
Double indirection,  
Pointer to arrays,  
Memory allocation - malloc and calloc

## **Structure and Unions**

Definition of Structure,  
Nested type Structure,  
Arrays of Structure,  
Structure and Pointers,  
Unions,  
Self-referential structure





## **Files and File Handling**

Operating a file in different modes (Read, Write, Append),  
Creating a file in different modes(Read, Write, Append)

COBOL



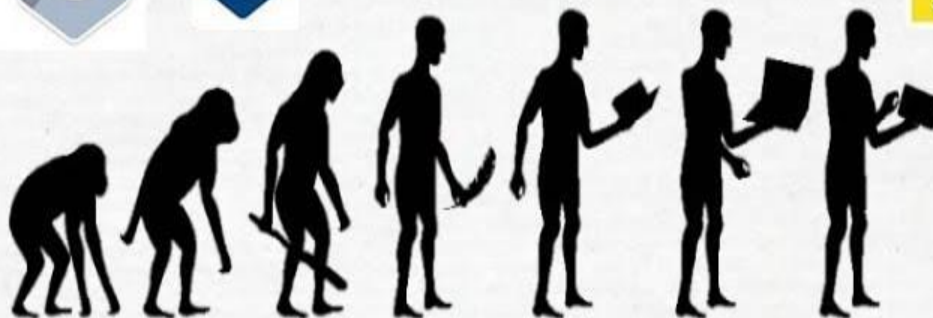
EVOLUTION



php



Java



c c++ java python go dart swift



Dart



Swift

# CHAPTER 1

## Thinking Object Oriented

# Computer Programing Languages

Generation	Discussion
First Generation	Machine Language - directly executable by the computer
Second Generation	Assembly Language - low-level language similar to machine language but more understandable by humans, is transformed to machine language by an assembler program
Third Generation	High-level language, which is procedural in format, is transformed into machine language by a compiler, examples include COBOL, FORTRAN, Pascal
Forth Generation	High-level language, which is non-procedural, meaning that the programmer specifies what to do, not how to do it; most fourth generation languages are interpreted where source code is changed to machine code on a line-by-line interactive basis
Fifth Generation	The evolving languages that are based on natural languages such as English; the user can "talk" to the computer as one would to another person

