1. Explain OOPs
Object-Oriented Programming System (OOPs) is a programming concept that works on the principles of abstraction, encapsulation, inheritance,
and polymorphism. It allows users to create objects they want and create methods to handle those objects. The basic concept of OOPs is to
create objects, re-use them throughout the program, and manipulate these objects to get results.

==========

2. Explain an Abstraction, Give real life examples.
Abstraction is a process of hiding the implementation details and showing only functionality to the user.
Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text
and send the message. You don't know the internal processing about the message delivery.

==========

3. Explain an Encapsulation, Give real life examples.
Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several
medicines.

==========

4. Explain the relationship among Abstraction and Encapsulation.
Abstraction is the method of hiding the unwanted information. Whereas encapsulation is a method to hide the data in a single entity or unit
along with a method to protect information from outside. ... We can implement abstraction using abstract class and interfaces.
The major difference between abstraction and encapsulation is that abstraction hides the code complexity while encapsulation hides the
internal working from the outside world.

==========

5. Explain Polymorphism.
Polymorphism in Java is the ability of an object to take many forms. ... Any Java object that can pass more than one IS-A test is considered
to be polymorphic and in java, all the java objects are polymorphic as it has passed the IS-A test for their own type and for the class Object.

==========

6. Explain Inheritance.
Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important
part of OOPs (Object Oriented programming system).The idea behind inheritance in Java is that you can create new classes that are built
upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class.
Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationshi

p which is also
known as a parent-child relationship.

==========

7. How Composition is better than Inheritance?
Composition offers better test-ability of a class than Inheritance. If one class consists of another class, you can easily construct a Mock Object
representing a composed class for the sake of testing. This privilege is not given by inheritance. Although both Com position and Inheritance allow you
to reuse code, one of Inheritance's disadvantages is that it breaks encapsulation. If the subclass depends on the action of the superclass for its
function, it suddenly becomes fragile.
When super-class behavior changes, sub-class functionality can be broken without any modification on its part.
In the timeless classic Design Patterns, several object-oriented design patterns listed by Gang of Four: Elements of R eusable Object-Oriented Software,
favor Composition over Inheritance.
Strategy design pattern, where composition and delegation are used to modify the behavior of Context without touching context code, is a classical example of this. Instead of getting it by inheritance, because Context uses comp osition to carry strategy,
it is simple to have a new implementation of strategy at run-time.Another reason why composition is preferred over i nheritance is flexibility.
If you use Composition, you are flexible enough to replace the better and updated version of the Composed class im plementation.
One example is the use of the comparator class, which provides features for comparison.

==========

8. Which oops concept is used as reuse mechanism?
Through inheritance, we can acquire the properties of the parent class. Thus we can reuse the methods and variables of the parent class.
==========

9. Which oops concept exposes only the necessary information to the calling function?
Encapsulation is the oops concept exposes only the necessary information to the calling function.

==========

10. Explain a class and create a class.
In object-oriented programming, a class is a blueprint for creating objects (a particular data structure), providing initi al values for
state (member variables or attributes), and implementations of behavior (member functions or methods). The user-de fined objects are
created using the class keyword.

```
class Animals
{
 void display()
 {
  System.out.println("I like dogs");
 }
```

```
 public static void main(String args[])
 {
  Animals a = new Animals();
  a.display();
 }
}
```

==========

11. Using class, Write in brief abstraction and encapsulation

An Abstraction is a process of exposing all the necessary details and hiding the rest. In Java,
Data Abstraction is defined as the process of reducing the object to its essence so that only the
necessary characteristics are exposed to the users.

Abstraction defines an object in terms of its properties (attributes), behavior (methods),
and interfaces (means of communicating with other objects).
Here, you can see that an Owner is interested in details like Car description, service history, etc.
Garage Personnel are interested in details like License, work description, bill, owner, etc; and Registration Office int
erested
in details like vehicle identification number, current owner, license plate, etc.

```
abstract class Bike{
  abstract void run();
}
class Honda4 extends Bike{
void run(){System.out.println("running safely");}
public static void main(String args[]){
 Bike obj = new Honda4();
 obj.run();
}
}
```

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abs
traction.
Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods)
together as a single unit. In encapsulation, the variables of a class will be hidden from other classes,
and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.
To achieve encapsulation in Java −

Declare the variables of a class as private.

Provide public setter and getter methods to modify and view the variables values.

Benefits of Encapsulation:
The fields of a class can be made read-only or write-only.

A class can have total control over what is stored in its fields.

```
public class Student{
//private data member
private String name;
//getter method for name
public String getName(){
```

```
return name;
}
//setter method for name
public void setName(String name){
this.name=name
}
}
```

==========

12. Explain difference among class and object?
following are the differences among class and object:

class:
1)class is declared using class keyword e.g class Test{}.
2)class is a logical entity.
3)Class is a group of similar objects.
4)class is a blueprint from which objects are created
5)class is declared once.
6)class doesn't allocate memory when it is created

object:
1)object is created using new keyword mainly e.g Test t1= new Test();
2)object is physical entity.
3)Object is a real world entity such as pen, laptop, mobile, bed,
keyboard, mouse, chair
4)Object is an instance of a class.
5)Object is created many times as per requirement.
6)Object is created many times as per requirement.

==========

13. Define access modifiers?
Access modifiers are object-oriented programming that is used to set the accessibility
of classes, constructors, methods, and other members of Java.
Using the access modifiers we can set the scope or accessibility of
these classes, methods, constructors, and other members.
Four Types of Access Modifiers:
1)Private: We can access the private modifier only within the same class and not from outside the class.

2)Default: We can access the default modifier only within the same package and not from outside the package.
And also, if we do not specify any access modifier it will automatically consider it as default.

3)Protected: We can access the protected modifier within the same package and also from outside the package
with the help of the child class. If we do not make the child class, we cannot access it from outside the
package. So inheritance is a must for accessing it from outside the package.

