What is OOPs?

Object-Oriented Programming or OOPs refers to languages that use objects in programming. Oops aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions.

OOPS helps in increasing the reusability of code.

What is a class?

A class is a collection of objects. Classes don't consume any space in the memory.

It is a user defined data type that act as a template for creating objects of the identical type.

A large number of objects can be created using the same class. Therefore, Class is considered as the blueprint for the object.

What is an object?

objects are used to assign values to the variables.

An object is basic building block of OOPs.

An object is a real world entity which have properties and functionalities.

Object is also called an instance of class. Objects take some space in memory.

instance means - member

For eg .

Fruit is **class** and its **object** s are mango ,apple , banana

Furniture is **class** and its **objects** are table , chair , desk

What is the difference between a class and an object?

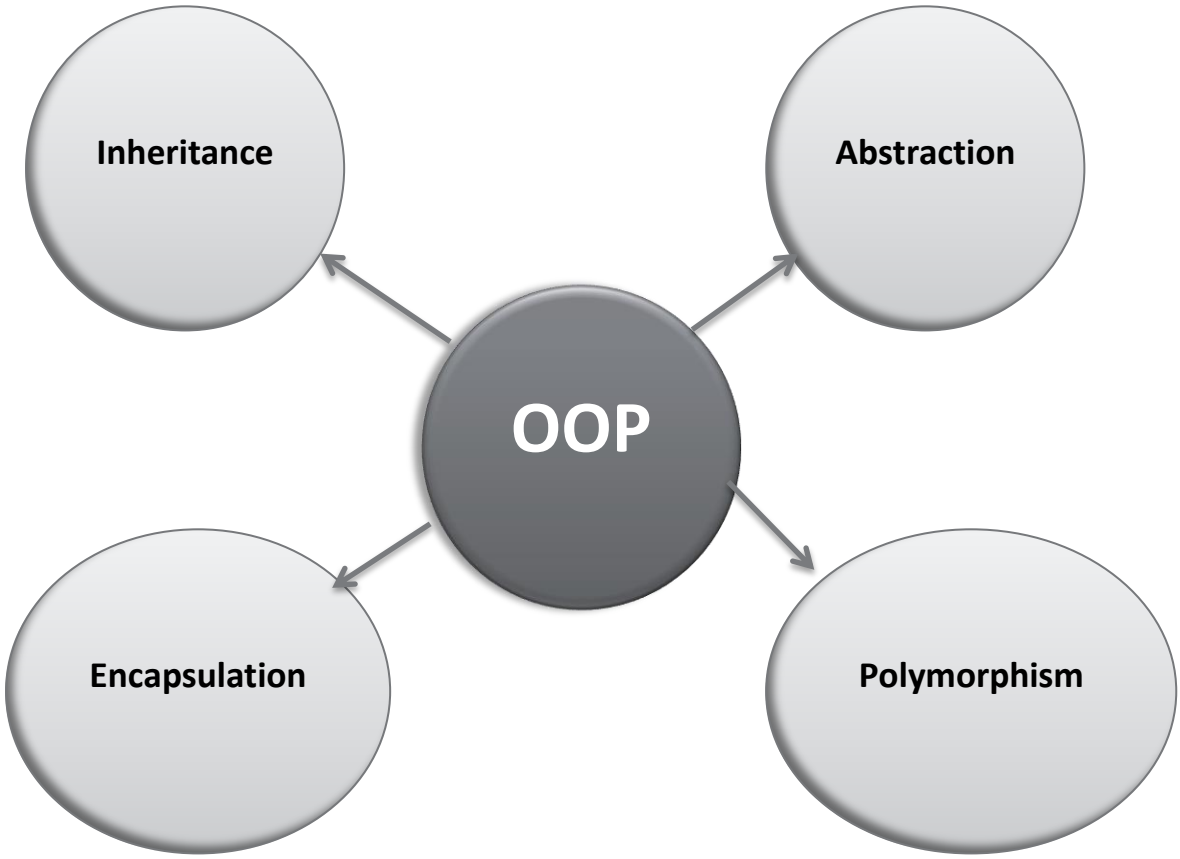
Class	Object
1. It is a collection of objects.	It is an instance of a class.
2. It doesn't take up space in memory.	It takes space in memory.
3. Class does not exist physically	Object exist physically.
4. Classes are declared just once	Objects can be declared as and when required

Mona Abdelmonem **monasallam400@gmail.com**

What is the difference between a class and a structure?