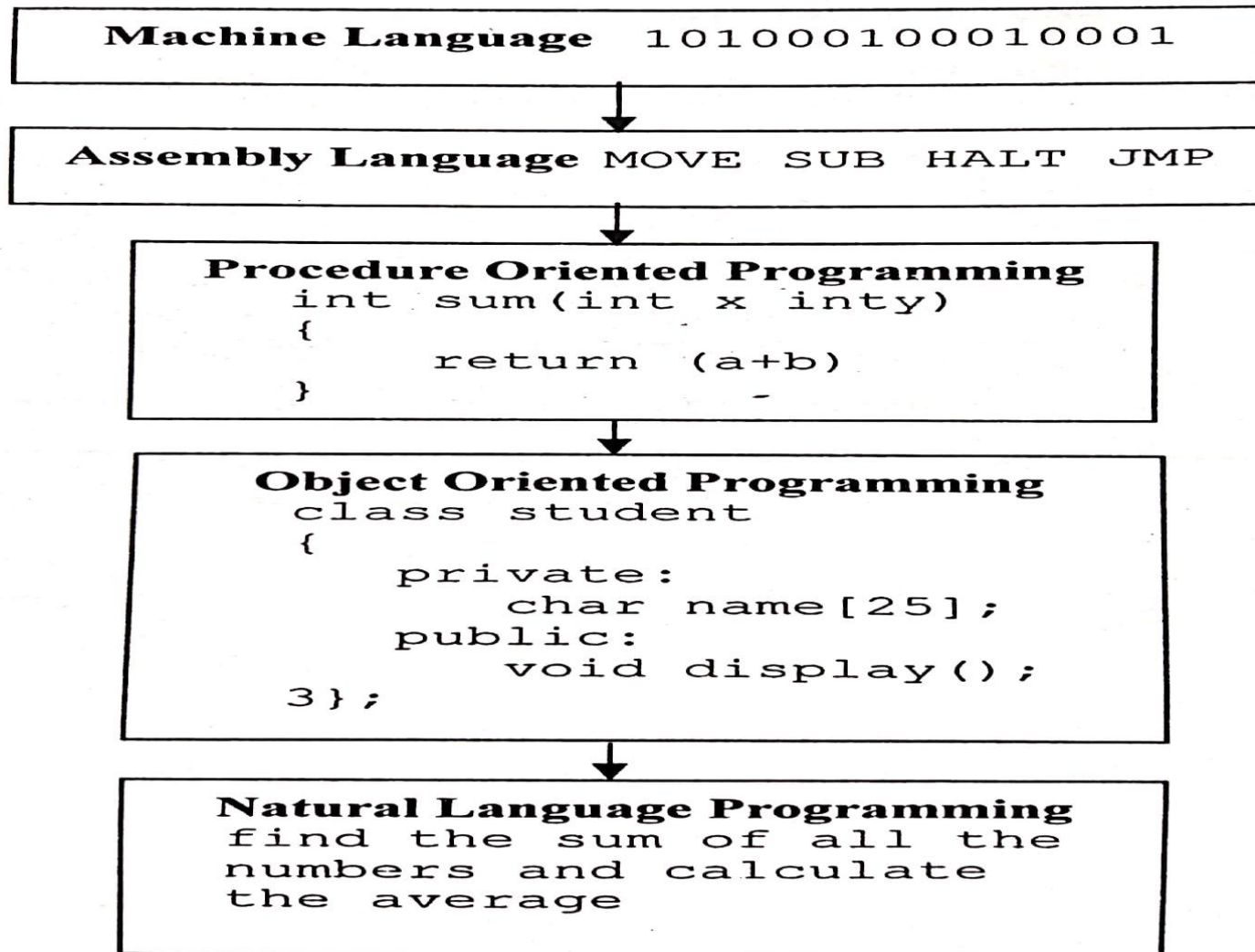


Figure: Phases in history of Software Development



# Procedural Oriented Programming

- Fundamentally, the procedural code is the one that directly instructs a device on how to finish a task in logical steps.
- This paradigm uses a linear top-down approach and treats data and procedures as two different entities.
- Based on the concept of a procedure call, Procedural Programming divides the program into procedures, which are also known as routines or functions, simply containing a series of steps to be carried out.
- Simply put, Procedural Programming involves writing down a list of instructions to tell the computer what it should do step-by-step to finish the task at hand.

# Example of POP

Program:


```
#include <stdio.h>
int main()
{ /* printf function displays the content that is * passed
between the double quotes. */
printf("Hello World");
return o; }
```

Output:

Hello World

1. **#include <stdio.h>** – This statement tells compiler to include this `stdio.h` file in the program. This is a standard input output file that contains the definitions of common input output functions such as `scanf()` and `printf()`. In the above program we are using `printf()` function.
2. **int main()** – Here `main()` is the function name and `int` is the return type of this function. Every C program must have this function because the execution of program begins with the `main()` function. The 0 return value of this function represents successful execution of program while the return value 1 represents the unsuccessful execution of program. This is the reason we have `return 0;` statement at the end of this `main` function.



- 
3. **printf("Hello World");** – This function displays the content within double quotes as it is on the screen.
  4. **return 0;** – As mentioned above, the value 0 means successful execution of `main()` function.

# Advantages

- Procedural Programming is excellent for general-purpose programming
- The coded simplicity along with ease of implementation of compilers and interpreters
- A large variety of books and online course material available on tested algorithms, making it easier to learn along the way
- The source code is portable, therefore, it can be used to target a different CPU as well

# Advantages

- The code can be reused in different parts of the program, without the need to copy it
- Through Procedural Programming technique, the memory requirement also slashes
- The program flow can be tracked easily

# Disadvantages

- Difficult to relate with real-world objects
- The importance is given to the operation rather than the data, which might pose issues in some data-sensitive cases
- The data is exposed to the whole program, making it not so much security friendly
- The Procedural code is often not reusable, which may pose the need to recreate the code if it is needed to use in another application

