What are the basic concepts of OOP?

The four basic concepts of OOP are:
a.) Abstraction
b.) Polymorphism
c.) Inheritance
d.) Encapsulation

What is dynamic or run time polymorphism?

- It is also referred to as method overriding. Here, there can be two methods with same name and signature but different implementation.
- The function here is resolved during run time rather than compile time.

What is Encapsulation?

- It is a technique to hide the properties and behaviours of an object.
- The access is provided only as required.
- It prevents other objects from altering or accessing the properties of an encapsulated object.

Differentiate between abstraction and encapsulation.

- Abstraction is design oriented while abstraction is implementation oriented.

- The focus of abstraction is on the interface i.e. the outside view of the object while encapsulation prevents other objects or methods from looking into the properties and behaviour of that object.

What is Inheritance?

- It is the process which allows the objects of one class to acquire the properties of objects of another class.
- The class that inherits is called sub-class while the class from which the object is inherited is called superclass.
- Inheritance helps in re-using the code and polymorphism.

Explain method overriding.

- When a subclass declares a method possessing similar arguments as a method declared by one of its superclass, method overriding takes place.
- Using this technique, the behaviour specific to a particular subclass can be defined.
- The methods marked public, final or static can not be overridden

Can you give some examples of tokens?

Some of the examples of tokens are:
- Keywords,
- Identifiers,
- Constants,

- Operators,
- Brackets,
- Commas.

What is friend function?

- Friend function is a friend of a class.
- It is allowed to access Public, private or protected data of that class.
- It can be declared anywhere in the class declaration
- It doesn't have any effect of access control keywords like private, public or protected.

Define Modularity?

- It is the property of big a system divided into small modules which can be integrated as per the requirement.

Explain: a.) Persistence. b.) Colloboration

a.) Persistence.
- It is the property of an object which extends its existence into space and time.

b.) Colloboration
- Process by which several objects co-operate to deliver a higher level result.

What is a ternary operator?

- It is also called as a conditional operator.
- It is an operator that can take three arguments.
- The data types of arguments and results are different.

What are sealed modifiers?

- Access modifiers which cannot be inherited by other methods.
- They can be applied to properties, events and methods but not to static members.

Explain: a.) Static binding b.) Dynamic binding

a.) Static binding-
It is a binding in which the name of the class can be associated with it during compilation. Also referred to as early binding.

b.) Dynamic binding –
It is a binding in which the name of the class can be associated with it during execution time. Also referred to as late binding.

What are the different ways for a method to be overloaded?

A method can be overloaded by:
- Varying the numbers of parameters
- Using different data types for the parameters

- Using different sequence of the parameters.

What is composition? Explain the purpose of composition.

Assembling existing components instead of creating new ones is called composition. In OOP it is called as objects composition. It is the process of placing object in another object. It is the model of has-a relationship. An employee object can contain an object of type project which is another object.

Purpose of composition:
A model by value aggregation can be derived by using composition which is semantically equivalent to an attribute.

The lifetime is coincident part as a whole for both objects

If one part is removed, the whole part also removed with out explicit removal of individual parts

Composition can be used to model by-value aggregation which is semantically equivalent to an attribute.

What is an abstraction and why is it important?

Representing essential features by hiding the background process / details. In other words, it is specifying what to do but not how to do.

Abstraction is important at the conceptual level of an application. It helps in understanding clearly, what the process and the flow of an application. The abstraction allows the complete flow and development of an application into a structured action. The conceptual level of project execution is derived from abstraction which the gateway for the entire application development.

How a program can be structured using the concept of Object oriented programming?

The program can be structured by splitting the one big task into subtasks, create some function for the tasks that has to be worked upon and then write a program to instruct the computer to perform the desired calculations on the basis of which the output will be generated. As, the large amount of data or tasks will be separated into less complex programs then it will be easier to manage. In object oriented program the program structure is maintained according to the objects made up from the class. Every function is in a particular class that has an object calling certain properties and methods for implementation. It can manage the complexity of the large pieces and will be able to create a modular approach towards the programming methodology.

What are the features that are provided to make a program modular?

To create a program requires self sufficient resources, the way to make a piece of code modular so that it can be handled properly. This shows the relationship between different objects or in

between the same objects. Different functions have to be used to keep the code different from the data provided. Object oriented programming allows creating a modular approach to make it more abstract and create the interactions between them.
The features that are being provided in object oriented programming is:
• **Encapsulation:** it deals with the grouping of data and functions together at one place or it can be said in an object that defines the interface of their interaction and the relationship between them.

• **Inheritance:** it deals with the relationship that exists between the parent and child classes. A child can inherit the properties of the parent and it remains helpful in creating a hierarchy in the system to keep things more modular.

• **Polymorphism:** is a feature that deals with different values that can be used by only one function. It doesn't have to re-create the same function again and again for different values but, it uses the same function to call different data types.

Write a program to show the inheritance used in a class.

Inheritance is a way to define the hierarchy of the related classes. It allows the property sharing from one class to another class. It is an easy way to inherit the properties of the parent by the child and use the same properties again and again without redefining the class. The program is as follows it shows a class student that inherits the properties (methods+ data) from the teacher class.

```
class teacher
{
  public:
  t(){}
};

class student : public teacher
{
  public:
  s(){}
};

void use(teacher const& some)
{
  some.t();
}
```

What is the function of pure virtual functions?