Class	Structure
1.Class is a collection of objects.	Structure is a collection of variables of different data types under a single unit
2. Class is used to combine data and methods together.	Structure is used to grouping data.
3. Class's objects are created on the heap memory.	Structure's objects are created on the stack memory.
4. A class can inherit another class.	A structure can't inherit another structure.
5. A class has all members private by default	A structure has all members public by default
6. Classes are ideal **for larger** or complex model objects	Structures are ideal **for small** and isolated model objects

Following are the basic features of OOPs -



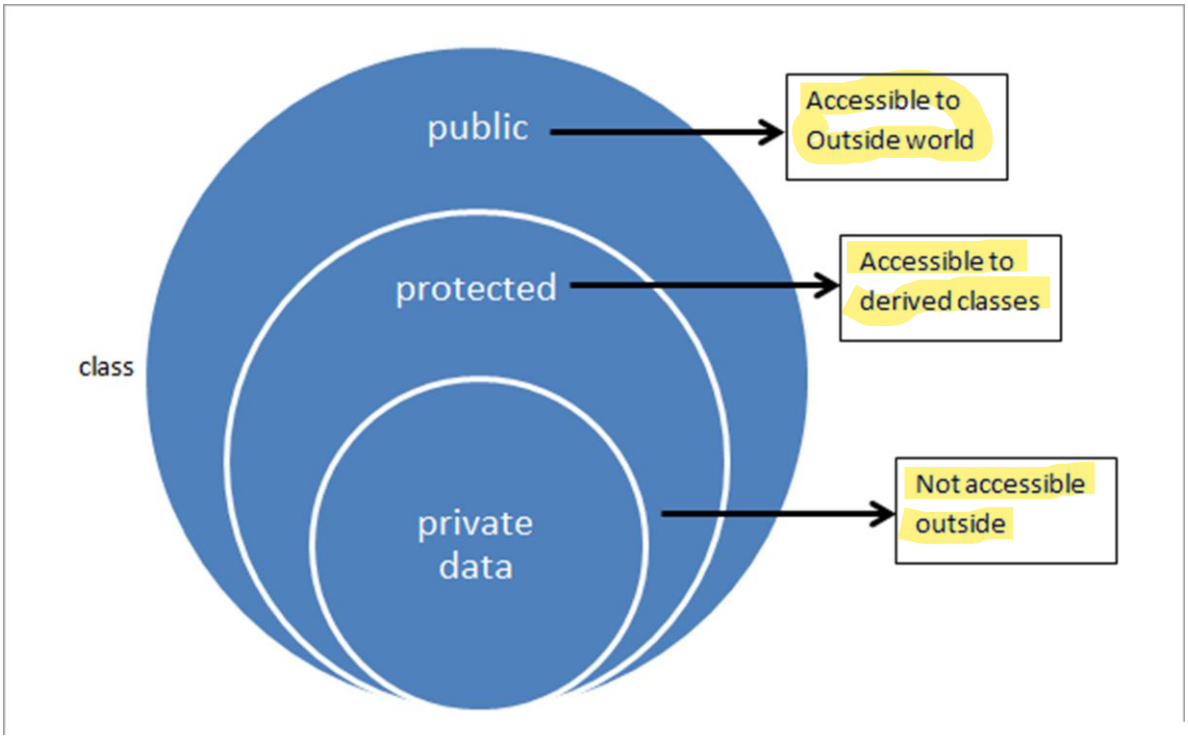
Mona Abdelmonem **monasallam400@gmail.com**