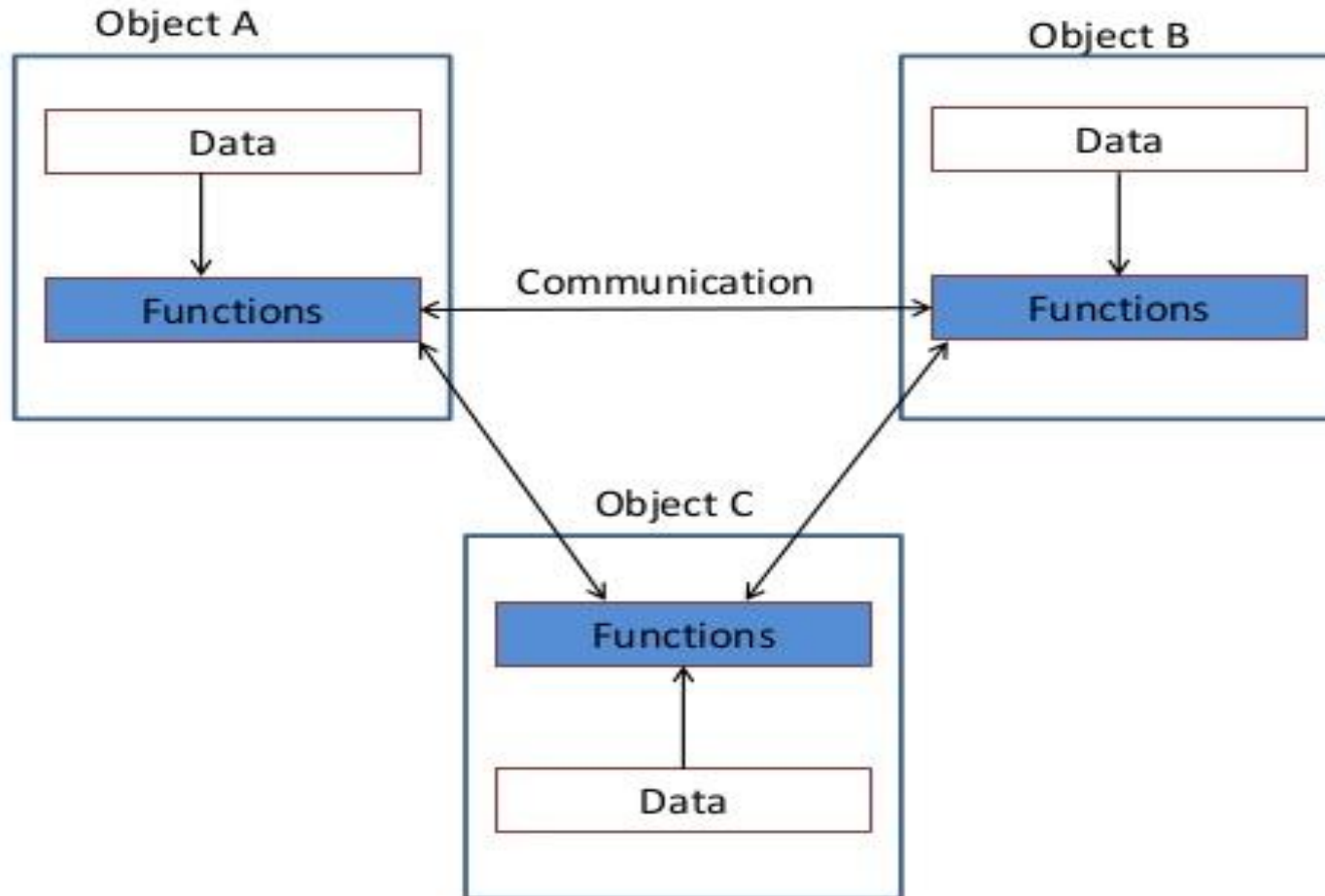
# Object Oriented Programming

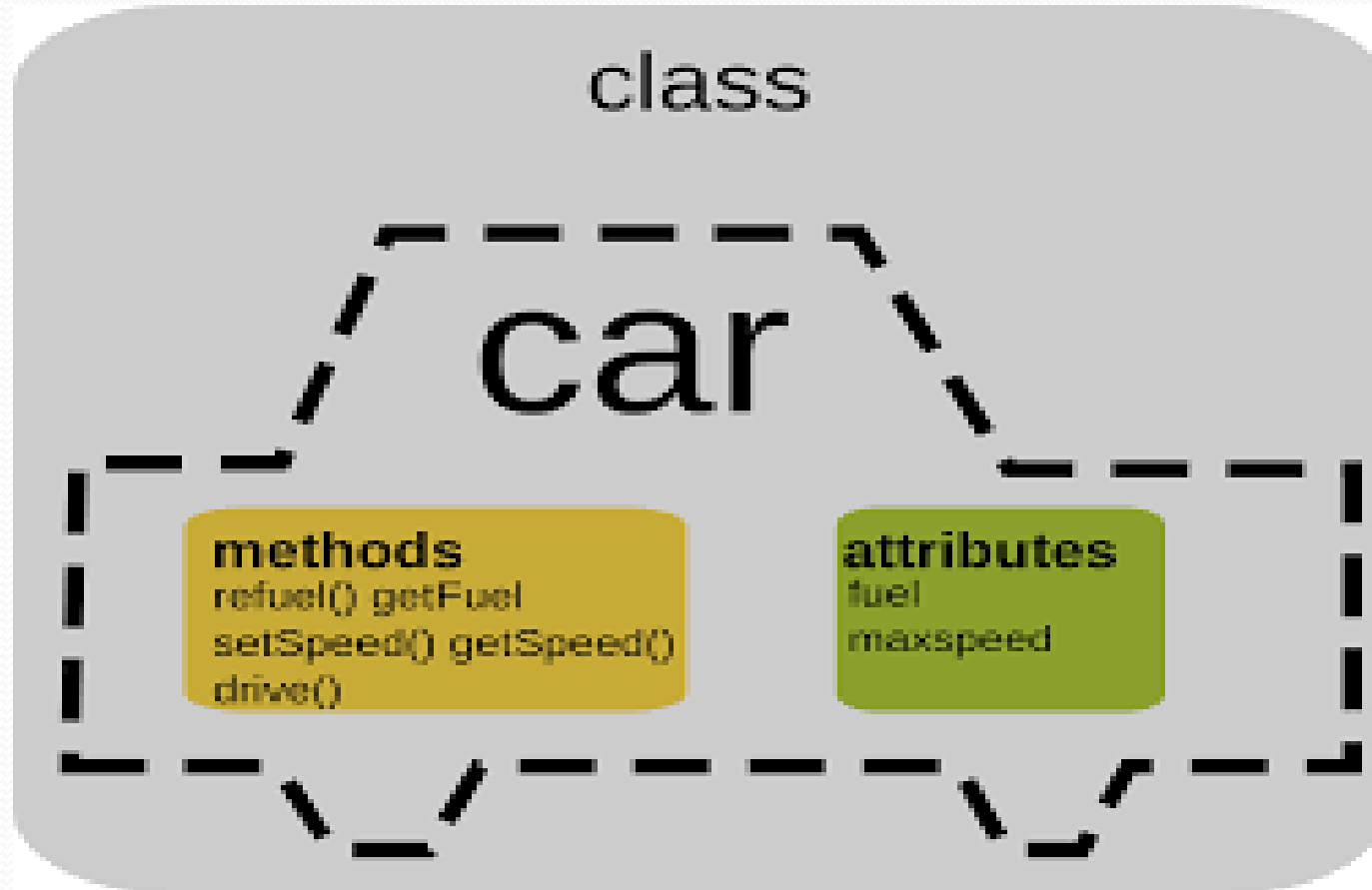
- OOP is an approach to programming which recognizes life as we know it as a collection of objects, which work in tandem with each other to solve a particular problem at hand.
- Invention of OOP is to remove the flaws of POP.
- The core of the pure object-oriented programming is to create an object, in code, that has certain properties and methods.
- While designing C++ modules, we try to see whole world in the form of objects.
- For example a car is an object which has certain properties such as color, number of doors, and the like. It also has certain methods such as accelerate, brake, and so on.

