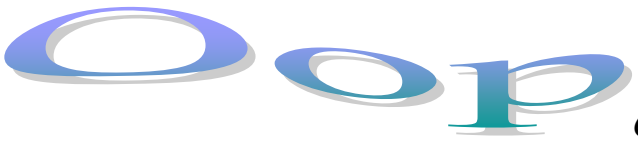


cs304

*Object oriented programming*

FAQs

What is Abstraction?

Answer: The importance of abstraction is derived from its ability to hide irrelevant details and from the use of names to reference objects. Abstraction is essential in the construction of programs. It places the emphasis on what an object is or does rather than how it is represented or how it works. Thus, it is the primary means of managing complexity in large programs.

---

Question: What is a Class Diagram?

Answer: A class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects.

---

Question: What is Method Overriding?

SONU MUGHAL

Answer: Method overriding is a language feature that allows a subclass to override a specific implementation of a method that is already provided by one of its super-classes. A subclass can give its own definition of methods but need to have the same signature as the method in its super-class. This means that when overriding a method, the subclass's method has to have the same name and parameter list as the super-class's overridden method.

---

Question: What is Operator Overloading?

Answer: The operator overloading is a specific case of polymorphisms in which some or all of operators like +, - or == are treated as polymorphic (multi) functions and as such have different behaviors depending on the types of its arguments.

---

Question: What is Method Overloading?

Answer: The method overloading is the ability to define several methods (in same class) all with the same name but different on the basis of i) number of parameters ii) types of parameters.

---

Question: What are Polymorphisms?

SONU MUGHAL

Answer: Polymorphism is a generic term that means 'many shapes'. More precisely Polymorphism means the ability to request that the same operations be performed by a wide range of different types of things.

---

Question: What is Inheritance?

Answer: Ability of a new class to be created, from an existing class by extending it, is called inheritance.

---

Question: What is a base class?

Answer: When inheritance is used to create a new class from another, the new class is called the subclass or derived class, and the class from which it was derived is called the base class.

---

Question: What is a concrete class?

Answer: A concrete class is one that can be used to directly create, or instantiate objects, unlike an abstract base class which can only be used as a base class for other classes which eventually lead to concrete classes

---

SONU MUGHAL

Question: What are data members?

Answer: Objects are miniature programs, consisting of both code and data. The code consists of a series of member functions. The data items are called data members.

---

What is a constructor?

Answer: Objects are complete, miniature programs and, like any good programs, have well defined initialization and termination phases. They have special routines (i.e. member functions) to look after this. The initialization routine is called the constructor, and C++ ensures that every object is properly initialized by calling its constructor. The designer of the object can have more than one constructor, a situation called overloading and then the compiler will select between them depending on exactly what arguments are passed to the constructor function. However, there must always be a default constructor, to be used when no information is supplied.

Question: What is a destructor?

Answer: The termination routine is called the destructor, and C++ will provide a default if none is supplied. If, during the

SONU MUGHAL

lifetime of the object, it uses heap memory then the designer of the object must provide a destructor function to release such memory to avoid a memory leak.

Question: What is global variable?

Answer: Global variables can be accessed throughout a program. Another way to put this is to say they have global scope.

Question: What is local variable?

Answer: Local variables can only be accessed within the function, or more specifically the compound statement in which they are declared. Another way to put this is to say they have local scope.

Question: What is a null pointer?

Answer: A null pointer is a pointer that is currently pointing to nothing. Often pointers are set to zero to make them null pointers or tested against zero to see if they are null or not.

Question: What is a pointer?

Answer: A pointer is a variable that holds the address of another variable or object.

Question: What is meant by protected?

SONU MUGHAL

Answer: The protected keyword in the class statement means that the following members of the class are not available to users of the objects of the class, but can be used by any subclass that inherits from it, and consequently forms part of its implementation.

Question: What is OOP?

Answer: The object-oriented programming is commonly known as OOP. Most of the languages are developed using OOP concept. Object-oriented programming (OOP) is a programming concept that uses "objects" to develop a system. An object hides the implementation details and exposes only the functionalities and parameters it requires to its client. Here also an object shares the same concept as that of a bike. While driving a motor bike, we are unaware of its implementation details such as how it is developed, internal working of gears etc.? We know only the functions or actions it can perform.

Question: What are the various elements of OOP?

Answer: Various elements of OOP are: Object Class Method Encapsulation Information Hiding Inheritance Polymorphism

Question: What are the characteristics of Object-Oriented programming language?