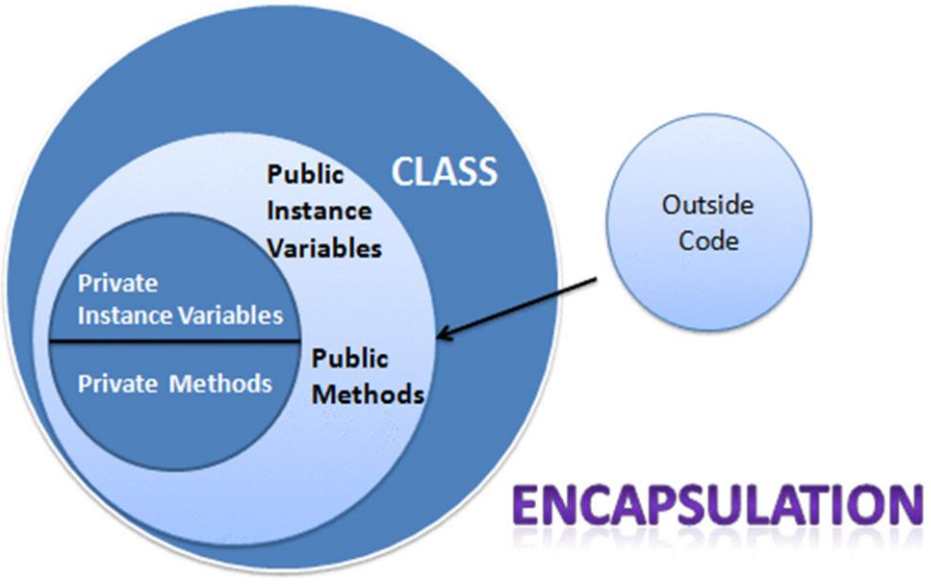
Encapsulation

The main advantage of encapsulation is that data is hidden and protected from access by outside non-member methods of a class.

Wrapping up of data members and member function a in single unit is called encapsulation.

In encapsulation, data(variables) are declared as private and methods are declared as public.





Mona Abdelmonem monasallam400@gmail.com

What are access specifiers ?

It allows us to restrict the visibility of a methods, variables, or other data members.

There are three types of most **common access specifiers,** which are following.

- Private
- Public
- Protected

Public Modifiers :

means that class, variable or method is accessible throughout from within and outside the class, within or outside the package, etc.

It provides **highest level** of accessibility.

Private Modifiers :

means that class, variable or method is not accessible from within or outside the class,

Private method can't be inherited to sub class.

This provides **lowest level** of accessibility.

Protected Modifiers :

Variables and data members are accessible only from derived Class.

Data cannot be accessed from outside the Class.

Data can be inherited within the class.

Access Modifiers	Accessible by classes in the same package	Accessible by classes in other packages	Accessible by subclasses in the same package	Accessible by subclasses in other packages
Private	NO	No	No	No
Public	Yes	Yes	Yes	Yes
Protected	Yes	NO	Yes	Yes

ALSO LEARN CODE OF OOPS OPERATIONS LIKE INHERITANCE etc

FROM SCHOOL COPY ALSO

Showing necessary details to the user and hiding unnecessary details from the user like internal coding is called data abstraction

If we talk about data abstraction in programming language, the code implementation is hidden from the user and only the necessary functionality is shown or provided to the user.

In other words , it deals with the outside view of an object (Interface).

Making data members and member function public is data abstraction.

Eg.

-All are performing operations on the ATM machine like cash withdrawal etc. but we can't know internal details about ATM

-phone call we don't know the internal processing

We can achieve data abstraction by using

1. Abstract class
2. Interface

What is an abstract class?

Abstract class is that class which contains abstract method.

Abstract methods are those methods which have only declaration not the implementation.

An abstract class is declared with abstract keyword.

An abstract class can also contain non-abstract methods.

Mona Abdelmonem **monasallam400@gmail.com**

Inheritance

Accessing the characteristics and features of Parent Class from child Class is called inheritance.

It inherits the properties and behaviour of other class

The Base Class, also known as the Parent Class is a class, from which other classes are derived.
or from which properties and features are accessed

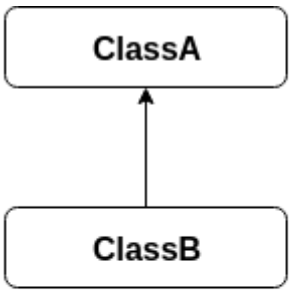
The Derived Class, also known as Child Class, is a class that is created from an existing class
They access properties and features from another class.

There are five types of inheritance in OOP:

- Single Level Inheritance
- Hierarchical Inheritance
- Multi-Level Inheritance
- Multiple Inheritance
- Hybrid inheritance

Single Level Inheritance

When a class inherits properties and behaviour of only one class.
In other words, in single inheritance there is only one base class and only one child class.



