Interview Questions.

Q1 – 4).

Page No.:

Q.1) Explains OOPs

Ans) Object oriented programming (OOPs) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. It has basically 4 principles as it follows

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

Q2) Explain an Abstraction & Real life example.

Ans: Hiding internal details and showing functionality is known as Abstraction. For example, phone, it has lot of functionalities, but we dont the internal processing of it.

Q3 Explain encapsulation & Real life example

Ans. Encapsulation is one of the best Java OOPs concept of wrapping data and code. In this OOPs concept, the variable of class are also hidden from other class. It can be only accessed using the methods of their current class. For example -) in school, a student cannot exist without class

Q4) Explain relationship among abstraction and encapsulatio

Ans Data abstraction can be achieved by using encapsulation we can hide operation and structure of actual

Q5 – 7).

① Topic : ..... Date : ..... Page No. :

program from the user and can show only required information by the user. For example in class student only essential like roll no, Student - name, course etc will be visiable. The secret info like calculation of grades, etc will be hidden.

Q5) What is inheritance

Ans Inheritance is one of the basic concepts of OOPs in which one object acquire the properties and behaviours of the parent Object. It's like creating parent - child relationship between two classes. It offers robust and natural mechanism for organizing and structure of any software

Q.6) What is polymorphism?

Ans Polymorphism refers to the one of OOPs concept in java which is the ability of variable, object or function to taken on multiple forms. For example, in English, the verb run has different meaning if we use it with laptops, a footrace and business

Q.7) How composition is better than Inheritance

Ans. The fact that Java doesn't support multiple inheritance is one of the reason for favouring composition over inheritance in Java. Since you

Q7 – 9).

Topic : ..... Date : .....
Page No.: .....

can extend only one class in Java, but if we need multiple features, such as reading and writing data into the file, you need reader - writer functionality. It makes your job simple To have them as privivate members, and this is called composition

ii) Although both Composition and ~~Abstraction~~ (Inheritance) allow To you reuse code, one of disadvantages of the Inheritance is that it breaks encapsulation If the subclass depends on the action of the super class for it's function, it suddenly becomes more fragile.

iii) Another reason why composition is preferred over inheritance is it's flexibility. If you use composition, you are flexiable enough To updated version of composed class implementation. One of it's example is comparator class.

Q.8) Which OOPS concept is used as reuse mechanism

Ans. Inheritance is the feature that provides a reuse mechanism. This mechanism provides reusibility To the user

Q.9 Which OOPS concept exposes only necessary information To calling function

Ans. Data hiding is the concept of OOPS which means exposing only necessary info To clients.

Q11.Using above class ,Write brief about abstraction and encapsulation?

Ans:

Abstraction:Abstraction is the process of hiding the internals details of application from the outer world

Abstraction is used to describe things in simple terms.Its used to create boundary between the application and

the clients programs.

Encapsulation:Encapsulation is the process by which data and code that acts upon them(methods) are integrated as single unit.By encapsulating class's variables ,other class cannot access them,and only the methods of the class can access them.

Q13.Define access modifiers ?

Ans:There are two types of modifiers in Java: access modifiers and non-access modifiers.

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:

Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

Default: 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.

Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

There are many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc. Here, we are going to learn the access modifiers only.

Q10-12).

Q) What is Class ? Give an example and create a class.

Ans. A class is representation of a Type of object. It is the blueprint / plan / Template that describe the details of an object. For example a Student is an object which have different attributes. A class in Java contain methods, constructors, blocks and nested class and interface.

```
class Student
{
    int rollno;
    String name;
    double phoneno;
```

Q.12 Explain difference between Classes and objects.

Class	Object
1) Class is the blueprint or Template from which objects are created	1) Object is an instance of a class
2) Object is real world entity such as pen, laptop, mobile bed etc	2) Class is the group of similar objects.
3) Object is the physical entity	3) Class is the logical entity.
4) Object is created many times as per requirement	4) Class is declared once
5) Object allocates memory when it is created	5) Class doesn't allocated memory when it is created.

Q15 – 17).

Topic : .....