Pure virtual function in object oriented programming is called as virtual method that acts as a function allowing its behavior to be overridden by the class that is inheriting the pure virtual function with the same signature. This is used in case of polymorphism. It is used when a base class is being driven by the derived class and an object of the derived class referred as base or derived class type. When a derived class overrides the base class method then the output or the

behavior will be called as ambiguous. To use the virtual function a virtual keyword is used. This allows the function to be defined in every derived class and use the functionality of it.

What is the syntax of object oriented method call?

The object oriented syntax tells about the object oriented features and methodologies that are present and shown as:
target.method(arguments);
This is the line that can be used to call the functions, but not invoking the methods on an object. A function is made up of both the operands and operators and they are very essential to call a method. So, the method will be written as:
a = sub(c,b);
Object oriented syntax allows easy to write ways and some specified rules that has to be remembered while programming. The operator has to always come in the middle of the statement. A class is made for every module and it also represents the template used for a class made up of objects.

What is the difference between data hiding and implementation?

- Data hiding is the concept of hiding the data from the other objects. It allows the data to be hidden from other objects that can prevent the use of any other objects. This way the direct manipulation of the object can't be done and the cell will hide the internal mechanism of it. A method uses an interface such as primary, public, etc. The implementation keeps the separate out data and let the user implement the structure or the data that is being provided already.

- Implementation allows the object to be changes according to the requirement. It deals with objects communication via messages in which when a target object then the implementation can be done with the method. The example is given below:

```
class teacher
{
  // properties of the teacher
  String name[];
  int numb;

  // behavior
  public void()
  // this is the function that is given and it will include all your code
}
```

What are the properties that are defined in an Object model?

Object oriented programming always consists of an object model that includes state and behavior. The state defines the present condition of the system or an entity. The behavior includes the data and the operations that are performed on those data. Object represents the group of related functions and data structure that is used to give functionality to the functions. The

functions that are being represented are the object models. It consists of the data structure, data type and the instance variables. Methods are used to hide the variables from the user and and define a high level in object model. Object model also consists of the flow properties through which the object can be accessed with a defined relationship.

What is the function of messaging metaphor?

Object oriented programming is having its own metaphors that are associated with the program or the code that has a role to play in the program. Messaging metaphor represent the relationship between the actors that are involved it and the roles each are playing. This allows easy passing of the responsibilities and information to the processes that are in contact with the one which is passing the information. Messaging method allows the user to use the objects rather than their functions. As, the objects perform all the actions that is been done by the methods or the functions alone. Objects consists of state and behavior so they are not active rather passive, but they act as an agent to allow the program's activity to be performed easily. This metaphor keeps all the objects separate each with the defined roles.

Why is message passing between the objects important?

Objects use the metaphor concept to pass the messages between different objects. The objects communicate through the messages only. The first step is to call the method as the functions are called in a normal way. The moment a function is called by an object a message will be passed to all the objects to allow the operations to be performed. Then the abstract method is separated from the particular data to show the correct behavior of the state. Object oriented programming keeps a start function that initiates the operation. Then the information that is contained in the object is archiving and then the program gets executed according to the method or the function. The different objects will have different functions to be performed. There are few abstract behaviors and to make it concrete the method has to be associated with an object.

What is the main use of message metaphor in object-oriented programming?

Methods belong to objects and they are invoked by using a particular receiver only. The receiver can be either owner of method or the data structure whose method will be. The receivers can have different implementation that can be done for the same method. Different receivers can also perform different operations and produce different responses in case of the same message. The result of the message will also depend on the object that is receiving the message. The message or the request can be separated from the receiver who is the owner of the function that is being called to capture the behavior of the abstract data type and implement the function properly.

Why is class hierarchies managed in object-oriented programming?

Class is used to store some methods and behavior. Class is used to access through an object and create a relationship between all the subclasses. A class can be made a super class of another classes that are present will be called as the subclass of it. Many classes can be linked together in a hierarchy of inheritance. The inheritance hierarchy consists of a root class and this is also

called as the super class. The root class hierarchy sends the branch downwards. This class inherits the features from the super class and from all the other classes that are at the top level in the hierarchy. Each class consists of the class definition and the properties that are inherited from other classes. Every class consists of one super class and can consist of any number of subclasses.

What are the methods required to create changes in Subclasses?

- Subclass is a class that is under the super class and it inherits the property of the subclass as well. The class definition can be expanded by using the class definition that inherits the methods and variables that are used in the program. The subclasses consist of their own methods and variables with the super class variables as well.
The methods that are used:
- Subclass can modify the behavior that it inherits from the existing method.
- Subclass can extend the behavior of the object by inheriting the new methods and functionalities. Subclass overrides any method that is being taken from the super class.
- Subclass fills the super class properties to make it more specific and specialized. There is addition of program or code take place and the replacement occurs only at the time of new functionalities coming up.

What is the function of dynamic typing?

Dynamic typing is a way to allow the compiler to replace an existing value with a type that is not functioning properly. There can be warning signs like: incompatible types in assignment. This warning sign shows that there is type checking that casts the object from one to another to make it compatible with the current system. The dynamic typing checks for the type of object that is being during the compile time and during the run time the value doesn't change. If the class of the variable can be found during the run time then it will be possible to find out the class of the variable. Dynamic binding provides the output to the dynamic binding. It allows the binding between the objects to be done at runtime.