# Object Oriented Programming

- The primary thing to know about OOP is encapsulation, which is the idea that each object which holds the program is self-sustainable, which means that all the components that make up the object are within the object itself.
- Now since each module within this paradigm is self-sustainable, objects can be taken from one program and used to resolve another problem at hand with little or no alterations.
- OOP treats data elements as a critical element and doesnot allow to move freely.
- It ties data more closer to the function and protect it from accidental modification from outside the functions called object.

# Object Oriented Programming







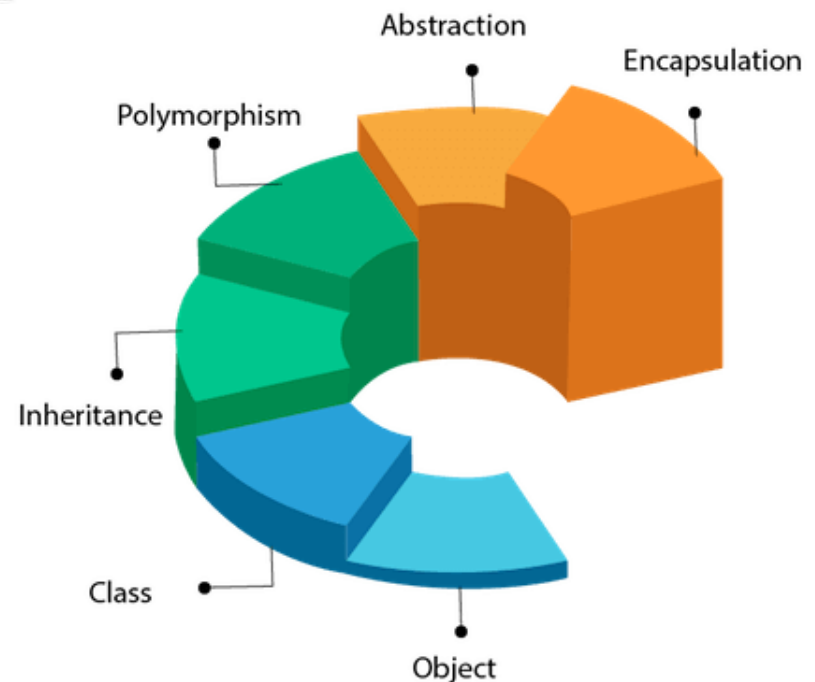
# Object Oriented Programming

## Characteristics of OOP

- Emphasis is on data rather than procedure.
- Programs are divide into objects.
- Data is hidden and cannot be accessed by external functions.
- Objects can communicate with each other through functions.
- New data and functions can be easily added whenever necessary.
- Follows bottom-up approach in program design.

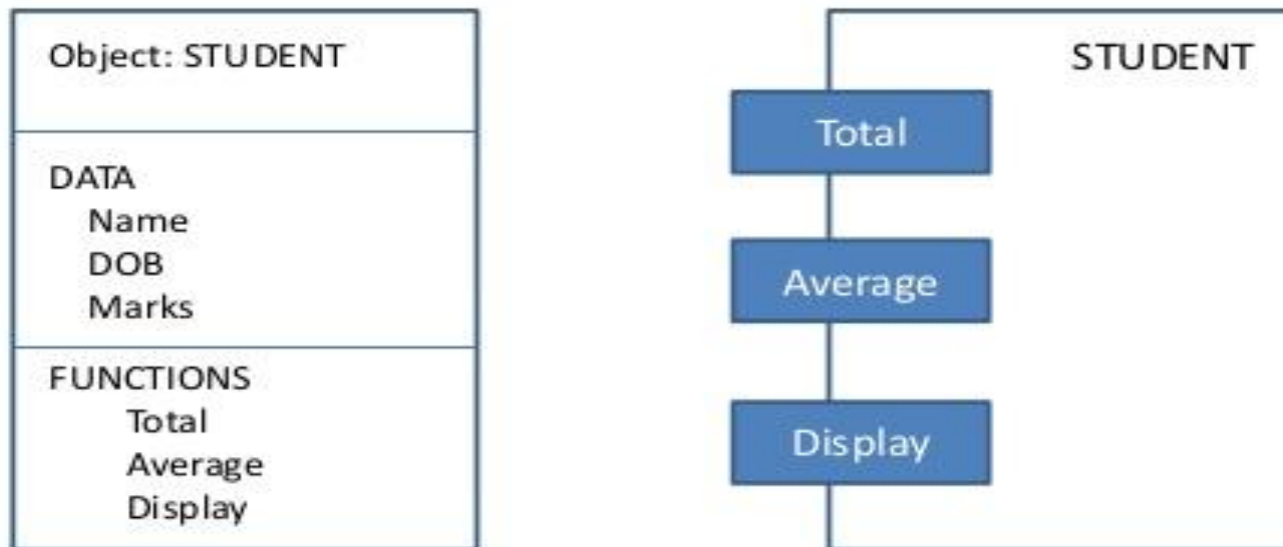
# Features of Object Oriented Programming(OOP)

- Objects
- Classes
- Data Abstraction and Encapsulation
- Inheritance
- Polymorphism
- Dynamic Binding
- Message Passing



# Objects:

- This is the basic run-time entities in an object oriented programming.
- That is both **data** and **function** that operate on data are bundled as a unit called as **object**.
- Program objects should be chosen such that they match closely with the real- world objects.
- Objects take up **space in the memory** and have an associated **address like structure in C**.