SONU MUGHAL

Answer: Some key features of the Object Oriented programming are: Emphasis on data rather than procedure Programs are divided into entities known as objects Data Structures are designed such that they characterize objects Functions that operate on data of an object are tied together in data structures Data is hidden and cannot be accessed by external functions Objects communicate with each other through functions New data and functions can be easily added whenever necessary Follows bottom up design in program des.

---

What are the basic Concepts used in the Object-Oriented Programming language?

Answer: Object Class Data Abstraction and Encapsulation Polymorphism Inheritance Message passing Dynamic binding

Question: What Is an Object? (Object-Oriented Technology)

SONU MUGHAL

Answer: There are many definitions of an object, such as found in [Booch 91, p77]: "An object has state, behavior, and identity; the structure and behavior of similar objects are defined in their common class; the terms instance and object are interchangeable". This is a "classical languages" definition, as defined in [Coplien 92, p280], where "classes play a central role in the object model", since they do not in prototyping/delegation languages. "The term object was first formally applied in the Simula language, and objects typically existed in Simula programs to simulate some aspect of reality" [Booch 91, p77]. Other definitions referenced by Booch include Smith and Tockey: "an object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain." and [Cox 91]: "anything with a crisply defined boundary" (in context, this is "outside the computer domain". A more conventional definition appears on pg 54). Booch goes on to describe these definitions in depth. [Martin 92, p 241] defines: "An "object" is anything to which a concept applies", and "A concept is an idea or notion we share that applies to certain objects in our awareness". [Rumbaugh 91] defines: "We define an object as a concept, abstraction or thing with crisp boundaries and meaning for the problem at hand." [Shlaer 88, p 14] defines: "An object is an abstraction of a set of real-world things such that:

Question: What Is Object Encapsulation (Or Protection)?

SONU MUGHAL

Answer: [Booch 91, p. 45] defines: "Encapsulation is the process of hiding all of the details of an object that do not contribute to its essential characteristics." [Coad 91, 1.1.2] defines: "Encapsulation (Information Hiding). A principle, used when developing an overall program structure, that each component of a program should encapsulate or hide a single design decision... The interface to each module is defined in such a way as to reveal as little as possible about its inner workings. [Oxford, 1986]" Some languages permit arbitrary access to objects and allow methods to be defined outside of a class as in conventional programming. Simula and Object Pascal provide no protection for objects, meaning instance variables may be accessed wherever visible. CLOS and Ada allow methods to be defined outside of a class, providing functions and procedures. While both CLOS and Ada have packages for encapsulation, CLOS's are optional while Ada's methodology clearly specifies class-like encapsulation (Adts). However most object-oriented languages provide a well-defined interface to their objects thru classes. C++ has a very general encapsulation/protection mechanism with public, private and protected members. Public members (member data and member functions) may be accessed from anywhere. A Stack's Push and Pop methods will be public. Private members are only accessible from within a class. A Stack's representation, such as a list or array, will usually be private.

SONU MUGHAL

Protected members are accessible from within a class and also from within subclasses (also called derived classes). A Stack's representation could be declared protected allowing subclass access. C++ also allows a class to specify friends (other (sub)classes and functions), that can access all members (its representation). Eiffel 3.0 allows exporting access to specific classes.

Question: What Is A Class?

Answer: A class is a general term denoting classification and also has a new meaning in object-oriented methods. Within the OO context, a class is a specification of structure (instance variables), behavior (methods), and inheritance (parents, or recursive structure and behavior) for objects. As pointed out above, classes can also specify access permissions for clients and derived classes, visibility and member lookup resolution. This is a feature-based or intentional definition, emphasizing a class as a descriptor/constructor of objects (as opposed to a collection of objects, as with the more classical extensional view, which may begin the analysis process). Original Aristotle a classification defines a "class" as a generalization of objects: [Booch 91, p93] "a group, set, or kind marked by common attributes or a common attribute; a group division, distinction, or rating based on quality, degree of competence, or condition".

SONU MUGHAL

Question: What Is A Meta-Class?

Answer: Meta-Class is a class' class. If a class is an object, then that object must have a class (in classical OO anyway). Compilers provide an easy way to picture Meta-Classes. Classes must be implemented in some way; perhaps with dictionaries for methods, instances, and parents and methods to perform all the work of being a class. This can be declared in a class named "Meta-Class". The Meta-Class can also provide services to application programs, such as returning a set of all methods, instances or parents for review (or even modification). [Booch 91, p 119] provides another example in Smalltalk with timers. In Smalltalk, the situation is more complex

Question: What Is Inheritance?