Why dynamic binding is used in object-oriented programming?

Dynamic binding allows delaying the decision of running the program. Dynamic binding is used to perform messaging transparently and routinely by allowing different objects to be binded on the run time. It allows the declared pointer to find out the assigned values of a give procedure. Messages that are given with the programs are invoked indirectly. Every method has a message expression that has to be found for implementation. The class of the receiver is checked to locate the implementation of the method that is mentioned in the message. When at run time the method is run then the complier dynamically bind the objects.

The example shows the dynamic binding of the object:
int strcmp(const char *, const char *);

int strcasecmp(const char *, const char *);

```
int (* compare)(const char *, const char *);

if ( **argv == 'i' )

compare = strcasecmp;

else

compare = strcmp;
```

What is the purpose of late binding in object-oriented programming?

Late binding is done at the run time and it requires the messages to be passed to the receiver statically by writing in the code or the program. An object can be attached to its own class or any class that inherits the properties of the class. The compiler has difficulty in understanding whether the source is specified in the type declaration or not. As, if an object is given then on run time it is not possible for the compiler to know its class then for that the instance of the specified class is binded with the method that is defined for that particular class to the messages. The late binding is done during the linking of the method to the procedure of the program. It consists of strict compile time rules that put the constraints on the program in execution.

What are the major differences between late and dynamic binding?

- Late binding allows the rules to be set up during the compilation time only, whereas the dynamic binding allows the methods to be dynamically bound together with the function at run time only.

- Late binding consists of some constraints strictly on the basis of compile time, whereas dynamic binding is unconstrained.

- Late binding doesn't exist with the dynamic binding, whereas dynamic binding can exist with the dynamic typing.

- Dynamic binding allows the code that is written to send the messages to the objects, whereas late binding just show the relationship between the sender and receiver.

Why dynamic loading is used in object-oriented programming?

Dynamic loading is required to load the object or class when the demand for it rises. Before the loading in the linking phase all the links of the objects that are linked together, gets contained in one file. Dynamic loading loads the entire environment to be loaded at once. It allows different parts of an executable program to be given in different files. Each piece in the program is

dynamically loading and some user actions can be performed on the software. It focuses on the programs that are in the memory. Large programs always required to be loaded at start. The modules can be dynamically added. The advantage of this is that an entire program that doesn't require the single feedback. It allows the extensibility of the program that has to be done to make an application.

Why are Outlet Connections required in object-oriented programming?

The object network is arranged using the design of object-oriented programming. The network is having dynamic requirements and can't remain static any time while running a program. The network has to maintain a relationship between the objects and some roles are assigned from time to time using a script. Outlet connections depend on the message that is passed between the objects. Messages are used to identify an object and it is usually used for communication with the receiver. The messages have to be recorded in the database using the object connections. There can be made some instances to keep the track of the objects that communicates with one another. The outlet connections record the messages and store them on the program network. The objects are having the four outlets like an agent, a friend, neighbor and a boss and the roles are interchanged from each other.

What are Extrinsic and Intrinsic Connections in object-oriented programming?

There are different objects that are part of one another. These objects have intrinsic and extrinsic properties that are applied. An object keeps other objects in the list of the processes. There are some building objects that have a list of all the objects that are used in the program. The objects that are used belong to building object. The extrinsic behavior comes when one object communicates with another object. Intrinsic outlet are different and they are used when an object is about to be freed or archived in a file. The archived message has to be kept in a file; these files are dependent to the program subcomponents. Extrinsic outlets allows the capture of the program for an organization at higher level and then the connection related settings to be recorded and stored using the independent program concept.

What is the difference between Aggregation and composition?

Aggregation is a collection of the entire program that consists of only one object and this object sends messages to itself and other objects by using the method of polymorphism. The modularity concept is used for the classes and program design to set the network of objects that are interconnected to each other. In this the program is hidden from the class definition. Composition on the other hand is a way to combine all the small sub-components to provide the functionalities in modular format. Doing this will make the structure less usable. The objects in this can be made part of another in this way the object only exists not anything else associated with an object.

1) What is difference between JDK,JRE and JVM?

JVM

JVM is an acronym for Java Virtual Machine, it is an abstract machine which provides the runtime environment in which java bytecode can be executed. It is a specification.

JVMs are available for many hardware and software platforms (so JVM is platform dependent).

JRE

JRE stands for Java Runtime Environment. It is the implementation of JVM.

JDK

JDK is an acronym for Java Development Kit. It physically exists. It contains JRE + development tools.

more details...

---

2) How many types of memory areas are allocated by JVM?

Many types:

1. Class(Method) Area
2. Heap
3. Stack
4. Program Counter Register
5. Native Method Stack

more details...

---

3) What is JIT compiler?

Just-In-Time(JIT) compiler: It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

---

4) What is platform?

A platform is basically the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform.

---

5) What is the main difference between Java platform and other platforms?

The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms.It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

---

6) What gives Java its 'write once and run anywhere' nature?

The bytecode. Java is compiled to be a byte code which is the intermediate language between source code and machine code. This byte code is not platform specific and hence can be fed to any platform.