4)Public: We can access the public modifier from anywhere. We can access public modifiers from within
the class as well as from outside the class and also within the package and from outside the package.

Let us see which all members of Java can be assigned with the access modifiers:

Members of JAVA	Private	Default	Protected	Public
Class	No	Yes	No	Yes
Variable	Yes	Yes	Yes	Yes

Method	Yes	Yes	Yes	Yes
Constructor	Yes	Yes	Yes	Yes
interface	No	Yes	No	Yes

==========

14. Explain an object? Create an object of above class.

A Java object is a member (also called an instance) of a Java class. Each object has an identity, a behavior and a state.

The state of an object is stored in fields (variables), while methods (functions) display the object's behavior. Objects are created at runtime from templates, which are also known as classes.

In Java, an object is created using the keyword "new".

```
public class Puppy {
  public Puppy(String name) {
    // This constructor has one parameter, name.
    System.out.println("Passed Name is :" + name );
  }

  public static void main(String []args) {
    // Following statement would create an object myPuppy
    Puppy myPuppy = new Puppy( "tommy" );
  }
}
```

==========

15. Give real life examples of object.
Ans-->Real-world objects share two characteristics: They all have state and behavior.
Dogs have state (name, color, breed, hungry) and behavior (barking, fetching, wagging tail).
Take the example of a car. When we describe a car as an object, there are a number of physical factors which contribute to define its state such as its brand name, colour, speed, size, number of seats, etc.
==========

16. Explain a Constructor.
Ans-->In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is cre ated.
At the time of calling constructor, memory for the object is allocated in the memory.
It is a special type of method which is used to initialize the object.
Every time an object is created using the new() keyword, at least one constructor is called.
It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a de fault constructor by default.

*Rules for creating Java constructor:

There are two rules defined for the constructor.

Constructor name must be the same as its class name

A Constructor must have no explicit return type
A Java constructor cannot be abstract, static, final, and synchronized

*Syntax of default constructor:
<class_name>(){}

==========

17. Define the various types of constructors?
Ans-->There are two types of constructors in Java:
1.no-arg constructor/Default constructor: A constructor is called "Default Constructor" when it doesn't have any parameter.

2.parameterized constructor: A constructor which has a specifc number of parameters is called a parameterized constructor.

==========

18. Whether static method can use nonstatic members?
Ans-->In static method, the method can only access only static data members and static methods of another class or same class
but cannot access non-static methods and variables.
Any method of a class which is not static is called non-static method or an instance method.
A static method is also called a class method and is common across the objects of the class and this method can be a ccessed using class name as well.

==========

19. Explain Destructor?
Ans--> A destructor is a member function that is invoked automatically when the object goes out of scope or is expli citly destroyed by a call to delete .
A destructor is the last function that is going to be called before an object is destroyed.
The thing is to be noted here, if the object is created by using new or the constructor uses new to allocate memory w hich resides in the heap memory or the free store, the destructor should use delete to free the memory.

Syntax:
~constructor-name();

==========

20. Explain an Inline function?
Ans-->inline function is powerful concept that is commonly used with classes. If a function is inline, the compiler pl aces a copy of the code of that function at each point where the function is called at compile time.
An inline function is one for which the compiler copies the code from the function definition directly into the code o f the calling function rather than creating a separate set of instructions in memory.
This eliminates call-linkage overhead and can expose significant optimization opportunities.

==========

21. Explain a virtual function?
A virtual function or virtual method in an OOP language is a function or method used to override

the behavior of the function in an inherited class with the same signature to achieve the polymorphism.
When the programmers switch the technology from C++ to Java,
they think about where is the virtual function in Java.
In C++, the virtual function is defined using the virtual keyword, but in Java,
it is achieved using different techniques.

==========

22. Explain a friend function?
A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class.
Even though the prototypes for friend functions appear in the class definition, friends are not member functions.
A friend can be a function, function template, or member function, or a class or class template,
in which case the entire class and all of its members are friends.

==========

23. Explain function overloading?
If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.
If we have to perform only one operation, having same name of the methods increases the readability of the program
.
Suppose you have to perform addition of the given numbers but there can be any number of arguments,
if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then
it may be difficult for you as well as other programmers to understand the behavior of the method because its name d iffers.
Advantage of method overloading:
Method overloading increases the readability of the program.
Different ways to overload the method:
There are two ways to overload the method in java-
1. By changing number of arguments
2. By changing the data type

==========

24. Explain a base class, sub class, super class?
The class which inherits the properties of other is known as subclass (derived class, child class)
The class whose properties are inherited is known as superclass (base class, parent class).
In Java, the superclass, also known as the parent class, is the class from which a child class (or a subclass) inherits its constructors, methods, and attributes.
For instance, in our above example, BankAccount was the superclass from which our subclass SavingsAccount inher ited its values.
The super keyword is similar to this keyword. Following are the scenarios where the super keyword is used.
-It is used to differentiate the members of superclass from the members of subclass, if they have same names.
-It is used to invoke the superclass constructor from subclass.

==========

25. Write in brief linking of base class, sub class and base object, sub object.
Difference between referencing using superclass reference and referencing using subclass reference is use superclass referencing can holds object of subclass and
could only access the methods which are defined/overridden by subclass while use subclass referencing can not hold object of superclass and
could access the methods of both superclass and subclass.
Object relation of Superclass (parent) to Subclass (child) exists while child to parent object relation never exists.
This means that reference of parent class could hold the child object while child reference could not hold the parent

object.
In case of overriding of non static method the runtime object would evaluate that which method would be executed o f subclass or of superclass.
While execution of static method depends on the type of reference that object holds.
Other basic rule of inheritance is related to static and non static method overriding that static method in java could n ot be overridden while non static method can be.
However subclass can define static method of same static method signature as superclass have hut that would not be consider as overriding while known as hiding of static method of superclass.