Answer: Inheritance provides a natural classification for kinds of objects and allows for the commonality of objects to be explicitly taken advantage of in modeling and constructing object systems. Natural means we use concepts, classification, and generalization to understand and deal with the complexities of the real world. See the example below using computers. Inheritance is a relationship between classes where one class is the parent base/superclass/ancestor/etc.) class of another. Inheritance provides programming by extension (as opposed to programming by reinvention [LaLonde

SONU MUGHAL

90]) and can be used as an is-a-kind-of (or is-a) relationship or for differential programming. Inheritance can also double for assignment

Question: What Is the Difference Between Object-Based and Object-Oriented?

Answer: Object-Based Programming usually refers to objects without inheritance [Cardelli 85] and hence without polymorphism, as in '83 Ada and Modula-2. These languages support abstract data types (Adts) and not classes, which provide inheritance and polymorphism. Ada95 and Modula-3; however, support both inheritance and polymorphism and are object-oriented. [Cardelli 85, p481] state "that a language is object-oriented if and only if it satisfies the following requirements: - It supports objects that are data abstractions with an interface of named operations and a hidden local state. - Objects have an associated type. - Types may inherit attributes from supertypes. object-oriented = data abstractions + object types + type inheritance These definitions are also found in [Booch 91, Ch2 and Wegner 87]. [Coad 91] provides another model: Object-Oriented = Classes and Objects + Inheritance + Communication with messages.

SONU MUGHAL

## Lecture 01 complete concept:

To understand OOP, you must first understand what programming was like before OOP.

Back then, the basic definition of programming was this : a program is a sequence of logical instructions followed by the computer. And that's it. All well and good, but let's face it, it's hardly inspiring. Until now, that is. It's been hiding in the background for quite some time now, but OOP has finally taken off. In an OO programming language, the emphasis is placed far more on the data, or the 'objects' used and how the programmer manipulates them. Before OOP, numbers were simply an address in memory; a sequence of bytes that meant nothing. Now, however, through OOP they have become far more than that. The program is now a solution to whatever problem it is you have, but now it is done in the terms of the objects that define that problem, and using functions that work with those objects

### A Historical Interlude

Hundreds of years ago, in Britain (specifically England), there was civil unrest. People were angry - the poor people to be more specific. They noticed that some people were richer than them, they did not like it. What to do about this problem? How to keep the people happy? Religion had already gone some of the way, but even the promise of eternal utopia

if the poor behaved themselves in life didn't seem to work. Capitalism already had sunk its powerful jaws into the world, and a new idea was needed to keep the masses happy. That idea became known as 'class'. The basis was that if everyone understood their place and role in society, they would feel secure and happy, and would not challenge the authority. It worked. There was the upper class (who were rich), the middle class (who were not so rich), and the poor sods class (who could barely afford to live). Quite unfair, but nevertheless it became reality. What has this got to do with C++ you ask? Well in C++, all Object Orientation comes in the form of classes. But enough of that; we're programmers, not social scientists.

**Data types**

Up to this point in your use of C++, you've used only the basic types of variables : int, float, bool, double, and so forth. These are called simple data types. However, they are very linear in what we can 'model' with them. Let's take an example. Let's say we wanted to represent a real life object, say a house. Obviously, we would have to examine the various attributes of a house : the number of rooms it has, its street number and whether or not it has a garden (okay, so there are more attributes, but I won't go into them now). In C++, we could show the house like this:

```
int number, rooms;
bool garden;
```

SONU MUGHAL

And it would work fine for this particular example. But suppose we wanted many houses? Suppose we wanted to make the program more complicated than this? Suppose we wanted to define our *own* data type to represent the house. C++ allows us to do this through the use of classes.

**History of oop?**

The basis for **OOP** started in the early 1960s. A breakthrough involving instances and objects was achieved at MIT with the PDP-1, and the first programming language to use objects was Simula 67. It was designed for the purpose of creating simulations, and was developed by Kristen Nygaard and Ole-Johan Dahl in Norway.

**Who invented OOP?**

Alan Kay

**Alan Kay** invented OOP and coined the term.

**What was the first OOP language?**

SIMULA 67 was formally standardized on the **first** meeting of the SIMULA Standards Group (SSG) in February 1968. Simula was influential in the development of Smalltalk and later **object-oriented** programming **languages**.

