what is oops?
It is basically a programming style that uses the concept of class and object in programming. The popular object-oriented programming languages are c++, java, python, PHP, c#, etc. The main objective of OOPs is to implement real-world entities such as polymorphism, inheritance, encapsulation, abstraction, etc.

what is class?
a user-defined type that describes what a particular kind of object will look like. Thus, a class is a template or blueprint for an object. A class contains variables, methods, and constructors.

```
class class_name{
    // properties
    // methods
};
```

methods - functions inside class.

Constructor:-Constructors are special class functionsthat perform the initialization of every object. In C++, the constructor is automatically called when an object is created. It is a special method of the class because it does not have any return type. It has the same name as the class itself.
There are three types of constructors in C++:
★ Default constructor
★ Parameterized Constructor - takes the arguments

```
class A{
    public:
    // variables or data members
    int a;
    // constructor
    A(){
        a=8;
    }
    // methods
    void f1(){
    }
};
```

★ Copy Constructor - These are a particular type of constructor that takes an object as an argument and copies values of one object’s data members into another object. We pass the class object into another object of the same class in this constructor. As the name suggests, you Copy means to copy the values of one Object into another Object of Class. This is used for Copying the values of a class object into another object of a class, so we call them Copy constructor and for copying the values.

```
#include <bits/stdc++.h>
using namespace std;

class smartphone{
private:
    // Data Members(Properties)
    string model;
    int year_of_manufacture;
    bool _5g_supported;

public:
    // default constructor
    smartphone()
    {
        model = "unknown";

        year_of_manufacture = 0;
        _5g_supported = false;
    }
    // parameterized constructor
    smartphone(string model_string, int manufacture, bool _5g_)
    {
        // initialising data members
        model = model_string;
        year_of_manufacture = manufacture;
        _5g_supported = _5g_;
    }
    // copy constructor
    smartphone(smartphone &obj)
    {
        // copies data of the obj parameter
        model = obj.model;
        year_of_manufacture = obj.year_of_manufacture;
        _5g_supported = obj._5g_supported;
    }
};

int main()
{
    // creating objects of smartphone class

    //  using default constructor
    smartphone unknown;
    //  using parameterized constructor
    smartphone iphone("iphone 11", 2019, false);
    // using copy constructor
    smartphone iphone_2(iphone);
    return 0;
}
```

why do we use private constructor?

constructor overloading - having more than one constructor with different parameters so that every constructor can perform a different task.

Destructor - A destructor is a special member function that works just opposite to a constructor; unlike constructors that are used for initializing an object, destructors destroy (or delete) the object. The purpose of the destructor is to free the resources that the object may have acquired during its lifetime.
start with tilde(~) sign.
no parameters allowed
no return type

```
#include <iostream>

class MyClass {
public:
    // Constructor
    MyClass() {
        std::cout << "Constructor called." << std::endl;
    }

    // Destructor
    ~MyClass() {
        std::cout << "Destructor called." << std::endl;
    }
};

int main() {
    {
        MyClass obj; // Object created within a block
    } // Object goes out of scope, destructor is automatically called

    std::cout << "After the block." << std::endl;

    return 0;
}

output:
Constructor called.
Destructor called.
After the block.
```

when object is created statically then the object get destroyed automatically(destructor is called automatically).
A destructor function is called automatically when:
➔ the object goes out of scope

➔ the program ends
➔ a scope (the { } parenthesis) containing local variable ends.
➔ a delete operator is called

when object is created dynamically then the we need to maually destroy the object.

```
delete object_name;
```

what is object?
An object is an instance of a Class. It is an identifiable entity with some characteristics and behavior. Objects are the basic units of object-oriented programming.

```
// Syntax to create an object in C++:
class_name objectName;
or
class_name objectName = class_name();
// Syntax to create an object dynamically in C++:
class_name* objectName = new class_name();
```

```
class A{
    public:
    int a;
    A(){

    }
    void f1(){

    }
};

int main(){
    // static creation of object
    A obj1;
    // static creation of object
    A obj2 = A();

    // dynamic creation of object
    A* obj3 = new A();

    return 0;
}
```

why do we need oops?
❖ To make the development and maintenance of projects more effortless.
❖ To provide the feature of data hiding that is good for security concerns.

❖ We can solve real-world problems if we are using object-oriented programming.
❖ It ensures code reusability.
❖ It lets us write generic code: which will work with a range of data, so we don't have to write basic stuff over and over again.
❖ Problems can be divided into subparts.
❖ It increases thereadability, understandability, andmaintainability of the code.
❖ Data and code are bound together by encapsulation.

disadvantages of opps:
❖ Requires pre-work and proper planning.
❖ In certain scenarios, programs can consume a large amount of memory.
❖ Not suitable for a small problem.
❖ Proper documentation is required for later use.

diffenences between class and object.
class is a blueprint of object and used to create object while object is an instance of the class.
no memory is allocated when a class is created but memory is created when object is created.

padding - processor doesnot read 1 byte at a time from memory it reads 1 word at a time.
1 word = 4 bytes in 32-bit architechture.
1 word = 8 bytes in 64-bit architechture.
in one cycle one word is read by the processor from the memory.

Size of char : 1
Size of short int : 2
Size of int : 4
Size of long : 8
Size of float : 4
Size of double : 8

each data type start its memory allocation from its multiple.

```
#include <bits/stdc++.h>
using namespace std;

class Employee{
    char name;
    char x;
    short int age;
    short int val;
    int y;
};

int main(){
    Employee aman;
    cout<<sizeof(aman)<<endl;
    return 0;
}
```

it is done to save cpu cycle.

If want to save memory use #pragma pack(1) above class.

Access Modifiers - used to assign access to the class members. It sets some restrictions on the class members from accessing the outside functions directly.
Public: All the class members and methods can be accessed everywhere (inside and outside the class).
Private: All the class members and methods can be acceessed only inside the same class.
By default all the members and methods are private.
Protected: All the class members and methods can be acceessed in the same class and derived/child class.

```
class person {
    // nothing written so private
        int a;
    private:
        int b;  // private
    public:
        int c;  // public
    protected:
        int d;  // protected
};
```

this - this pointer holds the address of the current object. In simple words, you can say that this pointer points to the current object of the class. It allows the member function to access the data members and member functions of the object it is associated with.
There can be three main usages of this keyword in C++.

● It can be used to refer to a current class instance variable.
● It can be used to pass the current object as a parameter to another method.
● It can be used to declare indexers.

Shallow and Deep Copy

const
initialisation list
static keyword : it belongs to class rather than object.
static datatype.
static function - can access only static members.this keyword can not be used in the static function.

using scope resolution operator ::

virtual keyword - used to override the function in derived class.
used to solve diamond problem where only one instance on the same function is created.

friend keyword

encapsulation - data or information hiding
wrapping up data members and functions.
fully encapsualted : all data members or variables are private.

abstraction - implementation hiding.

inherience
parent class / base class / super class
child class / sub class

child	public	protected	private
parent			
public	public	protected	private
protected	protected	protected	private
private	NA	NA	NA

types of inheritence

1. single inheritence
2. multilevel inheritence
3. multiple inheritence - inherit from two or more super class. have more than one parent.
4. hierarchical inheritence - parent have more than one child.
5. Hybrid inheritence - combination of two or more inheritence.

Inheritence ambiguity.

- can be solved using scope resolution operator.

polymorphism - many forms.

1. compile time polymorphism.
    1. function overloading.
    2. operator overloading.
2. run time polymorphism. - function overriding.

differences between compile time and runtime polymorphism.

differences between abstraction and encapsulation.