Two ways of representing an object



# Classes

- The entire set of data and code of an object can be made a **user-defined data type** with the help of a **class**.
- Objects are **variable of type class**.
- Once a class has been defined we can create any number of **objects belonging to that class**.
- A class is thus a collection of **objects of similar type**.
- Classes **are user-defined data types** and behave like the built-in types of a programming language.  
e.g: Fruits mango , apple.



**Class name**

**Class Vehicle**

**Data Members**

Model  
Color  
Registration

**Member Functions**

Forward()  
Backward()  
Stop()

# Class



Breed: Bulldog  
Size: large  
Colour: light gray  
Age: 5 years

Dog1Object



Breed: Beagle  
Size: large  
Colour: orange  
Age: 6 years

Dog2Object



Breed: German Shepherd  
Size: large  
Colour: white & orange  
Age: 6 years

Dog3Object

Dog

## Fields

Breed  
Size  
Colour  
Age

## Methods

Eat()  
Run()  
Sleep()  
Name()

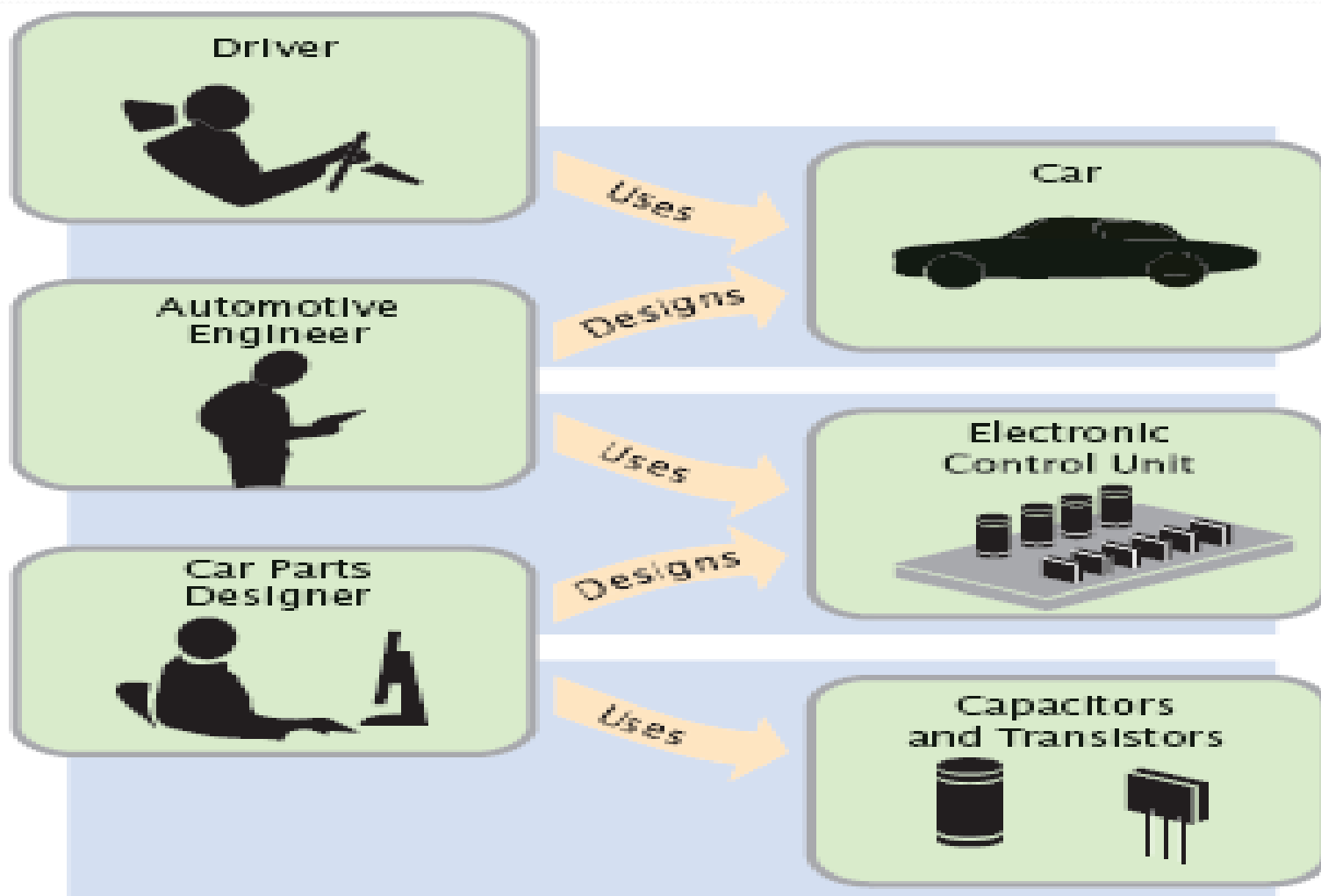
# Data Abstraction and Encapsulation

- The wrapping up of data and function into a single unit (**called class**) is known as *encapsulation*.
  - Data and encapsulation is the most striking feature of a class.
  - The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.
  - Functions provide the interface between the object's data and the program.
  - Insulation of the data is called *data hiding or information hiding*.
- 
- Abstraction refers to the act of representing essential features without including the background details or explanation.
  - Classes use the concept of abstraction
  - The **attributes** are some time called *data members* and The **functions** that operate on these data are sometimes *called methods or member function*.