**When OOP concept did first came into picture?**

**When OOP concept did first came into picture?**

Explanation: **OOP first came into picture in** 1970's by Alan

and his team. Later it was used by some programming languages and got implemented successfully, SmallTalk was **first**language **to** use pure **OOP** and followed all rules strictl

**Why was OOP developed?**

It was **created** for making simulation programs, in which what came to be called objects were the most important information representation. Smalltalk (1972 to 1980) is another early example, and the one with which much of the theory of **OOP was developed**.

Four principles of OOP are encapsulation,data abstraction, data hiding and inheritance.

Encapsulation: In technical terms, it means wrapping up of data and code in to a single unit(i.e Class) and also protecting the data from outside world.

There is also another term related with encapsulation is data hiding. Data hiding means hiding the data from world. Data can not be accessed directly.

Data Abstraction: Abstraction means hiding unessential details from user. And providing only essential information.

Inheritance: Inheriting the properties of super class in subclass. Basically subclass is more specialized one and it provides re usability.

In layman terms: For eg Mobile phone which we use everyday. Everything is wrapped inside the body of the phone. No one

can access the functionality of mobile directly. To access the stuff you need object(here object is mobile)

you can't make a call without having mobile. Now what is data hiding?

Data hiding means giving access to mobile in control manner.

In technical terms: we use getter and setter methods. But in our mobile: Password and pattern lock are the ways of providing data hiding. One who knows the password, only can access your phone(This is called data hiding)

Data hiding basically provides you security. Your phone is locked. Even if you are not having your phone nearby . you know that it is protected.

Data Abstraction: suppose you wanna make a call to your mom. What you need?

Just valid number!

To make a call, you never need to know the background details how calling is being done. (Like connecting to network whichever your mom is using either airtel or idea etc)

So What data abstraction gives? it provides easiness to us.

Inheritance: Suppose you are using android phone version suppose lollypop. So basically lollypop version has inherited all functionality of previous versions i.e kitkat

with some new functionalities.

SONU MUGHAL

So, Lollypop version basically is specialized version of Kitkat.

There is another principle also: Polymorphism

It means having same name but different functionality. In technical terms, function overloading, function overriding etc

Now let see how polymorphism is implemented in your mobile.

Every mobile has inbuilt camera. But still there are another apps which let us take photos like Retrica.

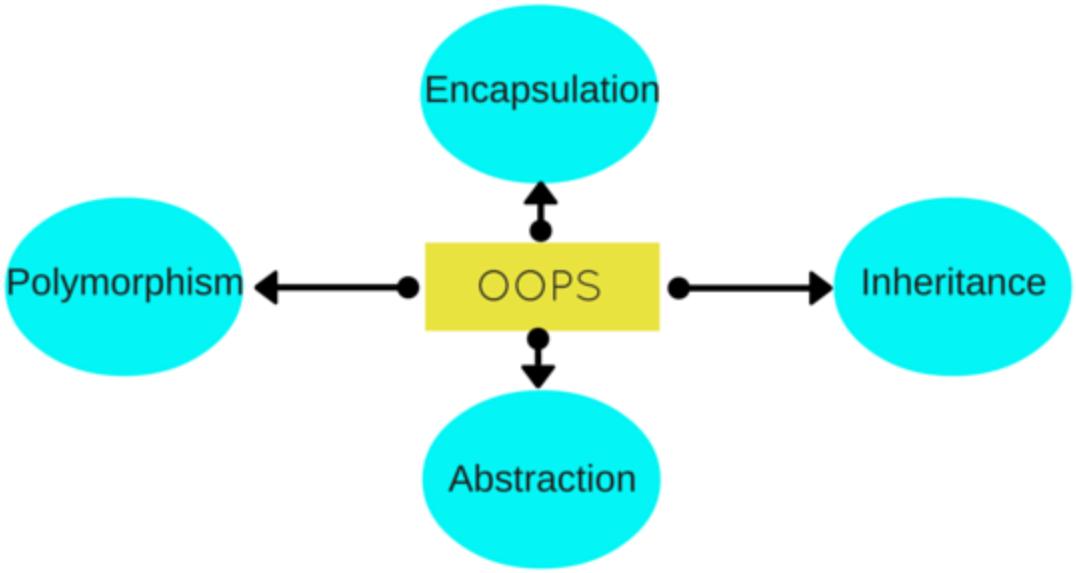
Retrica which has different functionality but does the same work(clicking photos).

**So, what is OOPS (Object Oriented Programming)???**

Object-oriented Programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.