Single Inheritance

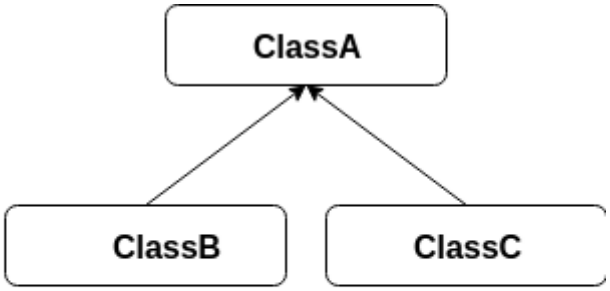
Hierarchical Inheritance

Hierarchical is like Parents- We have only one parent and may have many children.

one

When more than class inherit properties and behaviour of only one class

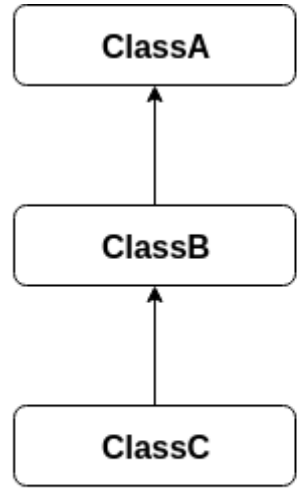
In Hierarchical Inheritance there are only one parent and many child class



Hierarchical Inheritance

Multi-Level Inheritance

In this type of inheritance, a derived class inherits the features and properties from another derived class.



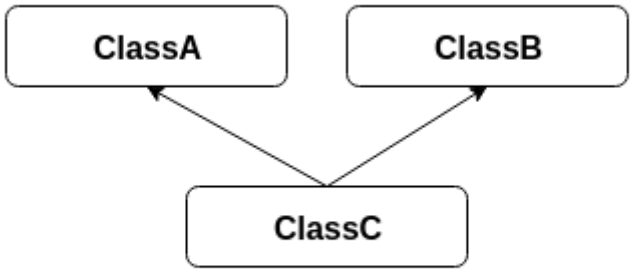
Multilevel Inheritance

Multiple Inheritance

In this ,there are many parents and only one child.

When a class inherits the properties and the behaviour of more than one class

Java, C#, most of high level language don't support Multiple Inheritance



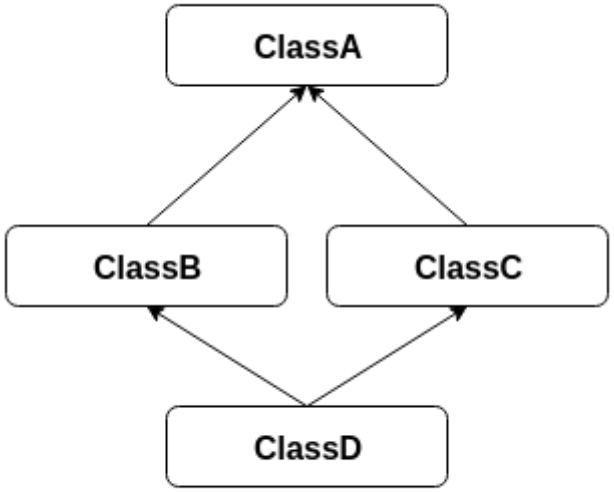
Multiple Inheritance

Hybrid Inheritance

type

Hybrid inheritance is a of inheritance is a combination of more than

one type of inheritance.



Hybrid Inheritance

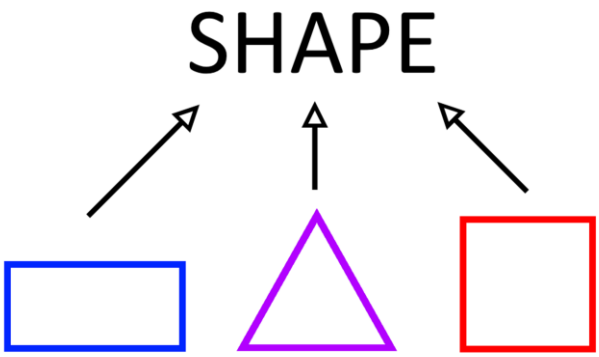
Why Java or C# don't support multiple inheritance?

because of Diamond Problem, Multiple inheritance complicate the design and make confusion for compiler

Polymorphism

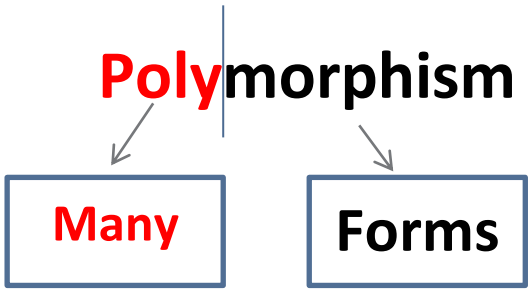
Poly means many
morphism means forms

Polymorphism is the ability of an object to take on **many forms.**
we can define polymorphism as the ability of a message to be displayed in more than one form. in programming