==========
==========s

Q26 Explain an abstract class?
Ans - An abstract class is a template definition of methods and variables of a class (category of object) that contains one or more abstracted methods. Abstract classes are used in all object-oriented programming (OOP) languages, incl uding Java. Objects or classes may be abstracted, which means that they are summarised into characteristics that are relevant to the current program's operation.

==========
==========

Q27 Explain operator overloading?
Ans - Operator overloading is a technique by which operators used in a programming language are implemented in u ser-defined types with customised logic that is based on the types of arguments passed.
Operator overloading facilitates the specification of user-defined implementation for operations wherein one or both operands are of user-defined class or structure type. This helps user-defined types to behave much like the fundamen tal primitive data types. Operator overloading is helpful in cases where the operators used for certain types provide s emantics related to the domain context and syntactic support as found in the programming language. It is used for sy ntactical convenience, readability and maintainability.
Java does not support operator overloading, except for string concatenation for which it overloads the + operator inte rnally.

==========
==========

Q28 Define different types of arguments? (Call by value/Call by reference)
Ans - Call by value method copies the value of an argument into the formal parameter of that function. Therefore, ch anges made to the parameter of the main function do not affect the argument.
In this parameter passing method, values of actual parameters are copied to function's formal parameters, and the pa rameters are stored in different memory locations. So any changes made inside functions are not reflected in actual p arameters of the caller.
Call by reference method copies the address of an argument into the formal parameter. In this method, the address is used to access the actual argument used in the function call. It means that changes made in the parameter alter the pa ssing argument.
In this method, the memory allocation is the same as the actual parameters. All the operation in the function are perf ormed on the value stored at the address of the actual parameter, and the modified value will be stored at the same ad dress.

==========
==========

Q29 Explain the super keyword?
Ans - The super keyword in Java is a reference variable which is used to refer immediate parent class object.
Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by su per reference variable.
Uses of super keyword
1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor

==========
Q30 Explain method overriding?
If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java
.
In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.
Usage of Java Method Overriding
◦ Method overriding is used to provide the specific implementation of a method which is already provided by its sup erclass.
◦ Method overriding is used for runtime polymorphism

Rules for Java Method Overriding
1. 1.The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance)
==========

31.Difference among overloading and overriding?

-Overloading happens at compile-time while Overriding happens at runtime: The binding of overloaded method call to its definition has happens at compile-time however binding of overridden method call to its definition happens at r untime.
-Static methods can be overloaded which means a class can have more than one static method of same name. Static methods cannot be overridden, even if you declare a same static method in child class it has nothing to do with the sa me method of parent class.
-The most basic difference is that overloading is being done in the same class while for overriding base and child cla sses are required. Overriding is all about giving a specific implementation to the inherited method of parent class.
-Static binding is being used for overloaded methods and dynamic binding is being used for overridden/overriding m ethods.
-Performance: Overloading gives better performance compared to overriding. The reason is that the binding of overri dden methods is being done at runtime.
-Private and final methods can be overloaded but they cannot be overridden. It means a class can have more than one private/final methods of same name but a child class cannot override the private/final methods of their base class.
-Return type of method does not matter in case of method overloading, it can be same or different. However in case of method overriding the overriding method can have more specific return type (refer this).
-Argument list should be different while doing method overloading. Argument list should be same in method Overri ding.

==========
32.Whether static method can use non-static members?

-In static method, the method can only access only static data members and static methods of another class or same c lass but cannot access non-static methods and variables.
==========

33.Explain a base class, sub class, super class?

Inheritance is an important pillar of OOP(Object-Oriented Programming). It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.
-Base Class: A base class is a class in Object-Oriented Programming language, from which other classes are derived.

The class which inherits the base class has all members of a base class as well as can also have some additional properties. The Base class members and member functions are inherited to Object of the derived class. A base class is also called parent class or superclass.
-Sub Class: A class that is created from an existing class. The derived class inherits all members and member functions of a base class. The derived class can have more functionality with respect to the Base class and can easily access the Base class. A Derived class is also called a child class or subclass.

==========

34.Write in brief linking of base class, sub class and base object, sub object.

-Object relation of Superclass (parent) to Subclass (child) exists while child to parent object relation never exists.This means that reference of parent class could hold the child object while child reference could not hold the parent object.
-In case of overriding of non static method the runtime object would evaluate that which method would be executed of subclass or of superclass.While execution of static method depends on the type of reference that object holds.
-Other basic rule of inheritance is related to static and non static method overriding that static method in java could not be overridden while non static method can be.However subclass can define static method of same static method signature as superclass have hut that would not be consider as overriding while known as hiding of static method of superclass.

==========

35.Explain an interface?

-Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body).
-Interfaces specify what a class must do and not how. It is the blueprint of the class.
An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So it specifies a set of methods that the class has to implement.
-If a class implements an interface and does not provide method bodies for all functions specified in the interface, then the class must be declared abstract.
-To declare an interface, use interface keyword. It is used to provide total abstraction. That means all the methods in an interface are declared with an empty body and are public and all fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface. To implement interface use implements keyword.
-Why And When To Use Interfaces?
1) To achieve security - hide certain details and only show the important details of an object (interface).
2) Java does not support "multiple inheritance" (a class can only inherit from one superclass). However, it can be achieved with interfaces, because the class can implement multiple interfaces.

==========

36. Explain exception handling?
Ans- An Exception is an unwanted event that interrupts the flow of the program.When any exception occurs program execution get
terminated.So we have to handle this error to succesfully run the program.So,Exception handling ensures that the flow of the
program doesn't break when an exception occurs.