**Here are the four principles of OOPS are : pillers of oop**

SONU MUGHAL




- **Abstraction:** Abstraction means using simple things to represents complexity. Abstraction can be understood by an example of the TV, where we have a simple button to switch it on and without being bothered by its internal working power and circuity we can start with a button.
- **Inheritance:** When an object acquires all the properties and behavior of a parent object. A car is a four-wheeler vehicle so assume that we have a class Four Wheeler and a subclass of it named Car.
- **Polymorphism:** Polymorphism means one name and many forms and it works on parent and child relationship. A task is performed in different ways. For example: to convince the customer differently.
- **Encapsulation:** Encapsulation is the technique used to implement abstraction in object-oriented programming. Encapsulation is used for access restriction to class

SONU MUGI

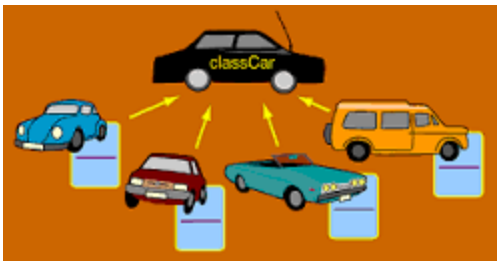
members and methods. An example of encapsulation is the class of java.util.Hashtable.

https://www.youtube.com/channel/UCbIedEXkM13ynuI1sNAI1cA

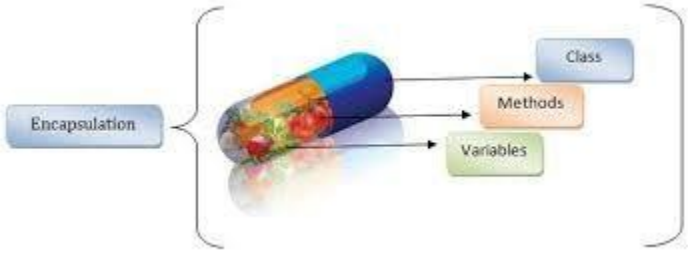
- **Abstraction:** Abstraction refers to showing only the essential features of the application and hiding the details. In C++/Java, classes provide methods to the outside world to access & use the data variables, but the variables are hidden from direct access. This can be done access specifiers. For example: phone call, we don't know the internal processing.

**Abstraction = Encapsulation + Data Hiding**


**Inheritance:** Inheritance is a way to reuse code. The class which is inherited from, is called the base class, and the class which inherits the code from the base class is called a derived class. A derived class can use all the functions which are defined in the base class, making the code reusable.




- **Encapsulation:** It can also be said data binding. Encapsulation is all about binding the data variables and functions together in class.




- **Polymorphism:** It is a feature, which lets us create functions with same name but different arguments, which will perform differently. That is function with same name, functioning in different way. Or, it also allows us to redefine a function to provide its new definition.


SONU MUGHAL

Hope this short yet crispy explanation help you to clear your OOPS concepts…

"some of from"

The four Pillars of OOP are in Grady Booch's original book on the topic,. Object Oriented Analysis and Design. They are:

Abstraction - Ability to represent complex software systems as a domain model of classes and objects, abstracting out the details of real world object models through the attributes (e.g. color, size, shape) and methods (actions) of these classes (e.g. read(), write(), playSound())

Encapsulation - Providing different levels of scope on how classes, methods, and attributes can be accessed and only allowing access on a need-to-know basis. Generally, oop uses private, public and protected keywords to implement encapsulation on attributes, methods and classes. For example, a private method in a class could only be accessed by the class and a public method could be accessed any other class.

Inheritance -. The ability for classes to be generalized in the base class and specialized in sublasses of the base class. A subclass can inherit the attributes and methods of a base class and a subclass can specialize with it's own additional attributes and methods.

Polymorphism - The ability for multiple objects of the same base class, but different subclasses to override a method of

SONU MUGHAL

the base class and perform different operations on that same method. For example, say I had a base class called animal and subclasses: cheetah, deer, and sloth. The base class had a method called CalculateTopSpeed. I could override that method in all 3 subclasses and each subclass would return a different answer. I could have a collection of animal objects. The animal collection could consist of 3 objects: a cheetah, a sloth and a deer object. If I loop through this collection and call CalculateTopSpeed on each item in the collection, I would get polymorphic behavior on each object. Even though all the objects in the collection are all animals, they are different kind of animals in their subclass and give different answers for the overridden method CalculateTopSpeed.