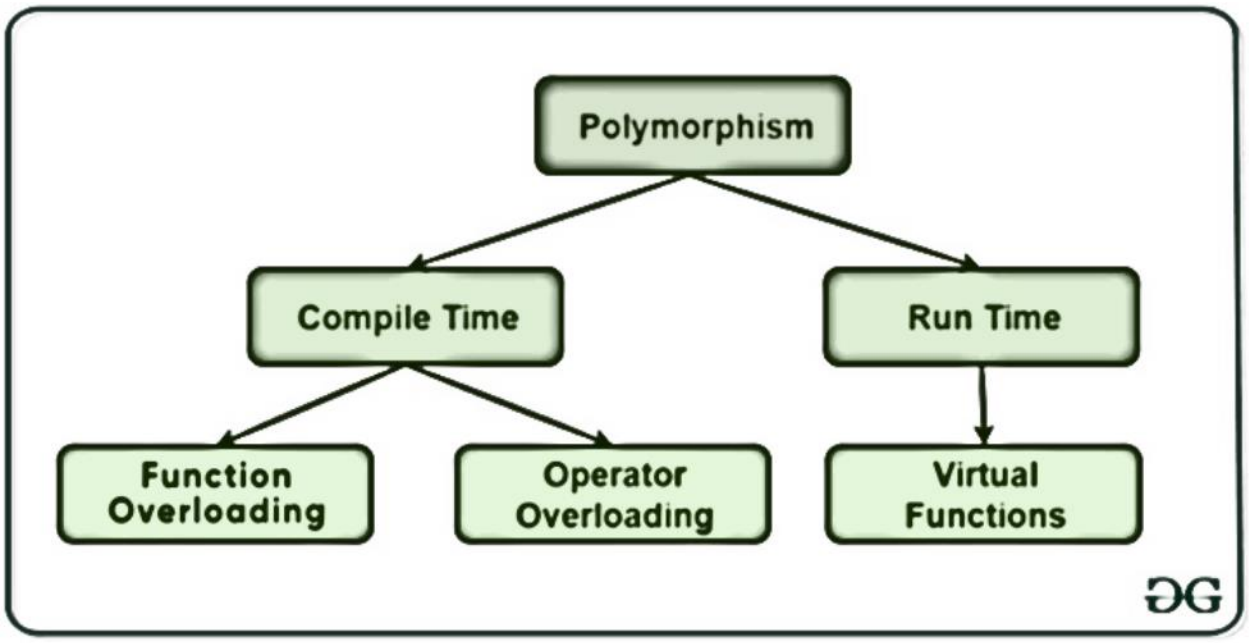
A real-life example of polymorphism, a man at the same time is a father, a husband, an employee.

Another good real time example of polymorphism is water. Water is a liquid at normal temperature, but it can be changed to solid when it frozen, or same water changes to a gas when it is heated at its boiling point .Thus, same water exhibiting different roles is polymorphism.



polymorphism is mainly divided into two types:

monasallam400@gmail.com



Types of Polymorphism:

1. Compile time polymorphism: Function Overloading

This type of polymorphism is achieved by function overloading or operator overloading.

Function overloading:

When there are multiple functions with same name but different parameters then these functions are said to be overloaded

multiple methods of same names performs different tasks within the same class.