==========

37. Explain the difference among structure and a class?
Ans- There are two categories of types, reference types and value types.Structure are value types and classes are reference types.
The general different is that a reference type lives on the heap, and a value type lives inline, that is, wherever it is yo

ur
variable or field is defined.Class can support inheritance but structure cannot support inheritance.

==========

38. Explain the default access modifier in a class?
Ans- The access level of a default modifier is only within the package. It cannot be accessed from outside the packag
e.
If you do not specify any access level, it will be the default.When we don't use any keyword explicitly, Java will set
a default
access to a given class, method or property.
==========

39. Explain a pure virtual function?
Ans- A pure virtual function is a virtual function that is required to be implemented by a derived class if the derived
class
is not abstract.Classes containing pure virtual methods are termed "abstract" and they cannot be instantiated directly.

==========

40. Explain dynamic or run time polymorphism?
Ans- Dynamic or Runtime polymorphism dispatch is a process in which a call to an overridden method is resolved at

runtime rather than compile-time.

==========

41. Do we require a parameter for constructors?
==> No, it is not compulsory.There are two types of constructors in Java: no-arg constructor, and parameterized con
structor.

==========

42. Explain static and dynamic binding?
==> When type of the object is determined at compiled time(by the compiler), it is known as static binding.If there i
s any private, final or static method in a class, there is static binding.
When type of the object is determined at run-time, it is known as dynamic binding.
Calling overloaded method/constructor are example of compile time binding.
Calling overriden method is an example of run time binding.

==========

43. How many instances can be created for an abstract class?
==> No you can't, instead you can create instance of all other classes extending that abstract class. Because it's abstr
act and an object is concrete. An abstract class is sort of like a template, or an empty/partially empty structure, you h
ave to extend it and build on it before you can use it.

==========

44. Explain the default access specifiers in a class definition?
==> The access level of a default modifier is only within the package. It cannot be accessed from outside the packag e. If you do not specify any access level, it will be the default.

==========

45. Which OOPS concept is used as reuse mechanism?
==> Inheritance is the feature of OOPS, which allows the users of OOPS to reuse the code which is already written. This OOPS feature inherits the features of another class in the programs. This mechanism actually inherits the fields and methods of the superclass.

==========

46. Define the Benefits of Object Oriented Programming?
Object Oriented Programming are
1. Re-usability
When we say re-usability, it means that "write once, use it multiple times" i.e., reusing some facilities rather tha n building it again and again, which can be achieved by using class. We can use it n number of times whenever r equired.
2. Data redundancy
It is one of the greatest advantages in oops. This is the condition which is created at the data storage when the sa me piece of data is held at two different places. If we want to use a similar functionality in multiple classes, we c an just write common class definitions for the similar functionalities by inheriting them.

3. Code maintenance
It is easy to modify or maintain existing code as new objects which can be created with small differences for the existing ones. It also helps users from doing rework many times. It is time saving as we modify the existing cod es incorporating new changes to it.

4. Security
Data hiding and abstraction are used to filter out limited exposure which means we are providing only necessary data to view as we maintain security.

5. Design benefits
The designers will have a longer and extensive design phase, which results in better designs. At a point of time when the program has reached critical limits, it will be easier to program all non oops one separately.

6. Easy troubleshooting
Using encapsulation objects are self-constrained. So, if developers face any problem easily it can be solved. And there will be no possibility of code duplicity.

==========

47. Explain method overloading?
- Overloading allows different methods to have the same name, but different signatures where the signature can di ffer by the number of input parameters
or type of input parameters or both.
- Overloading is related to compile-time (or static) polymorphism.
- Method overloading increases the readability of the program.
==========

48. Explain the difference among early binding and late binding?

Early Binding	Late Binding
1. It is a compile-time process	1. It is a run-time process
2. The method definition and method call are linked during the compile time.	2. The method definition and method call are linked during the run time.
3. Actual object is not used for binding.	3. Actual object is used for binding.
4. For example: Method overloading	4. For example: Method overriding
5. Program execution is faster	5. Program execution is slower

==========

49. Explain early binding? Give examples?

Early Binding :

When type of the object is determined at compiled time(by the compiler), it is known as static binding.

If there is any private, final or static method in a class, there is static binding.

- Example of static binding

```
class Dog
{
  private void eat(){System.out.println("dog is eating...");}
    public static void main(String args[])
    {
     Dog d1=new Dog();
     d1.eat();
    }
}
```

==========\

50. Explain loose coupling and tight coupling?

Tight coupling means classes and objects are dependent on one another. In general, tight coupling is usually not good because it reduces the flexibility and re-usability of the code while Loose coupling means reducing the dependencies of a class that uses the different class directly.

Tight Coupling

The tightly coupled object is an object that needs to know about other objects and is usually highly dependent on each other's interfaces.

Changing one object in a tightly coupled application often requires changes to a number of other objects.

In the small applications, we can easily identify the changes and there is less chance to miss anything. But in large applications, these inter-dependencies are not always known by every programmer and there is a chance of overlooking changes.

Loose Coupling

Loose coupling is a design goal to reduce the inter-dependencies between components of a system with the goal of reducing the risk that changes in one component will require changes in any other component.

Loose coupling is a much more generic concept intended to increase the flexibility of the system, make it more maintainable and makes the entire framework more stable.

==========

Que 51--->Give an example among tight coupling and loose coupling.

ANSWER--->TIGHT COUPLING: Tight coupling means the two classes often change together. In other words, if A knows more than it should about the way in which B was implemented, then A and B are tightly coupled.

EXAMPLE:Example : If you want to change the skin, you would also have to change the design of your body as wel l because the two are joined together – they are tightly coupled.
LOOSE COUPLING: In simple words, loose coupling means they are mostly independent. If the only knowledge th at class A has about class B, is what class B has exposed through its interface, then class A and class B are said to be loosely coupled.
EXAMPLE: f you change your shirt, then you are not forced to change your body – when you can do that, then you have loose coupling.