**Advantages of OOP:**

- It provides a clear ***modular structure*** for programs which makes it good for defining abstract datatypes in which implementation details are hidden
- Objects can also be ***reused*** within an across applications. The reuse of software also lowers the cost of development. More effort is put into the object-oriented analysis and design, which lowers the overall cost of development.
- It makes software ***easier to maintain.*** Since the design is modular, part of the system can be updated in case of issues without a need to make large-scale changes

SONU MUGHAL

- Reuse also enables ***faster development***. Object-oriented programming languages come with rich libraries of objects, and code developed during projects is also reusable in future projects.
  - It provides a good framework for code libraries where the supplied software components can be ***easily adapted and modified by the programmer***. This is particularly useful for developing graphical user interfaces.
  - ***Better Productivity*** **as** OOP techniques enforce rules on a programmer that, in the long run, help her get more work done; finished programs work better, have more features and are easier to read and maintain. OOP programmers take new and existing software objects and "stitch" them together to make new programs. Because object libraries contain many useful functions, software developers don't have to reinvent the wheel as often; more of their time goes into making the new program

SONU MUGHAL

What is coop ?

**Object-oriented programming** (*OOP*) is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as *attributes*), and code, in the form of procedures (often known as *methods*). A feature of objects is an object's procedures that can access and often modify the data fields of the object with which they are associated (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.[1][2] OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

What is object?

an **object** can be a variable, a data structure, a function, or a method, and as such, is a value in memory referenced by an identifier.

In the class-based object-oriented programming paradigm, *object* refers to a particular instance of a class, where the object can be a combination of variables, functions, and data structures.

What is class?

To make object in programming we make classes

Data + function = classes

## What is Object-Orientation?

It is a technique in which we visualize our programming problems in the form of
objects and their interactions as happen in real life.

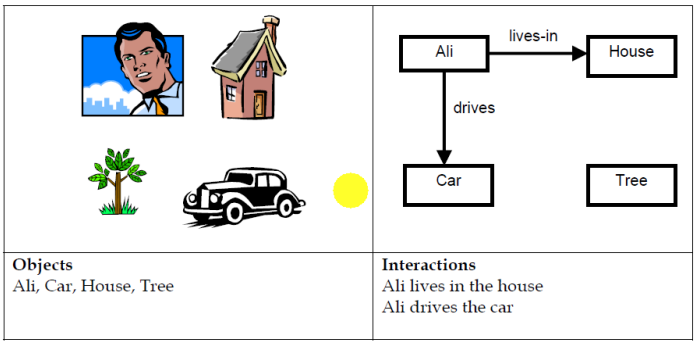
## What is a Model?

A model is an abstraction of something real or conceptual.
We need models to understand an aspect of reality.

## Objects

Ali, Car, House, Tree

Example 1- Object Oriented Model




| Objects | Interactions |
|---|---|
| Ali, Car, House, Tree | Ali lives in the house<br>Ali drives the car |

## Interactions

Ali lives in the house
Ali drives the car

MUGHAL

**Object-Orientation - Advantages**

As Object Oriented Models map directly to reality as we have seen in examples
above therefore,
We can easily **develop** an object-oriented model for a problem.
Everyone can easily **understand** an object-oriented model.
We can easily implement an object-oriented model for a problem using any object
oriented language like c++ using its features1 like classes, inheritance, virtual
functions and so on…

**What is an Object?**

*An object is,*
1. Something tangible (Ali, School, House, Car).
2. Something conceptual (that can be apprehended intellectually for example
time, date and so on…).

**Summary:**

• Model is the abstraction of some real word scenario. It helps us to understand
that scenario.
• Object oriented model of any scenario (problem) describes that scenario
(problem) in the form of interacting objects.

SONU MUGHAL

• We use Object Orientation because it helps us in mapping real world problem
in a programming language.
• Object Orientation is achieved using objects and their relationships.
• Properties of an object are described using its *data members* and behavior of an
object is described using its *functions.*
• Objects may be tangible (physical) or intangible (also called conceptual or
virtual).
• Generally, when we have given a certain problem description, **nouns** in that
problem description are *candidates* for becoming objects of our system.
• There may be more than one aspect of an object
• It is not necessary that every object has a specific role in implementation of a
problem there may be some objects without any role, like school parking in
our school.
• It is easier to develop programs using Object Oriented Programming because
it is closer to real life.

```
#include<iostream>


using namespace std;
```