---

7) What is classloader?

The classloader is a subsystem of JVM that is used to load classes and interfaces.There are many types of classloaders e.g. Bootstrap classloader, Extension classloader, System classloader, Plugin classloader etc.

---

8) Is Empty .java file name a valid source file name?

Yes, save your java file by .java only, compile it by **javac .java** and run by **java yourclassname** Let's take a simple example:

```
1. //save by .java only
2. class A{
3. public static void main(String args[]){
4. System.out.println("Hello java");
5. }
6. }
7. //compile by javac .java
8. //run by     java A
```

compile it by **javac .java**

run it by **java A**

---

9) Is delete,next,main,exit or null keyword in java?

No.

---

10) If I don't provide any arguments on the command line, then the String array of Main method will be empty or null?

It is empty. But not null.

---

11) What if I write static public void instead of public static void?

Program compiles and runs properly.

---

12) What is the default value of the local variables?

The local variables are not initialized to any default value, neither primitives nor object references.

1. What is OOPS?

OOPS is abbreviated as Object Oriented Programming system in which programs are considered as a collection of objects. Each object is nothing but an instance of a class.

2. Write basic concepts of OOPS?

Following are the concepts of OOPS and are as follows:.

1. Abstraction.
2. Encapsulation.
3. Inheritance.
4. Polymorphism.

3. What is a class?

A class is simply a representation of a type of object. It is the blueprint/ plan/ template that describe the details of an object.

4. What is an object?

Object is termed as an instance of a class, and it has its own state, behavior and identity.

5. What is Encapsulation?

Encapsulation is an attribute of an object, and it contains all data which is hidden. That hidden data can be restricted to the members of that class.

Levels are Public,Protected, Private, Internal and Protected Internal.

6. What is Polymorphism?

Polymorphism is nothing butassigning behavior or value in a subclass to something that was already declared in the main class. Simply, polymorphism takes more than one form.

7. What is Inheritance?

Inheritance is a concept where one class shares the structure and behavior defined in another class. Ifinheritance applied on one class is called Single Inheritance, and if it depends on multiple classes, then it is called multiple Inheritance.

8. What are manipulators?

Manipulators are the functions which can be used in conjunction with the insertion (<<) and extraction (>>) operators on an object. Examples are endl and setw.

9. Define a constructor?

Constructor is a method used to initialize the state of an object, and it gets invoked at the time of object creation. Rules forconstructor are:.

- Constructor Name should be same asclass name.
- Constructor must have no return type.

10. Define Destructor?

Destructor is a method which is automatically called when the object ismade ofscope or destroyed. Destructor name is also same asclass name but with the tilde symbol before the name.

11. What is Inline function?

Inline function is a technique used by the compilers and instructs to insert complete body of the function wherever that function is used in the program source code.

12. What is avirtual function?

Virtual function is a member function ofclass and its functionality can be overridden in its derived class. This function can be implemented by using a keyword called virtual, and it can be given during function declaration.

Virtual function can be achieved in C++, and it can be achieved in C Languageby using function pointers or pointers to function.

13. What isfriend function?

Friend function is a friend of a class that is allowed to access to Public, private or protected data in that same class. If the function is defined outside the class cannot access such information.

Friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public or protected.

14. What is function overloading?

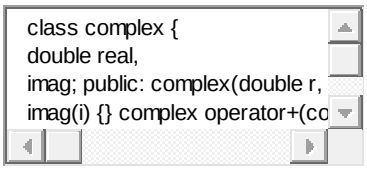
Function overloading is defined as a normal function, but it has the ability to perform different tasks. It allowscreation of several methods with the same name which differ from each other by type of input and output of the function.

Example
```
void add(int& a, int& b);
void add(double& a, double& b);
void add(struct bob& a, struct bob& b);
```

15. What is operator overloading?

Operator overloading is a function where different operators are applied and depends on the arguments. Operator,-,* can be used to pass through the function , and it has their own precedence to execute.

Example:




```
1 class complex {
2 double real,
3 imag; public: complex(double r, double i) : real(r),
4 imag(i) {} complex operator+(complex a, complex b);
5 complex operator*(complex a, complex b);
6 complex& operator=(complex a, complex b);
7 }
```

a=1.2, b=6

16. What is an abstract class?

An abstract class is a class which cannot be instantiated. Creation of an object is not possible with abstract class , but it can be inherited. An abstract class can contain only Abstract method. Java allows only abstract method in abstract class while for other language it allows non-abstract method as well.

17. What is a ternary operator?

Ternary operator is said to be an operator which takes three arguments. Arguments and results are of different data types , and it is depends on the function. Ternary operator is also called asconditional operator.

18. What is the use of finalize method?

Finalize method helps to perform cleanup operations on the resources which are not currently used. Finalize method is protected , and it is accessible only through this class or by a derived class.

19. What are different types of arguments?

A parameter is a variable used during the declaration of the function or subroutine and arguments are passed to the function , and it should match with the parameter defined. There are two types of Arguments.

- Call by Value – Value passed will get modified only inside the function , and it returns the same value whatever it is passed it into the function.

- Call by Reference – Value passed will get modified in both inside and outside the functions and it returns the same or different value.