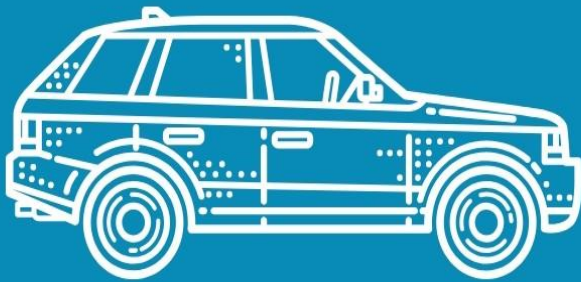
**Real Life Example of Abstraction**



**Figure 1**

Levels of Abstraction in Automotive Design

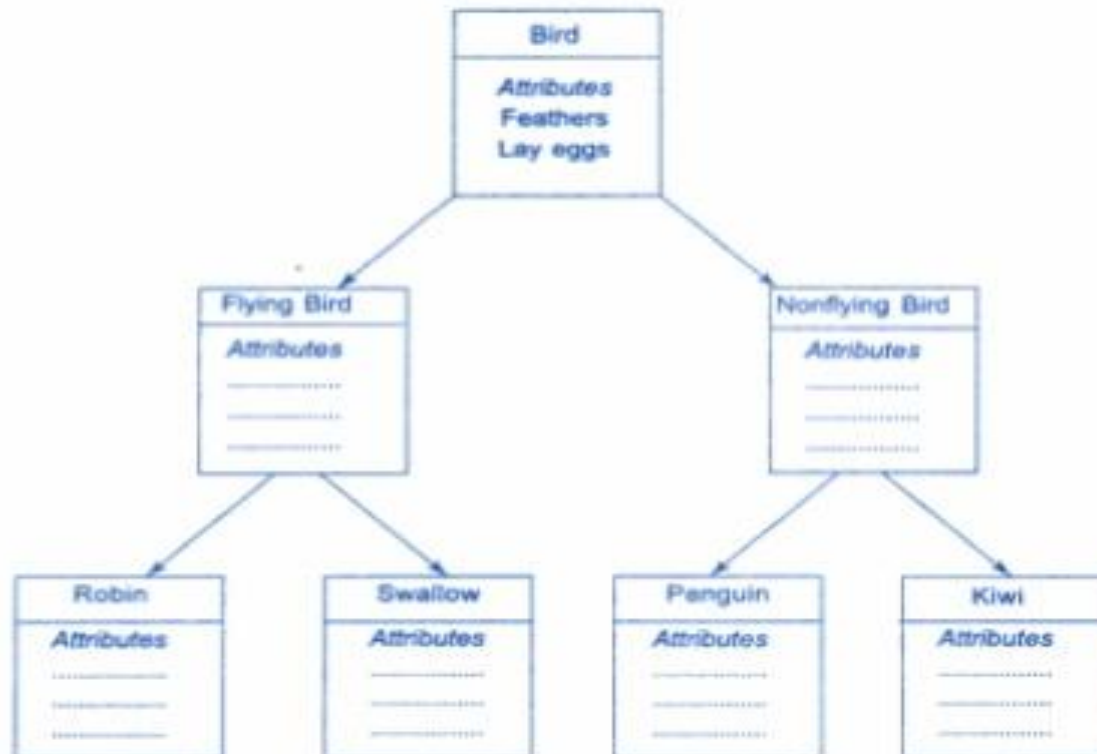
# Encapsulation



```
Car
model
speed
engine
speedLimit
drive()
stop()
setSpeed(number)
```

# Inheritance

- This is the process by which a class can be derived from a base class with all features of base class and some of its own.
- This increases code reusability.

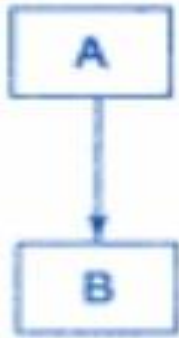


# Types of Inheritance

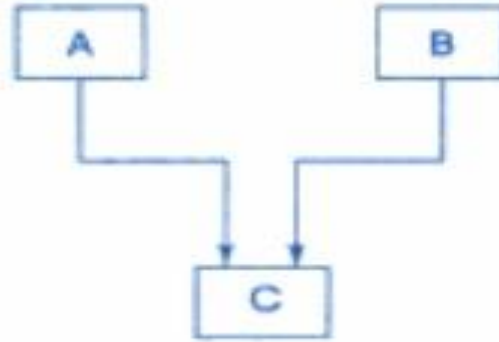
In C++, we have 5 different types of Inheritance.  
Namely,

1. Single Inheritance
2. Multiple Inheritance
3. Hierarchical Inheritance
4. Multilevel Inheritance
5. Hybrid Inheritance (also known as Virtual Inheritance)