2.Runtime polymorphism: Function Overriding

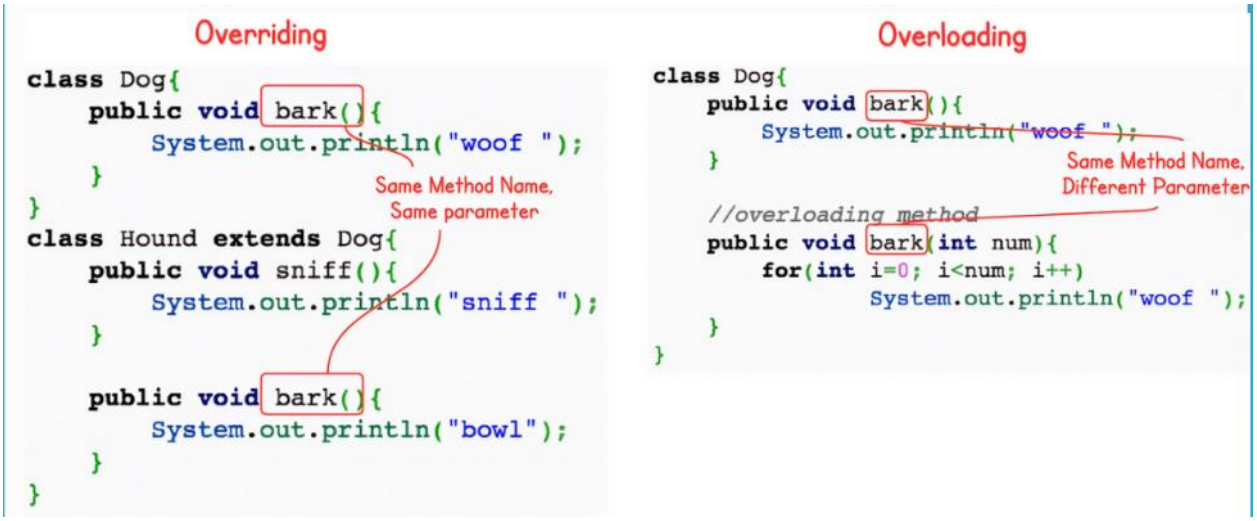
This type of polymorphism is achieved by Function Overriding.

Function Overriding:

Derived Class redefines the function in it , as it is already defined in base class is called function overriding. It is used to achieve runtime polymorphism.

% Function overloading occurs in same class.
% Function overriding occurs in atleast two classes.

Mona Abdelmonem **monasallam400@gmail.com**



Abstract Classes & Interfaces

Features	Abstract Class	Interface
Multiple inheritance	A class can inherit only one abstract class	A class can inherit multiple interfaces
Default Implementation	Provide signature ,partial and full implementation of its methods ,variables and other members	Provide only the signature of its methods ,variables ,properties and other member
Access Modifier	It allows to assign access modifier to its members	No access modifier can be assigned .All the members are treated as public
Nature VS Peripheral	It defines the core identity of the class and there is use for objects of same type	It identifies the peripheral identity of the class .it means human and vehicle can inherit from IMovable interface
Homogeneity	If various implementation of same nature which requires shared code that represent same status or behaviour then use Abstract class	If various implementation of different nature and requires the member with same signature ,then use interface
Performance	It is faster to access the implemented class member	It takes time to find the members of the corresponding class
Extensibility (Versioning)	If any changes made to the abstract class ,not necessarily we need to change all the implementation classes	If any changes made to the interfaces , changes should be made in all the implemented classes
Field and Constants	Fields and constants can be defined	No fields and constants can be defined

Mona Abdelmonem **monasallam400@gmail.com**

What is static function?

Static functions are those functions that can be called without creating an object of the class. That means, Static methods do not use any instance variables of any object of the class they are defined in.

Static methods can not be overridden. They **are stored in heap** space of the memory.

What are virtual functions?

It is same as function overriding

A Virtual function is a member function that is declared within a base class and is redefined (overridden) by a derived class.

It is used to achieve runtime polymorphism.

A Virtual function is overridden.

What are pure virtual functions?

A pure virtual function is that function which have no definition. That means a virtual function that doesn't need implementation is called pure virtual function.

A pure virtual function have not definitions but we must override that function in the derived class, otherwise the derived class will also become abstract class.

What is Constructor?

It is same as functions but it's name should be same as its class name and must have no return type.

because

It is called when an object of the class is created. At the time of calling constructor, space for the object is allocated in the memory.

USE OF CONSTRUCTOR:-

We use constructor to assign values to the variables at the time of object creation.

Mona Abdelmonem **monasallam400@gmail.com**

What are the types of Constructor?

Constructor have following types –

- Default constructor
- Parameterized constructor
- Copy constructor
- Private constructor

What is default constructor?

A constructor with 0 parameters is known as default constructor.

What is private constructor?

if a constructor is declared private, we cannot create an object of the class.

What is copy constructor?

A copy constructor is that constructor which use existing object to create a new object.

It copy variables from another object of the same class to create a new object.

What is Parameterized constructor?

Constructors that can take at least one argument are termed as parameterized constructors

What is destructor?

Destructor is a type of member function which is used to destroy an object.

It is called automatically when the object goes out of scope or destroyed by a call to delete.