Q.15) Give real life examples of object

Ans. Class -) Human Being
Object -) Man, Women are object of Class Human Being.

ii) Class -) Colours.
Red, Blue -) These colours are an object of Class Colours.

Q.) Explain Constructors

Ans. i) It is used To initialize the object's data at the Time of object creation.

ii) It is special method class whose name is similar To class name.

iii) It constructs the value i.e. provide data for the object. That is why it is known as Constructors.

iv) Constructor must have no explicit return Type and it cannot be static, abstract, final and synchronized.

Q) Define various Types of Constructors.

Ans. There are mainly two Types of Constructors in Java

1) Default Constructor (no arg constructor)
A constructor is called defaulter constructor when it doesn't have any parameter.
e.x. class Student

2022/11/11 19:54

Q17 – 18).

Topic : ..... Date : .....
Page No.: .....

* The Defaultor constructor is used To provide The default values to the object like, 0, null etc.
* If There is no constructor in class, compiler automatically creates default constructor.

```
class Bike                                  class Bike
{                  -> (Compiler) ->         {
}                                             Bike() { }
                                            }
```

ii) Parameterized Constructor.
A constructor which has specific number of parameters is called parameterized constructor. The parameterized constructor is used To provide different values to distinct objects.

ex.
```
class Student
{
    int id;
    String name;
    Student(int n, String name) // 2 arg constructor
    {                           // Parameterized constructor
        name = a;
        id = n;
    }
```

Q.) Whether static method can use non-static members?
Ans Inside static method only static data can be accessed or used, instance variable cannot be accessed hence static method cannot use non-static members.

Q19 – 22).

Topic : ..... Page No. .....

Q) Explain destructor?

Ans A destoructor is a member function that is invoked automatically when object goes out of scope or is explicitly destroyed by a call To delete. A destructor has the same name as the class, preceded by tilde (~) e.x. Class Stud
:~ student( )

Q) What is Inline function?

Ans An Inline function is one for which the compiler copies the code from the function defination directly into the code of calling function rather can creating sperate set of instructions in the memory.

Q) Explain Virtual function?

Ans. A virtual function is member function which is declared within base class and is re-defined (overridden) by derived class. They are mainly used To achieve Run-Time polymorphism

Q) Explain friend function

Ans. A friend function is defined as function that can access private, protected and public members of class. The friend function is declared using friend keyword

```
class class Named
    friend returnType (arg);
```

2022/11/11 19:

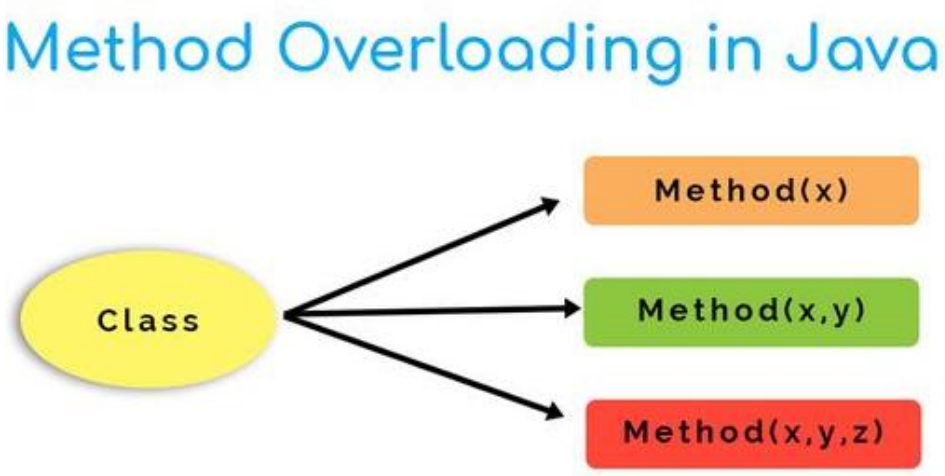
Q23. Explain function overloading?

Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters, or a mixture of both.

Method overloading is also known as Compile time Polymorphism**,** or Early binding Java. In Method overloading compared to parent argument, child argument will get the highest priority.