20. What is super keyword?

Super keyword is used to invoke overridden method which overrides one of its superclass methods. This keyword allows to access overridden methods and also to access hidden members of the superclass.

It also forwards a call from a constructor to a constructor in the superclass.

21. What is method overriding?

Method overriding is a feature that allows sub class to provide implementation of a method that is already defined in the main class. This will overrides the implementation in the superclass by providing the same method name, same parameter and same return type.

22. What is an interface?

An interface is a collection of abstract method. If the class implements an inheritance, and then thereby inherits all the abstract methods of an interface.

23. What is exception handling?

Exception is an event that occurs during the execution of a program. Exceptions can be of any type – Run time exception, Error exceptions. Those exceptions are handled properly through exception handling mechanism like try, catch and throw keywords.

24. What are tokens?

Token is recognized by a compiler and it cannot be broken down into component elements. Keywords, identifiers, constants, string literals and operators are examples of tokens.

Even punctuation characters are also considered as tokens – Brackets, Commas, Braces and Parentheses.

25. Difference between overloading and overriding?

Overloading is static binding whereas Overriding is dynamic binding. Overloading is nothing but the same method with different arguments , and it may or may not return the same value in the same class itself.

Overriding is the same method names with same arguments and return types associates with the class and its child class.

26. Difference between class and an object?

An object is an instance of a class. Objects hold any information , but classes don't have any information. Definition of properties and functions can be done at class and can be used by the object.

Class can have sub-classes, and an object doesn't have sub-objects.

27. What is an abstraction?

Abstraction is a good feature of OOPS , and it shows only the necessary details to the client of an object. Means, it shows only necessary details for an object, not the inner details of an object. Example – When you want to switch On television, it not necessary to show all the functions of TV. Whatever is required to switch on TV will be showed by using abstract class.

28. What are access modifiers?

Access modifiers determine the scope of the method or variables that can be accessed from other various objects or classes. There are 5 types of access modifiers , and they are as follows:.

- Private.
- Protected.
- Public.
- Friend.
- Protected Friend.

29. What is sealed modifiers?

Sealed modifiers are the access modifiers where it cannot be inherited by the methods. Sealed modifiers can also be applied to properties, events and methods. This modifier cannot be applied to static members.

30. How can we call the base method without creating an instance?

Yes, it is possible to call the base method without creating an instance. And that method should be,.

Static method.

Doing inheritance from that class.-Use Base Keyword from derived class.

31. What is the difference between new and override?

The new modifier instructs the compiler to use the new implementation instead of the base class function. Whereas, Override modifier helps to override the base class function.

32. What are the various types of constructors?

There are three various types of constructors , and they are as follows:.

– Default Constructor – With no parameters.

– Parametric Constructor – With Parameters. Create a new instance of a class and also passing arguments simultaneously.

– Copy Constructor – Which creates a new object as a copy of an existing object.

33. What is early and late binding?

Early binding refers to assignment of values to variables during design time whereas late binding refers to assignment of values to variables during run time.

34. What is 'this' pointer?

THIS pointer refers to the current object of a class. THIS keyword is used as a pointer which differentiates between the current object with the global object. Basically, it refers to the current object.

35. What is the difference between structure and a class?

Structure default access type is public , but class access type is private. A structure is used for grouping data whereas class can be used for grouping data and methods. Structures are exclusively used for dataand it doesn't require strict validation , but classes are used to encapsulates and inherit data which requires strict validation.

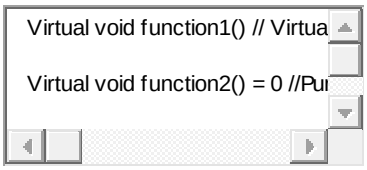
36. What is the default access modifier in a class?

The default access modifier of a class is Private by default.

37. What is pure virtual function?

A pure virtual function is a function which can be overridden in the derived classbut cannot be defined. A virtual function can be declared as Pure by using the operator =0.

Example -.



1 Virtual void function1() // Virtual, Not pure
2
3 Virtual void function2() = 0 //Pure virtual

38. What are all the operators that cannot be overloaded?

Following are the operators that cannot be overloaded -.

1. Scope Resolution (::)
2. Member Selection (.)
3. Member selection through a pointer to function (.*)

39. What is dynamic or run time polymorphism?

Dynamic or Run time polymorphism is also known as method overriding in which call to an overridden function is resolved during run time, not at the compile time. It means having two or more methods with the same name,same signature but with different implementation.

40. Do we require parameter for constructors?

No, we do not require parameter for constructors.

41. What is a copy constructor?

This is a special constructor for creating a new object as a copy of an existing object. There will be always only on copy constructor that can be either defined by the user or the system.

42. What does the keyword virtual represented in the method definition?

It means, we can override the method.

43. Whether static method can use non static members?

False.

44. What arebase class, sub class and super class?

Base class is the most generalized class , and it is said to be a root class.

Sub class is a class that inherits from one or more base classes.

Super class is the parent class from which another class inherits.

45. What is static and dynamic binding?

Binding is nothing but the association of a name with the class. Static binding is a binding in which name can be associated with the class during compilation time , and it is also called as early Binding.

Dynamic binding is a binding in which name can be associated with the class during execution time , and it is also called as Late Binding.

46. How many instances can be created for an abstract class?

Zero instances will be created for an abstract class.