==========

Que 52: Write in brief abstract class.
ANSWER: A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abs tract and non-abstract methods (method with the body). It needs to be extended and its method implemented. It cann ot be instantiated.
It can have constructors and static methods also.It can have final methods which will force the subclass not to chang e the body of the method.

==========

Que 53.Define the Benefits of oops over pop?
ANSWER: Some of the benefits of object oriented programming(OOPS) over process oriented programming (POP) are as follows:
1)OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.

2)OOPs provides data hiding whereas in Procedure-oriented programming language a global data can be accessed fr om anywhere.

3)OOPs provides ability to simulate real-world problems much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.
==========

Que 54: Explain Generalization and Specialization?
ANSWER: GENERALIZATION : Converting a subclass type into a superclass type is called ‘Generalization‘ becau se we are making the subclass to become more general and its scope is widening. This is also called widening or up casting. Widening is safe because the classes will become more general.
SPECIALIZATION: Converting a super class type into a sub class type is called ‘Specialization‘. Here, we are comi ng down from more general form to a specific form and hence the scope is narrowed. Hence, this is called narrowing or down-casting.

==========

Que 55.Write in brief Association, Aggregation and Composition?
ASSOCIATION: Association refers to the relationship between multiple objects. It refers to how objects are related t o each other and how they are using each other's functionality. Composition and aggregation are two types of associa tion.
AGGREGATION: Aggregation is a weak association. An association is said to be aggregation if both Objects can e xist independently. For example, a Team object and a Player object. The team contains multiple players but a player can exist without a team.
COMPOSITION: The composition is the strong type of association. An association is said to composition if an Obje ct owns another object and another object cannot exist without the owner object. Consider the case of Human having a heart. Here Human object contains the heart and heart cannot exist without Human.
==========

56) Write in brief Object Composition vs. Inheritance.

Ans - Inheritance: a class may inherit - use by default - the fields and methods of its superclass. Inheritance is transitive, so a class may inherit from another class which inherits from another class, and so on, up to a base class (typically Object, possibly implicit/absent). Subclasses may override some methods and/or fields to alter the default behavior.

Composition: when a Field's type is a class, the field will hold a reference to another object, thus creating an association relationship between them.

Without getting into the nuances of the difference between simple association, aggregation, and composition, let's intuitively define composition as when the class uses another object to provide some or all of its functionality.

==========

57) Explain cohesion?

Ans - In object oriented design, cohesion refers all about how a single class is designed. Cohesion is the Object Oriented principle most closely associated with making sure that a class is designed with a single, well-focused purpose.

The more focused a class is, the cohesiveness of that class is more. The advantages of high cohesion is that such classes are much easier to maintain (and less frequently changed) than classes with low cohesion. Another benefit of high cohesion is that classes with a well-focused purpose tend to be more reusable than other classes.

==========

58) Explain "black-box-reuse" and "white-box-reuse"?

Ans - Black box reuse is using a class/function/code unmodified in a different project.

Black-Box reuse means that you use component without knowing it's internals. All you have is a component interface.

White box reuse is taking a class/function/code from one project and modifying it to suit the needs of another project.

White-box reuse means that you know how component is implemented. Usually White-box reuse means class inheritance.

==========

59) Explain "this"

Ans - The this keyword refers to the current object in a method or constructor.

The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter). If you omit the keyword in the example above, the output would be "0" instead of "5".

*this can also be used to:

current class constructor
Invoke current class method
Return the current class object
Pass an argument in the method call
Pass an argument in the constructor call

==========

60. Write in brief static member and member function.

Static Data Member (Class variable):
Static Data member has the following properties:
It is initialized by zero when first object of class is created.
Only one copy of static data member is created for the entire class and all object share the same copy.
Its scope is within class but its lifetime is entire program.
They are used to store the value that are common for all the objects.

Static variable can be declared using the following syntax:
static data-type variable-name;

The above syntax only declare the static variable. We can initialize static variable by using following syntax:
data-type class-name :: variable-name = initial-value ;

Along with the definition we can also initialize the static variable.
If static data members (variables) are declared under the public section than it can be accessed from outside the class and if it is declared in private section than it can only be accessed within the class itself.
Static variables are also known as class variables because they are not the variables of any particular class.
Let's consider the program given below to understand the static variable in which, we are creating static variable 'co unter' to count the number of objects created for the class 'MyClass'.

Static Member Function:
Member variables are known as instance variables in java. Instance variables are declared in a class, but outside a m ethod, constructor or any block. When space is allocated for an object in the heap, a slot for each instance variable va lue is created.
Static member function has following properties:
A static function can be access only by other static data member (variables) and function declared in the class.
A static function is called using class name instead of object name.
Consider the following program (Given in above section).
Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the ob ject is destroyed.
Instance variables hold values that must be referenced by more than one method, constructor or block, or essential pa rts of an object's state that must be present throughout the class.
Instance variables can be declared in a class level before or after use.
Access modifiers can be given for instance variables.
The instance variables are visible for all methods, constructors, and block in the class. Normally, it is recommended t o make these variables private (access level). However, visibility for subclasses can be given for these variables with the use of access modifiers.
Instance variables have default values. For numbers, the default value is 0, for Booleans it is false, and for object ref erences it is null. Values can be assigned during the declaration or within the constructor.
Instance variables can be accessed directly by calling the variable name inside the class. However, within static meth ods (when instance variables are given accessibility), they should be called using the fully qualified name. ObjectRef erence.VariableName.

==========

61. How will you relate unrelated classes or how will you achieve polymorphism without using base class?

=> We can implement the common interface in both different classes and can relate them by redefining the same met hod in those classes.