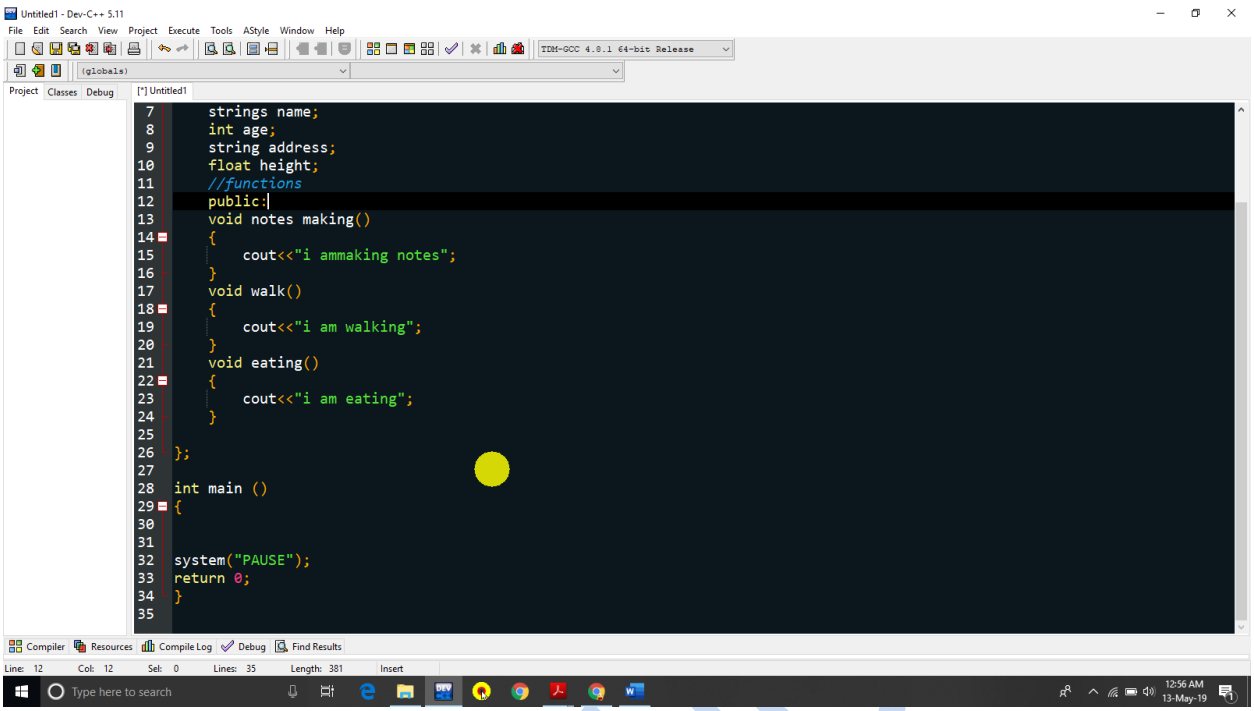
SONU MUGHAL

```
class sonu
{
	//data members
	strings name;
	int age;
	string address;
	float height;
	//functions
	public:
	void notes making()
	{
		cout<<"i ammaking notes";
	}
	void walk()
	{
		cout<<"i am walking";
	}
	void eating()
```

SONU MUGHAL

```
        {
                cout<<"i am eating";
        }


};


int main ()
{



system("PAUSE");
return 0;
}
```

SONU MUGHAL




Use of classes and object in coding

**Classy!**

Continuing with our example of the house, let's have a look at how we could 'model' a house using a C++ class:

```
class house
{
public:
    int number, rooms;
        bool garden;
};

main()
{
        house my_house;
```

SONU MOU

```
        my_house.number=40;
        my_house.rooms=8;
        my_house.garden=1;
        return 0;
}
```

A class defines a data type

A region of storage with associated semantics.

After the declaration int i; we say that "i is an object of type int." In OO/C++, "object" usually means "an instance of a class." Thus a class defines the behavior of possibly many objects (instances).

**Lecture No.02**

**Information Hiding:**

Information hiding is one of the most important principles of OOP inspired from real
life which says that all information should not be accessible to all persons. Private
information should only be accessible to its owner.

By Information Hiding we mean "*Showing only those details to the outside world which*

*are necessary for the outside world and hiding all other details from the outside world."*

**Real Life Examples of Information Hiding**

1. Ali's name and other personal information is stored in his brain we can't access this information directly. For getting this information we need to ask Ali about it and it will be up to Ali how much details he would like to share with us.
2. An email server may have account information of millions of people but it will share only our account information with us if we request it to send anyone else accounts information our request will be refused

In object oriented programming approach we have objects with their attributes and behaviors that are hidden from other classes, so we can say that object oriented programming *follows the principle of information hiding.*

*"Hiding the object details (state and behavior) from the users"*

Information Hiding is achieved in Object Oriented Programming using the following principles,

SONU MUGHAL

•All information related to an object is stored within the object

•It is hidden from the outside world

•It can only be manipulated by the object itself

**Advantages of Information Hiding**

Following are two major advantages of information hiding,

***It simplifies our Object Oriented Model:***

As we saw earlier that our object oriented model only had objects and their interactions hiding implementation details so it makes it easier for everyone to understand our object oriented model.