47. Which keyword can be used for overloading?

Operator keyword is used for overloading.

48. What is the default access specifier in a class definition?

Private access specifier is used in a class definition.

49. Which OOPS concept is used as reuse mechanism?

Inheritance is the OOPS concept that can be used as reuse mechanism.

50. Which OOPS concept exposes only necessary information to the calling functions?

Data Hiding / Abstraction

1. What is object-oriented programming (OOP)?

   OOP is a technique to develop logical modules, such as classes that contain properties, methods, fields, and events. An object is created in the program to represent a class. Therefore, an object encapsulates all the features, such as data and behavior that are associated to a class. OOP allows developers to develop modular programs and assemble them as software. Objects are used to access data and behaviors of different software modules, such as classes, namespaces, and sharable assemblies. .NET Framework supports only OOP languages, such as Visual Basic .NET, Visual C#, and Visual C++.

2. What is a class?

A class describes all the attributes of objects, as well as the methods that implement the behavior of member objects. It is a comprehensive data type, which represents a blue print of objects. It is a template of object.

A class can be defined as the primary building block of OOP. It also serves as a template that describes the properties, state, and behaviors common to a particular group of objects.

A class contains data and behavior of an entity. For example, the aircraft class can contain data, such as model number, category, and color and behavior, such as duration of flight, speed, and number of passengers. A class inherits the data members and behaviors of other classes by extending from them.

3. What is an object?

   They are instance of classes. It is a basic unit of a system. An object is an entity that has attributes, behavior, and identity. Attributes and behavior of an object are defined by the class definition.

4. What is the relationship between a class and an object?

   A class acts as a blue-print that defines the properties, states, and behaviors that are common to a number of objects. An object is an instance of the class. For example, you have a class called *Vehicle* and *Car* is the object of that class. You can create any number of objects for the class named *Vehicle*, such as *Van, Truck*, and *Auto*.

   The *new* operator is used to create an object of a class. When an object of a class is instantiated, the system allocates memory for every data member that is present in the class.

5. Explain the basic features of OOPs.

   The following are the four basic features of OOP:

   - **Abstraction** - Refers to the process of exposing only the relevant and essential data to the users without showing unnecessary information.
   - **Polymorphism** - Allows you to use an entity in multiple forms.
   - **Encapsulation** - Prevents the data from unwanted access by binding of code and data in a single unit called object.
   - **Inheritance** - Promotes the reusability of code and eliminates the use of redundant code. It is the property through which a child class obtains all the features defined in its parent class. When a class inherits the common properties of another class, the class inheriting the properties is called a derived class and the class that allows inheritance of its common properties is called a base class.

6. What is the difference between arrays and collection?

Array:

1. You need to specify the size of an array at the time of its declaration. It cannot be resized dynamically.
2. The members of an array should be of the same data type.

Collection:

3. The size of a collection can be adjusted dynamically, as per the user's requirement. It does not have fixed size.
4. Collection can have elements of different types.

7. What are collections and generics?

   A collection can be defined as a group of related items that can be referred to as a single unit. The *System.Collections* namespace provides you with many classes and interfaces. Some of them are - *ArrayList*, *List*, *Stack*, *ICollection*, *IEnumerable*, and *IDictionary*. Generics provide the type-safety to your class at the compile time. While creating a data structure, you never need to specify the data type at the time of declaration. The *System.Collections.Generic* namespace contains all the generic collections.

8. How can you prevent your class to be inherited further?

   You can prevent a class from being inherited further by defining it with the *sealed* keyword.

9. What is the index value of the first element in an array?

   In an array, the index value of the first element is *0* (zero).

10. Can you specify the accessibility modifier for methods inside the interface?

    All the methods inside an interface are always *public*, by default. You cannot specify any other access modifier for them.

11. Is it possible for a class to inherit the constructor of its base class?

    No, a class cannot inherit the constructor of its base class.

12. How is method overriding different from method overloading?

    Overriding involves the creation of two or more methods with the same name and same signature in different classes (one of them should be parent class and other should be child).

Overloading is a concept of using a method at different places with same name and different signatures within the same class.

13. What is the difference between a class and a structure?

    Class:

    1. A class is a reference type.
    2. While instantiating a class, CLR allocates memory for its instance in heap.
    3. Classes support inheritance.
    4. Variables of a class can be assigned as null.
    5. Class can contain constructor/destructor.

    Structure:

    6. A structure is a value type.
    7. In structure, memory is allocated on stack.
    8. Structures do not support inheritance.
    9. Structure members cannot have null values.
    10. Structure does not require constructor/destructor and members can be initialiazed automatically.

14. What are similarities between a class and a structure.

    Structures and classes are the two most important data structures that are used by programmers to build modular programs by using OOP languages, such as Visual Basic .NET, and Visual C#. The following are some of the similarities between a class and a structure:

    - Access specifiers, such as *public*, *private*, and *protected*, are identically used in structures and classes to restrict the access of their data and methods outside their body.
    - The access level for class members and struct members, including nested classes and structs, is private by default. Private nested types are not accessible from outside the containing type.
    - Both can have constructors, methods, properties, fields, constants, enumerations, events, and event handlers.
    - Both structures and classes can implement interfaces to use multiple-inheritance in code.
    - Both structures and classes can have constructors with parameter.
    - Both structures and classes can have delegates and events.