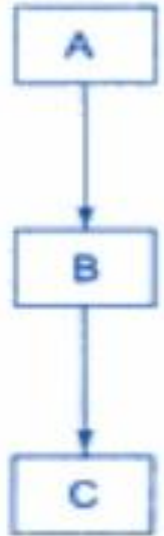
# Types of Inheritance



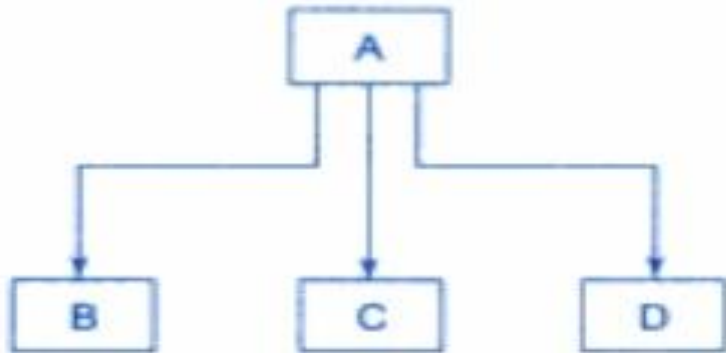
Single inheritance



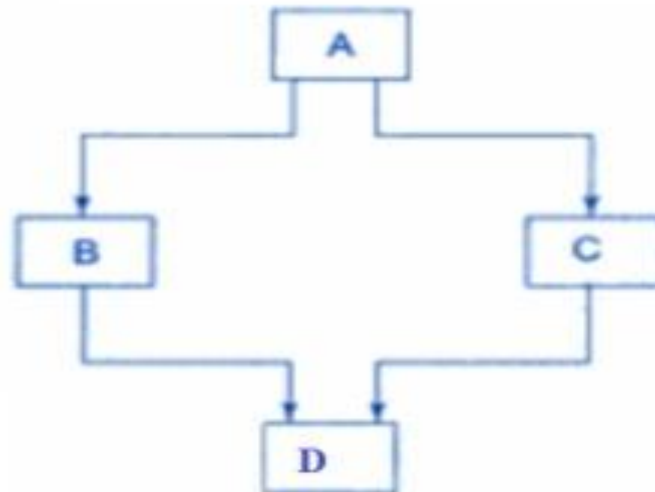
Multiple Inheritance



Multilevel Inheritance



Hierarchical Inheritance



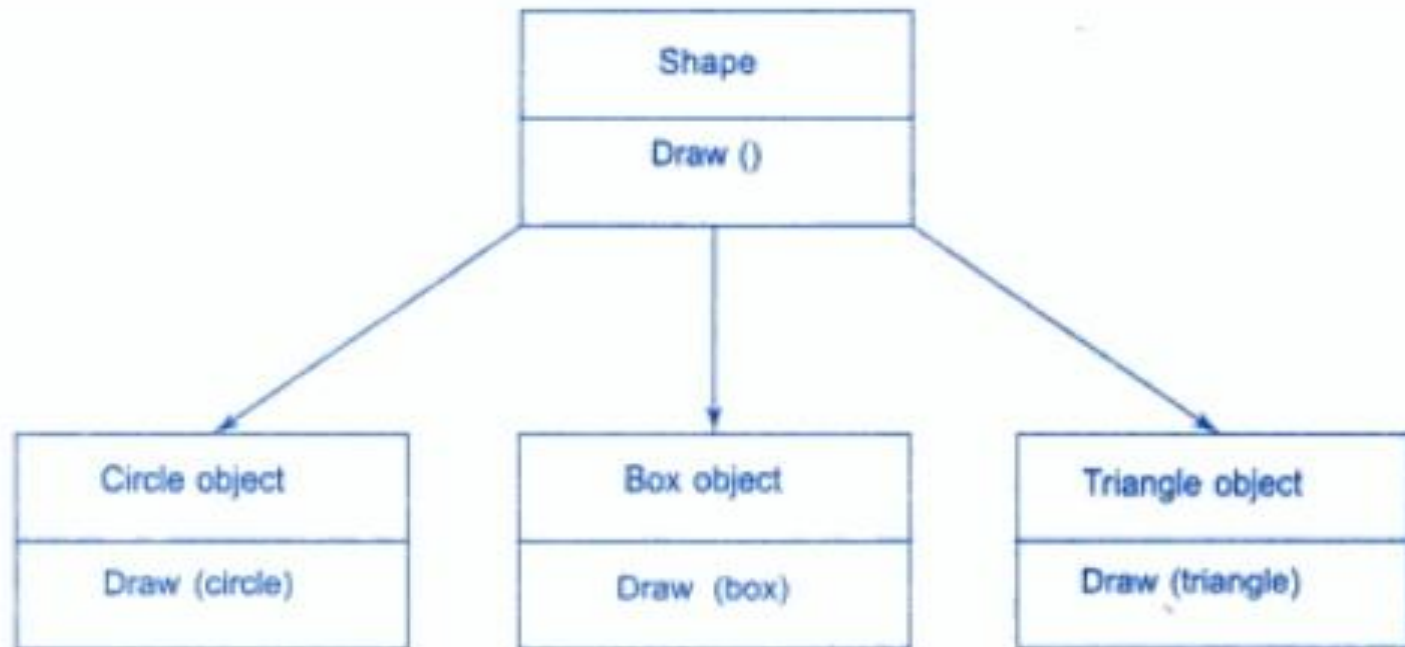
Hybrid Inheritance

# Polymorphism

- *Polymorphism is another important OOP concept .*
- *Polymorphism means the ability to take more than one form.*
- An operation may exhibit different behavior in different instances.
- *The behavior depends upon the types of data used in the operation.*
- The process of making an operator to exhibit different behaviors in different instances is known as *operator overloading*.
- Using a single function name to perform different type of task is known as *function overloading*.