***It is a barrier against change propagation***

As implementation of functions is limited to our class and we have only given the name of functions to user along with description of parameters so if we change implementation of function it doesn't affect the object oriented model.

We can achieve information hiding using **Encapsulation** and **Abstraction**, so we see these two concepts in detail now,

**Encapsulation**

Encapsulation means *"we have enclosed all the characteristics of an object in the object itself"*

SONU MUGHAL

Encapsulation and information hiding are much related concepts (information
hiding is achieved using Encapsulation)

Consider the same example of object Ali of previous lecture we described it as
follows,

object characteristics include data members and behavior of the object in the form of functions.

we can say that Data and Behavior are tightly coupled inside an object and both the information structure and implementation details of its operations are hidden from the outer world.

You can see that Ali stores his personal information in itself and its behavior is
also implemented in it.
Now it is up to object Ali whether he wants to share that information with
outside world or not. Same thing stands for its behavior if some other object in
real life wants to use his behavior of walking it can not use it without the
permission of Ali.

So we say that attributes and behavior of Ali are encapsulated in it.

SONU MUGHAL

**Advantages of Encapsulation**

The following are the main advantages of Encapsulation,

**a. Simplicity and clarity**

As all data and functions are stored in the objects so there is no data or function
around in program that is not part of any object and is this way it becomes very
easy to understand the purpose of each data member and function in an object.

**b. Low complexity**

As data members and functions are hidden in objects and each object has a
specific behavior so there is less complexity in code there will be no such
situations that a functions is using some other function and that functions is
using some other function.

**c. Better understanding**

Everyone will be able to understand whole scenario by simple looking into object
diagrams without any issue as each object has specific role and specific relation
with other objects.

**Interface :**

SONU MUGHAL

Interface is a set of functions of an object that he wants to expose to other objects

• Different objects may need different functions of an object so interface of
an object may be different for different objects.
• Interfaces are necessary for object communication. Each object provides
interface/s (operations) to other objects through these interfaces other
objects communicate with this object.

**Example – Interface of a Phone**
• Input Number
• Place Call
• Disconnect Call
• Add number to address book
• Remove number
• Update number

**Implementation**

It is actual implementation of the behavior of the object in any Object Oriented
language.
It has two parts,
• Internal data structures to hold an object state that will be hidden from us
it will store values for an object data members.

SONU MUGHAL

• Functionality in the form of member functions to provide required
behavior.

• **Data Structure** in the form of Mechanical structure of gear box
• **Functionality** mechanism to change gear

**b. Address Book in a Phone**

Similarly take the example of contact details saved in the SIM of a phone,
In that case we can say physical structure of SIM card as **Data Structure**
And Read/write operations provided by the phone as **Functionality**.

**Real Life example of separation of interface and implementations**

 Driver has a standard interface to drive a car and using that interface
he drive can drive any car regardless of its model or type whatever
engine type it has or whatever type of fuel it is using.

Question

What is the difference between an Object Model Diagram and a Class Diagram in IBM Rational Rhapsody?

SONU MUGHAL

Cause

You may want to understand which diagram is more suitable as per your requirement.

Answer

**Class diagram** is a graph of classifier elements connected by their various static relationships. A “class” diagram may also contain interfaces, packages, relationships, and even instances,
such as objects and links. Perhaps a better name would be “static structural diagram”, but “class diagram” is shorter and well established.

**Object diagram** on the other hand is a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram. It shows a snapshot of the detailed state of a system at a point in time. The use of object diagrams is fairly limited, mainly to show examples of data structures.

The actual differences lie in their purpose. A Class diagram shows your classes and their relationships. An Object Model Diagram shows the interaction between objects at some point, during run time.

The actual differences lie in their purpose. A Class diagram shows your classes and their relationships. An Object Model Diagram shows the interaction between objects at some point, during run time.

A Class Diagram will show what the Objects in your system consist of (members) and what they are capable of doing (methods) mostly static. In contrast, an Object Diagram will show how objects in your system are interacting with each other at some point in time, and what values those objects contain when the program is in this state.