Destructor is called because
It destroy the objects when they are no longer in use.

A destructor has the same name as the class, preceded by a tilde (~).

Mona Abdelmonem **monasallam400@gmail.com**

Prototype design pattern means -- sample.

Shallow Copy and Deep Copy

Shallow Copy and Deep Copy play important role in copying the objects in Prototype Design Pattern.

Shallow copy

In Shallow copy ,an object is created by simply copying the data of all variables of the original object

In the case of **reference type**, it will only copy the reference, not the referred object itself.

Deep Copy

In Deep Copy , an object is created by copying data of all variables and it also allocates memory with the same value to the object.
In the case of reference type, it copy the reference as well as object itself.

Pointers

A Pointer is a variable that stores the memory address of another variable as it's value.

* - value at

& - address of

example:-- int a =10;
address of a=1048

```
int a=10;

int *ptr;
ptr=&a;
```

Now ptr store the address of 'a'

cout<<ptr;

output- 1048

cout<< *ptr;

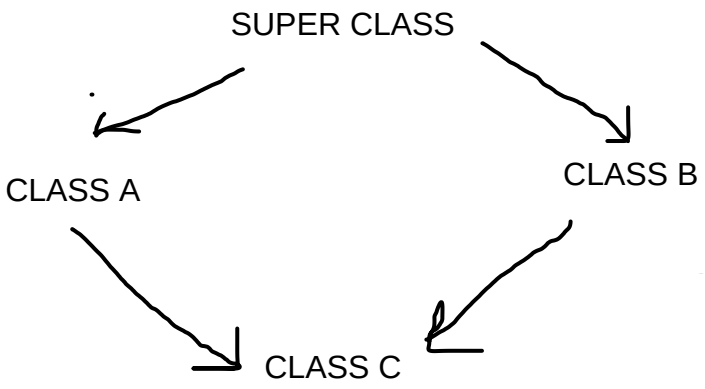
output- 10

EASIEST

DIAMOND PROBLEM

A diamond pattern is created when the child class inherits from the two base/parent class, and these parent classes have inherited from one common grandparent class(superclass).

As shown in the figure, ClassC inherits the traits of SuperClass twice—once from ClassA and again from ClassB. This creates confusion since the compiler fails to understand which way to go.



Solution to Diamond Problem

The solution is to use the keyword "virtual" on the two parent classes, ClassA and ClassB

```
ex. Class A : virtual public Super Class
        {
        }

    Class B : virtual public Super Class
        {
        }
```

In C++ we can pass arguments into a function in different ways. These different ways are

- Call by Value
- Call by Reference
- Call by Address

Sometimes the call by address is referred to as call by reference, but they are different in C++. In call by address, we use pointer variables to send the exact memory address, but in call by reference we pass the reference variable (alias of that variable). This feature is not present in C, there we have to pass the pointer to get that effect. In this section we will see what are the advantages of call by reference over call by value, and where to use them

Mona Abdelmonem **monasallam400@gmail.com**

Call by Value

In call by value, the actual value that is passed as argument is not changed after performing some operation on it. When call by value is used, it creates a copy of that variable into the memory. When the value is changed, it changes the value of that copy, the actual value remains the same.

Example Code

```
#include<iostream>
using namespace std;

void my_function(int x) {
  x = 50;
  cout << "Value of x from my_function: " << x << endl;
}

main() {
  int x = 10;
  my_function(x);
  cout << "Value of x from main function: " << x;
}
```

Output

```
Value of x from my_function: 50
Value of x from main function: 10
```

Mona Abdelmonem **monasallam400@gmail.com**

Call by Reference

In call by reference the actual value that is passed as argument is changed after performing some operation on it. When call by reference is used, it creates a copy of the reference of that variable into the memory. It uses a reference to get the value. So when the value is changed using the reference it changes the value of the actual variable.

```
#include<iostream>
using namespace std;

void my_function(int &x) {
  x = 50;
  cout << "Value of x from my_function: " << x << endl;
}

main() {
  int x = 10;
  my_function(x);
  cout << "Value of x from main function: " << x;
}
```

Output

Value of x from my_function: 50

Value of x from main function: 50

Where to use Call by reference?

- The call by reference is mainly used when we want to change the value of the passed argument into the invoker function.
- One function can return only one value. When we need more than one value from a function, we can pass them as an output argument in this manner.

Pointers

Mona Abdelmonem **monasallam400@gmail.com**