Or we can Declare the object of one class into another class as it's member and can relate those both classes by achi eving Association.
And we can achieve polymorphism without using base class by directly using method overloading,
which is the form of polymorphism ahieved in the same class by defining multiple methods with the same name, but with different parameters.
It is unrelated to either overriding or dynamic polymorphism.

==========

62. Explain Diamond problem?

=> Inheritance is a relation between two classes, the parent and child class.
The child class (sub-class) inherits all the properties of the parent class (super-class).
To define the relation, we use extends keyword.
When we inherit the properties of a class into another class, a copy of the super-class (parent class) is created in the sub-class (child class) object.
Hence, by using the sub-class object, we can access the member of super-class, also.
Diamond problem is related to the multiple inheritance.
Multiple Inheritance :
It is a feature of an object-oriented concept, where a class can inherit properties of more than one parent class.
The feature creates a problem when there exist methods with the same name and signature in both the super-class an d sub-class.
When we call the method, the compiler gets confused and cannot determine which class method to be called and ev en on calling which class method gets the priority.
It is an ambiguity that can rise as a consequence of allowing multiple inheritance.
It is a serious problem for other OPPs languages.
It is sometimes referred to as the deadly diamond of death.

==========

63. Explain the solution for diamond problem?
=> The solution to the diamond problem is default methods and interfaces.
We can achieve multiple inheritance by using these two things.
The default method is similar to the abstract method.
The only difference is that it is defined inside the interfaces with the default implementation.
We need not to override these methods. Because they are already implementing these interfaces.
The advantage of interfaces is that it can have the same default methods with the same name and signature in two di fferent interfaces.
It allows us to implement these two interfaces, from a class.
We must override the default methods explicitly with its interface name.
Therefore, we can achieve multiple inheritance by using the interface.
It is also the solution to the diamond problem.

==========

64. Explain the need of abstract class?

=> An abstract class can be used as a type of template for other classes.
The abstract class will hold common functionality for all classes that extend it.
When we have the requirement of a class that contains some common properties or methods with some common pro perties whose implementation is different for different classes,
in that situation, it's better to use Abstract Class.

We will use abstract class where we want to restrict the user from creating the object of parent class because by crea ting object of parent class, you can't call child class methods.
So, the developer has to restrict accidental creation of parent class object by defining it as abstract class.
So, I think, in these ways, we can use abstract class in our real time project.

==========

65. Why can’t we instantiate abstract class?

=> It's because an abstract class isn't complete. One or more parts of its public interface are not implemented.
You've said what they're going to be - what they're called, what their name is, what they return - but no implementat ion has been provided, so nobody would be able to call them.
Compilers prevent you from instantiating such incomplete objects on the basis that if you do instantiate an object yo u're going to want to use its entire public interface (this assumption is related to the idea that an object should do a m inimal set of things, so that the public interface is not too large to manage).
It's also a useful safety net.
Abstract classes represent such abstract concepts as vehicle.
Hence the idea of instantiating one is non-sensical because to actually instantiate it you need to know what you're in stantiating.

==========

66) Can abstract class have constructors?
Ans - When we create an object of any subclass all the constructors in the corresponding inheritance tree are invoked in the top to bottom approach.
The same case applies to abstract classes. Though we cannot create an object of an abstract class, when we cre ate an object of a class which
is concrete and subclass of the abstract class, the constructor of the abstract class is automatically invoked. He nce we can have a constructor
in abstract classes.

==========

67) How many instances can be created for an abstract class?
Ans - No you can't, instead you can create instance of all other classes extending that abstract class. Because it's abst ract and an object is concrete.
An abstract class is sort of like a template, or an empty/partially empty structure, you have to extend it and bui ld on it before you can use it.

==========

68) Which keyword can be used for overloading?
Ans - Overloaded operators are functions with special names: the keyword "operator" followed by the symbol for th e operator being defined. Like any
other function, an overloaded operator has a return type and a parameter list.

==========

69) Explain the default access specifiers in a class definition?
Ans - Default. When we don't use any keyword explicitly, Java will set a default access to a given class, method or p roperty. The default access modifier
is also called package-private, which means that all members are visible within the same package but aren't ac cessible from other packages: package com.

==========

70) Define all the operators that cannot be overloaded?

Ans - Not all C++ operators can be overloaded. For an operator to be overloaded, at least one of the operands must be a user-defined object. Only existing operators can be overloaded. You cannot overload new operators.

Operators that cannot be overloaded in C++

? “.” Member access or dot operator.
? “? : ” Ternary or conditional operator.
? “::” Scope resolution operator.
? “. *” Pointer to member operator.
? “ sizeof” The object size operator.
? “ typeid” Object type operator.

==========

71) Explain the difference among structure and a class?

Ans - A class is a user-defined blueprint or prototype from which objects are created. Basically, a class combines the fields and methods(member function which defines actions) into a single unit.

A structure is a collection of variables of different data types under a single unit. It is almost similar to a class because both are user-defined data types and both hold a bunch of different data types.

==========

72) Explain the default access modifier in a class?

Ans - If you don't use any modifier, it is treated as default by default. The default modifier is accessible only within package. It cannot be accessed from outside the package.

It provides more accessibility than private. But, it is more restrictive than protected, and public.

Default access modifier means we do not explicitly declare an access modifier for a class, field, method, etc. A variable or method declared without any access control modifier is available to any other class in the same package.

==========

73) Can you list out the different types of constructors?

Ans - A constructor is a special type of function with no return type. Name of constructor should be same as the name of the class. We define a method inside the class and constructor is also defined inside a class. A constructor is called automatically when we create an object of a class. We can’t call a constructor explicitly.

Constructor Types

Default Constructor
Parameterized Constructor
Copy Constructor
Static Constructor
Private Constructor

==========

74) Explain a friend function?
Ans - A friend function is a function that is specified outside a class but has the ability to access the class members' p rotected and private data. A friend can be a member's function,
function template, or function, or a class or class template, in which case the entire class and all of its member s are friends.
A friend function in C++ is a function that is preceded by the keyword “friend”. When the function is declared as a friend, then it can access the private and protected data
members of the class. A friend function is declared inside the class with a friend keyword preceding as shown below.