15. What is a multicast delegate?

    Each delegate object holds reference to a single method. However, it is possible for a delegate object to hold references of and invoke multiple methods. Such delegate objects are called multicast delegates or combinable delegates.

16. Can you declare an overridden method to be static if the original method is not static?

    No. Two virtual methods must have the same signature.

17. Why is the virtual keyword used in code?

    The *virtual* keyword is used while defining a class to specify that the methods and the properties of that class can be overridden in derived classes.

18. Can you allow a class to be inherited, but prevent a method from being overridden in C#?

    Yes. Just declare the class *public* and make the method *sealed*.

- Define enumeration?

Enumeration is defined as a value type that consists of a set of named values. These values are constants and are called enumerators. An enumeration type is declared using the *enum* keyword. Each enumerator in an enumeration is associated with an underlying type that is set, by default, on the enumerator. The following is an example that creates an enumeration to store different varieties of fruits:

enum Fruits {Mango, Apple, orange, Guava};

In the preceding example, an enumeration *Fruits* is created, where number *0* is associated with *Mango*, number *1* with *Apple*, number *2* with *Orange*, and number *3* with *Guava*. You can access the enumerators of an enumeration by these values.

- In which namespace, all .NET collection classes are contained?

The *System.Collections* namespace contains all the collection classes.

- Is it a good practice to handle exceptions in code?

Yes, you must handle exceptions in code so that you can deal with any unexpected situations that occur when a program is running. For example, dividing a number by zero or passing a string value to a variable that holds an integer value would result in an exception.

- Explain the concept of constructor?

Constructor is a special method of a class, which is called automatically when the instance of a class is created. It is created with the same name as the class and initializes all class members, whenever you access the class. The main features of a constructor are as follows:

- Constructors do not have any return type.
- Constructors can be overloaded.
- It is not mandatory to declare a constructor; it is invoked automatically by .NET Framework.

- Can you inherit private members of a class?

No, you cannot inherit *private* members of a class because *private* members are accessible only to that class and not outside that class.

- Does .NET support multiple inheritance?

.NET does not support multiple inheritance directly because in .NET, a class cannot inherit from more than one class. .NET supports multiple inheritance through interfaces

25. How has exception handling changed in .NET Framework 4.0?

    In .NET 4.0, a new namespace, *System.Runtime.ExceptionServices*, has been introduced which contains the following classes for handling exceptions in a better and advanced manner:

    - *HandleProcessCorruptedStateExceptionsAttribute* Class - Enables managed code to handle the corrupted state exceptions that occur in an operating system. These exceptions cannot be caught by specifying the *try...catch* block. To handle such exceptions, you can apply this attribute to the method that is assigned to handle these exceptions.
    - *FirstChanceExceptionEventArgs* Class - Generates an event whenever a managed exception first occurs in your code, before the common language runtime begins searching for event handlers.

26. What is a delegate?

    A delegate is similar to a class that is used for storing the reference to a method and invoking that method at runtime, as required. A delegate can hold the reference of only those methods whose signatures are same as that of the delegate. Some of the examples of delegates are type-safe functions, pointers, or callbacks.

27. What is the syntax to inherit from a class in C#?

    When a class is derived from another class, then the members of the base class become the members of the derived class. The access modifier used while accessing members of the base class specifies the access status of the base class members inside the derived class.

    The syntax to inherit a class from another class in C# is as follows:
    class MyNewClass : MyBaseclass

28. State the features of an interface.

    An interface is a template that contains only the signature of methods. The signature of a method consists of the numbers of parameters, the type of parameter (value, reference, or output), and the order of parameters. An interface has no implementation on its own because it contains only the definition of methods without any method body. An interface is defined using the *interface* keyword. Moreover, you cannot instantiate an interface. The various features of an interface are as follows:

    - An interface is used to implement multiple inheritance in code. This feature of an interface is quite different from that of abstract classes because a class cannot

derive the features of more than one class but can easily implement multiple interfaces.
    - It defines a specific set of methods and their arguments.
    - Variables in interface must be declared as *public*, *static*, and *final* while methods must be *public* and *abstract*.
    - A class implementing an interface must implement all of its methods.
    - An interface can derive from more than one interface.

29. Can you use the 'throws' clause to raise an exception?

    No, the *throws* clause cannot be used to raise an exception. The *throw* statement signals the occurrence of an exception during the execution of a program. When the program encounters a *throw* statement, the method terminates and returns the error to the calling method.

30. Define an array.

    An array is defined as a homogeneous collection of elements, stored at contiguous memory locations, which can be referred by the same variable name. All the elements of an array variable can be accessed by index values. An Index value specifies the position of a particular element in an array variable.

31. What are methods?

    Methods are the building blocks of a class, in which they are linked together to share and process data to produce the result. In other words, a method is a block of code that contains a series of statements and represents the behavior of a class. While declaring a method you need to specify the access specifier, the return value, the name of the method, and the method parameters. All these combined together is called the signature of the method.

32. What is a namespace?

    Namespace is considered as a container that contains functionally related group of classes and other types.

33. Do events have return type?

    No, events do not have return type.