# Polymorphism



Example of Polymorphism





**Polymorphism**



# Dynamic Binding

- Binding refers to the linking of a procedure call to the code to be executed in response to the call.
- Dynamic binding also known as **late binding**, means that the code associated with a given procedure call is not known until the **time of the call at Run-Time**.

# Message Passing

- A message for an object is a request for execution of a procedure .
- Invoke a function(procedure) in the receiving object that generates the desired result.
- Message passing involves :
  1. Specifying name of the object.
  2. Name of the function(message).
  3. Information to be sent.



# Advantages

## 1. Re-usability

It means reusing some facilities rather than building it again and again. This is done with the use of a class. We can use it 'n' number of times as per our need.

## 2. Data Redundancy

This is a condition created at the place of data storage (you can say Databases) where the same piece of data is held in two separate places. So the data redundancy is one of the greatest advantages of OOP. If a user wants a similar functionality in multiple classes he/she can go ahead by writing common class definitions for the similar functionalities and inherit them.

# Advantages

## 3. Code Maintenance

This feature is more of a necessity for any programming languages, it helps users from doing re-work in many ways. It is always easy and time-saving to maintain and modify the existing codes with incorporating new changes into it.

## 4. Security

With the use of data hiding and abstraction mechanism, we are filtering out limited data to exposure which means we are maintaining security and providing necessary data to view.

# Disadvantages

1. Larger program size: OOP typically involves more lines of code than procedural programs.
2. Slower program: OOP typically slower than POP, as they require more instructions to be executed.
3. Not suitable for all types of programs.
4. To convert a real world problem into an object oriented model is difficult.
5. OOP software development, debugging and testing tools are not standardized.
6. Polymorphism and dynamic binding also requires processing time , due to overload of function calls during run time.

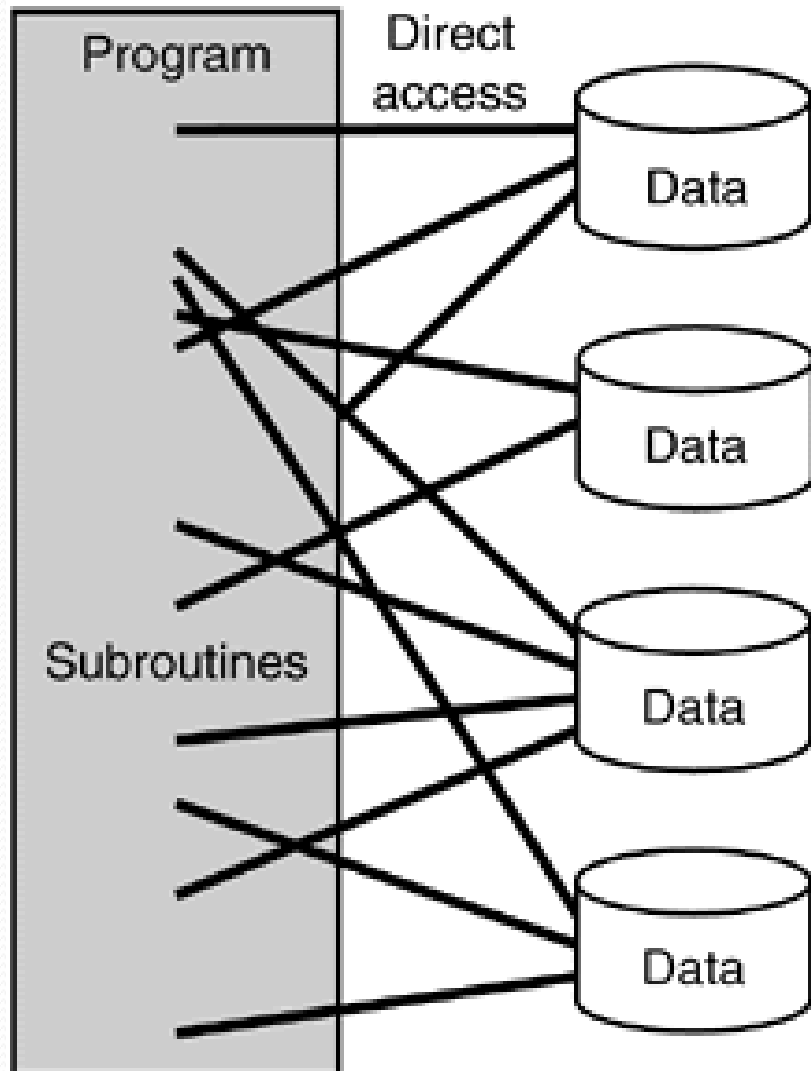


### 1.2.3 Procedure Oriented versus Object Oriented Programming

The differences between procedural and Object oriented programming are tabulated below:

<b>Procedure Oriented Programming</b>	<b>Object Oriented Programming</b>
Emphasis is given on procedures.	Emphasis is given on data.
Programs are divided into functions.	Programs are divided into objects.
Follow top-down approach of program design.	Follow bottom-up approach of program design.
Generally data cannot be hidden.	Data can be hidden, so that non-member function cannot access them.
It does not model the real world problem perfectly.	It models the real world problem very well.
Data move from function to function.	Data and function are tied together. Only related function can access them
Maintaining and enhancing code is still difficult.	Maintaining and enhancing code is easy.
Code reusability is still difficult.	Code reusability is easy in compare to procedure oriented approach.
Examples: FORTRAN, COBOL, Pascal, C	Example: C++, JAVA, Smalltalk

## Procedural technology



## Object-oriented technology