==========

75) Explain a ternary operator?
Ans - The ternary operator is an operator that exists in some programming languages, which takes three operands rat her than the typical one or two that most operators use. It provides
a way to shorten a simple if else block. With this type of comparison, you can shorten the code using the terna ry operator.
operands: a condition followed by a question mark (?), then an expression to execute if the condition is truthy followed by a colon (:), and finally the expression to execute if the condition is falsy.

==========

76. Do We Require Parameter For Constructors?
Ans-->No-argument constructor: A constructor that has no parameter is known as the default constructor.
If we don't define a constructor in a class, then the compiler creates default constructor(with no arguments) for the cl ass.
Default constructor provides the default values to the object like 0, null, etc.
It is called constructor because it constructs the values at the time of object creation.
It is not necessary to write a constructor for a class.
It is because java compiler creates a default constructor if your class doesn't have any.

==========

77. Explain Sealed Modifiers?
Ans-->When applied to a class, the sealed modifier prevents other classes from inheriting from it.
In the following example, class B inherits from class A , but no class can inherit from class B.

==========

78. Explain The Difference Among New And Override?
Ans-->virtual keyword must be defined to override the method. The method using override keyword that regardless of reference type(reference of base class or derived class) if it is instantiated with base class, the method of base clas s runs. Otherwise, the method of derived class runs.
new: if the keyword is used by a method, unlike override keyword, the reference type is important.
If it is instantiated with derived class and the reference type is base class, the method of base class runs.
If it is instantiated with derived class and the reference type is derived class, the method of derived class runs.
Namely, it is contrast of override keyword. En passant, if you forget or omit to add new keyword to the method, the compiler behaves by default as new keyword is used.

==========

79. How Can We Call The Base Method Without Creating An Instance?
Ans-->Its possible If its a static method.
Its possible by inheriting from that class also.
Its possible from derived classes using base keyword.

==========

80. Define The Various Types Of Constructors?
-->There are two types of constructors in Java:
1.no-arg constructor/Default constructor: A constructor is called "Default Constructor" when it doesn't have any parameter.

2.parameterized constructor: A constructor which has a specifc number of parameters is called a parameterized constructor.
==========

81. Define Manipulators?
==> Manipulators are functions specifically designed to be used in conjunction with the insertion (<<) and extractio n (>>) operators on stream objects.
==========

82. Can you give some examples of tokens?
==> The Java compiler breaks the line of code into text (words) is called Java tokens. These are the smallest element of the Java program. The Java compiler identified these words as tokens. These tokens are separated by the delimite rs. It is useful for compilers to detect errors. Remember that the delimiters are not part of the Java tokens.
token Eg: identifier, keyword, separator, operator, literal, comment

==========
83. Explain structured programming and its disadvantage?
==> In structured programming, we sub-divide the whole program into small modules so that the program becomes easy to understand. The purpose of structured programming is to linearize control flow through a computer program so that the execution sequence follows the sequence in which the code is written.
The following are the disadvantages of structured programming:
1. A high level language has to be translated into the machine language by translator and thus a price in computer time is paid.
2. The object code generated by a translator might be inefficient compared to an equivalent assembly language pro gram.
3. Data type are proceeds in many functions in a structured program. When changes occur in those data types, the corresponding change must be made to every location that acts on those data types within the program. This is really a very time consuming task if the program is very large.
4. Let us consider the case of software development in which several programmers work as a team on an applicati on. In a structured program, each programmer is assigned to build a specific set of functions and data types. Since di fferent programmers handle separate functions that have mutually shared data type. Other programmers in the team must reflect the changes in data types done by the programmer in data type handled. Otherwise, it requires rewriting several functions.

==========
84. Explain the advantage of C++ being a block-structured language?
==> C++ is a highly portable language and is often the language of selection for multi-device, multi-platform app de

velopment.
C++ is an object-oriented programming language and includes concepts like classes, inheritance, polymorphism, d ata abstraction, and encapsulation which allow code reusability and makes programs very maintainable.
C++ use multi-paradigm programming. The Paradigm means the style of programming .paradigm concerned abou t logics, structure, and procedure of the program. C++ is multi-paradigm means it follows three paradigm Generic, I mperative, Object Oriented.
It is useful for the low-level programming language and very efficient for general purpose.

==========

85. Can Struct be inherited?
==> A struct cannot inherit from another kind of struct.

==========

86.When to use interface over abstract class.

-You should use an interface if you want a contract on some behavior or functionality. You should not use an interfa ce if you need to write the same code for the interface methods. In this case, you should use an abstract class, define the method once, and reuse it as needed.

==========

87.Explain a private constructor? Where will you use it?

-In Java, the constructor is a special type of method that has the same name as the class name. Internally, a construct or is always called when we create an object of the class. It is used to initialize the state of an object.
-In the same way, Java also allows us to create a private constructor.
-A private constructor in Java is used in restricting object creation. It is a special instance constructor used in static member-only classes. If a constructor is declared as private, then its objects are only accessible from within the decla red class. You cannot access its objects from outside the constructor class

==========

88.Can you override private virtual methods?

-No, we cannot override private or static methods in Java.
-Private methods in Java are not visible to any other class which limits their scope to the class in which they are decl ared.

==========

89.Can you allow class to be inherited, but prevent from being over-ridden?
-Except for the Object class, a class has exactly one direct superclass. You can prevent a class from being subclassed by using the final keyword in the class's declaration.
-Similarly, you can prevent a method from being overridden by subclasses by declaring it as a final method.