34. What is the function of the Try-Catch-Finally block?

    The *try* block encloses those statements that can cause exception and the *catch* block handles the exception, if it occurs. Catch block contains the statements that have to be executed, when an exception occurs. The *finally* block always executes, irrespective of the fact whether or not an exception has occurred. The *finally* block is generally used to perform the cleanup process. If any exception occurs in the *try* block, the program control

directly transfers to its corresponding *catch* block and later to the *finally* block. If no exception occurs inside the *try* block, then the program control transfers directly to the *finally* block.

35. How can you prevent a class from overriding in C# and Visual Basic?

    You can prevent a class from overriding in C# by using the *sealed* keyword; whereas, the *NotInheritable* keyword is used to prevent a class from overriding in Visual Basic.

36. What are abstract classes? What are the distinct characteristics of an abstract class?

    An abstract class is a class that cannot be instantiated and is always used as a base class. The following are the characteristics of an abstract class:

    - You cannot instantiate an abstract class directly. This implies that you cannot create an object of the abstract class; it must be inherited.
    - You can have abstract as well as non-abstract members in an abstract class.
    - You must declare at least one abstract method in the abstract class.
    - An abstract class is always public.
    - An abstract class is declared using the *abstract* keyword.

    The basic purpose of an abstract class is to provide a common definition of the base class that multiple derived classes can share.

43. What do you mean by data encapsulation?

    Data encapsulation is a concept of binding data and code in single unit called object and hiding all the implementation details of a class from the user. It prevents unauthorized access of data and restricts the user to use the necessary data only.

44. What is the difference between procedural and object-oriented programming?

    Procedural programming is based upon the modular approach in which the larger programs are broken into procedures. Each procedure is a set of instructions that are executed one after another. On the other hand, OOP is based upon objects. An object consists of various elements, such as methods and variables.

    Access modifiers are not used in procedural programming, which implies that the entire data can be accessed freely anywhere in the program. In OOP, you can specify the scope of a particular data by using access modifiers - *public, private, internal, protected,* and *protected* internal.

45. Explain the concept of destructor?

A destructor is a special method for a class and is invoked automatically when an object is finally destroyed. The name of the destructor is also same as that of the class but is followed by a prefix tilde (~).

A destructor is used to free the dynamic allocated memory and release the resources. You can, however, implement a custom method that allows you to control object destruction by calling the destructor.

The main features of a destructor are as follows:

- Destructors do not have any return type
- Similar to constructors, destructors are also always public
- Destructors cannot be overloaded.

46. Can you declare a private class in a namespace?

    The classes in a namespace are *internal*, by default. However, you can explicitly declare them as *public* only and not as *private*, *protected*, or *protected internal*. The nested classes can be declared as *private*, *protected*, or *protected internal*.

47. A structure in C# can implement one or more interfaces. Is it true or false?

    Yes, it is true. Like classes, in C#, structures can implement one or more interfaces.

48. What is a static constructor?

    Static constructors are introduced with C# to initialize the static data of a class. CLR calls the static constructor before the first instance is created.

    The static constructor has the following features:

    - No access specifier is required to define it.
    - You cannot pass parameters in static constructor.
    - A class can have only one static constructor.
    - It can access only static members of the class.
    - It is invoked only once, when the program execution begins.

49. What are the different ways a method can be overloaded?

    The different ways to overload a method are given as follows:

    - By changing the number of parameters used
    - By changing the order of parameters
    - By using different data types for the parameters

50. Differentiate between an abstract class and an interface.

    Abstract Class:

1. A class can extend only one abstract class
2. The members of abstract class can be private as well as protected.
3. Abstract classes should have subclasses
4. Any class can extend an abstract class.
5. Methods in abstract class can be abstract as well as concrete.
6. There can be a constructor for abstract class.
7. The class extending the abstract class may or may not implement any of its method.
8. An abstract class can implement methods.

Interface

9. A class can implement several interfaces
10. An interface can only have public members.
11. Interfaces must have implementations by classes
12. Only an interface can extend another interface.
13. All methods in an interface should be abstract
14. Interface does not have constructor.
15. All methods of interface need to be implemented by a class implementing that interface.
16. Interfaces cannot contain body of any of its method.

51. What are queues and stacks?

Stacks refer to a list in which all items are accessed and processed on the Last-In-First-Out (LIFO) basis. In a stack, elements are inserted (push operation) and deleted (pop operation) from the same end called **top**.

Queues refer to a list in which insertion and deletion of an item is done on the First-In-First-Out (FIFO) basis. The items in a queue are inserted from the one end, called the **rear** end, and are deleted from the other end, called the **front** end of the queue.

52. Define an event.

Whenever an action takes place in a class, that class provides a notification to other classes or objects that are assigned to perform particular tasks. These notifications are called events. For example, when a button is clicked, the class generates an event called Click. An event can be declared with the help of the event keyword.

53. What are structures?

Structure is a heterogeneous collection of elements referenced by the same name. A structure is declared using the struct keyword. The following is an example that creates a structure to store an employee's information:

```
struct emp
{
```

```
 fixed int empID[15];
 fixed char name[30];
 fixed char addr[50];
 fixed char dept[15];
 fixed char desig[15];
}
```

The preceding example defines a structure *emp* and the members of this structure specify the information of an employee.

54. When do you really need to create an abstract class?

We define abstract classes when we define a template that needs to be followed by all the derived classes.