==========

90.Why can’t you specify accessibility modifiers for methods inside interface?
-Interface members are always public because the purpose of an interface is to enable other types to access a class or struct. No access modifiers can be applied to interface members. All the interface methods are Public. You can't cre ate an access modifier in interface.

==========

91. Can static members use non static members? Give reasons.

No. In static method, the method can access only static data members and static methods of another class or same class but cannot access non-static methods and variables. It cannot access non-static data (instance variables).

==========

92. Define different ways a method can be overloaded?

Overloaded methods are differentiated based on the number and type of the parameters passed as an argument to the methods. You can not define more than one method with the same name, Order and the type of the arguments. It would be a compiler error. The compiler does not consider the return type while differentiating the overloaded method. But you cannot declare two methods with the same signature and different return type. It will throw a compile-time error. Method overloading can be done by changing:

1. The number of parameters in two methods.
2. The data types of the parameters of methods.
3. The Order of the parameters of methods.
4. If both methods have the same parameter types, but different return type, then it is not possible.

==========

93. Can we have an abstract class without having any abstract method?

Yes we can have an abstract class without Abstract Methods as both are independent concepts. Declaring a class abstract means that it can not be instantiated on its own and can only be sub classed. Declaring a method abstract means that Method will be defined in the subclass. An abstract class means that hiding the implementation and showing the function definition to the user. An abstract class having both abstract methods and non-abstract methods. For an abstract class, we are not able to create an object directly. But Indirectly we can create an object using the subclass object. A Java abstract class can have instance methods that implement a default behavior. An abstract class can extend only one class or one abstract class at a time. Declaring a class as abstract with no abstract methods means that we don't allow it to be instantiated on its own. An abstract class used in Java signifies that we can't create an object of the class directly.

==========

94. Explain the default access modifier of a class?

The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default. If you don't use any modifier, it is treated as default by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But, it is more restrictive than protected, and public. When we don't use any keyword explicitly, Java will set a default access to a given class, method or property. The default access modifier is also called package-private, which means that all members are visible within the same package but aren't accessible from other packages: package com.

==========

95. Can function overriding be explained in same class?

While overriding, Both methods should be in two different classes and, these classes must be in an inheritance relation. Both methods must have the same name, same parameters and, same return type else they both will be treated as different methods. The method in the child class must not have higher access restrictions than the one in the superclass. If you try to do so it raises a compile-time exception. If the super-class method throws certain exceptions, the method in the sub-class should throw the same exception or its subtype (can leave without throwing any exception). Therefore, you cannot override two methods that exist in the same class, you can just overload them.

==========

96. Does function overloading depends on Return Type?

No, you cannot overload a method based on different return type but same argument type and number in java. In overloading it is must that the both methods have same name & different parameters (different type or, different number or both). The return type doesn't matter. If they don't have different parameters, they both are still considered as same method and a compile time error will be generated.

==========

97. Define different ways to declare an array?

You can declare an array in different methods, some are mentioned below:

```
int[] myIntArray = new int[3];
int[] myIntArray = {1, 2, 3};
int[] myIntArray = new int[]{1, 2, 3};
int num[] = {1, 2, 3, 4, 5};
int[] num = {1,2,3,4,5};
int num[] = new int[5];
int[] num = new int[5];
String[] myStringArray;
myStringArray = new String[]{"a", "b", "c"};
```

==========

98. Can abstract class have a constructor?

Yes, an Abstract class always has a constructor. If you do not define your own constructor, the compiler will give a default constructor to the Abstract class. Constructor Chaining is a concept when a constructor calls another constructor in the same class or its base class. We know that Abstract classes can have constructors and we can not instantiate an abstract class. We can call the Abstract class's constructor through constructor chaining.

==========

99. Define rules of Function overloading and function overriding?

Some rules of FUNCTION OVERLOADING are as mentioned below:
Two methods will be treated as overloaded if both follow the mandatory rules below:
Both must have the same method name.
Both must have different argument lists.

And if both methods follow the above mandatory rules, then they may or may not:
Have different return types.
Have different access modifiers.
Throw different checked or unchecked exceptions.

Some rules of FUNCTION OVERRIDING are as mentioned below:
With respect to the method it overrides, the overriding method must follow following mandatory rules:
It must have the same method name.
It must have the same arguments.
It must have the same return type. From Java 5 onward, the return type can also be a subclass (subclasses are a covariant type to their parents).
It must not have a more restrictive access modifier (if parent --> protected then child --> private is not allowed).

It must not throw new or broader checked exceptions.

And if both overriding methods follow the above mandatory rules, then they:
May have a less restrictive access modifier (if parent --> protected then child --> public is allowed).
May throw fewer or narrower checked exceptions or any unchecked exception.

Apart from the above rules, there are also some facts to keep in mind:
Only inherited methods can be overridden. That means methods can be overridden only in child classes.
Constructors and private methods are not inherited, so they cannot be overridden.
Abstract methods must be overridden by the first concrete (non-abstract) subclass.
final methods cannot be overridden.
A subclass can use super.overridden_method() to call the superclass version of an overridden method.

==========

100. Explain the concept of Pure Virtual Functions?

A pure virtual function or pure virtual method is a virtual function that is required to be implemented by a derived class if the derived class is not abstract. Classes containing pure virtual methods are termed "abstract" and they cannot be instantiated directly. A subclass of an abstract class can only be instantiated directly if all inherited pure virtual methods have been implemented by that class or a parent class. Pure virtual methods typically have a declaration (signature) and no definition (implementation). Although pure virtual methods typically have no implementation in the class that declares them, pure virtual methods in some languages (e.g. C++ and Python) are permitted to contain an implementation in their declaring class, providing fallback or default behaviour that a derived class can delegate to, if appropriate Pure virtual functions can also be used where the method declarations are being used to define an interface - similar to what the interface keyword in Java explicitly specifies. In such a use, derived classes will supply all implementations.