Different ways of Method Overloading in java:

- Changing the Number of Parameters.
- Changing Data Types of the Arguments.
- Changing the Order of the Parameters of Methods

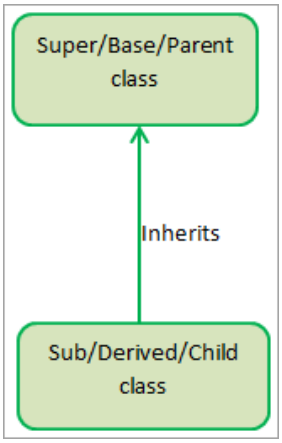


Q24. Explain a base class, sub class, super class?

Base Class: Base Class is the class from where a subclass inherits the features. It is also called a base class or a parent class.

Sub Class: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

Super Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.



26. Explain an abstract class?

→A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Points to Remember

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method

27. Explain operator overloading?

→ In Java, two or more methods may have the same name if they differ in parameters (different number of parameters, different types of parameters, or both). These methods are called overloaded methods and this feature is called method overloading. For example:

void func() { ... }

void func(int a) { ... }

float func(double a) { ... }

float func(int a, float b) { ... }

Here, the func() method is overloaded. These methods have the same name but accept different arguments.

28. Define different types of arguments? (Call by value/Call by reference)

→Call By Value:

While calling a function, we pass values of variables to it. Such functions are known as “Call By Values”.

In this method, the value of each variable in calling function is copied into corresponding dummy variables of the called function.

Call By Reference:

While calling a function, instead of passing the values of variables, we pass address of variables(location of variables) to the function known as “Call By References.

In this method, the address of actual variables in the calling function are copied into the dummy variables of the called function.

29. Explain the super keyword?

→ 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 super reference variable.

Usage of Java super Keyword

super can be used to refer immediate parent class instance variable.

super can be used to invoke immediate parent class method.

super() can be used to invoke immediate parent class constructor.

30. 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 superclass.
- Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

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?

→

No.	Method Overloading	Method Overriding
1)	Method overloading is used *to increase the readability* of the program.	Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class.
2)	Method overloading is performed *within class*.	Method overriding occurs *in two classes* that have IS-A (inheritance) relationship.
3)	In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
4)	Method overloading is the example of *compile time polymorphism*.	Method overriding is the example of *run time polymorphism*
5)	In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter.	*Return type must be same or covariant* in method overriding.

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

→A static method can only access static data members and static methods of another class or same class but **cannot access non-static methods and variables**

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

- →**Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- Base Class: In an object-oriented programming language, a base class is an existing class from which the other classes are determined and properties are inherited.

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

→ There are two approaches to refer a subclass object. Both have some advantages/disadvantages over the other. The declaration affect is seen on methods that are visible at compile-time.

1)First approach (Referencing using Superclass reference): A reference variable of a superclass can be used to a refer any subclass object derived from that superclass. If the methods are present in SuperClass, but overridden by SubClass, it will be the overridden method that will be executed.

2)Second approach (Referencing using subclass reference) : A subclass reference can be used to refer its object.

35. Explain an interface?

→ An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Java Interface also **represents the IS-A relationship**.

It cannot be instantiated just like the abstract class.

Since Java 8, we can have **default and static methods** in an interface.

Since Java 9, we can have **private methods** in an interface.

Q 36 . Explain exception handling?

- An Exception is an unexpected event that interrupts the normal flow of the program. When an exception occurs

program execution gets terminated.

- Exceptions can be checked or unchecked and Can occur at compile time or runtime.

- Exception handling is to design the program in such a way that even if there is an exception, all operations

are performed then only the program should be terminated.This is called exception handling.

- Following five keywords are used to handle the exceptions:

* try - Defines a block of code in which exceptions may occur

* catch - Catches this exception and handle it in some rational manner

* throw - Used to explicitly throw an exception object

* throws - Used in the signature of method to indicate that this method might throw one of the listed type exceptions.

* finally - A block that is used to execute important code such as closing connection, stream etc. This block is guaranteed to execute whether any exception is generated or not.

Q 37 . Explain the difference among structure and a class?

(i) Class :

- It is defined using ‘class’ keyword.

- It is possible that some objects may have similar properties and actions. Such objects belong to same category called a ‘class’.

- It is only a logical component and not the physical entity e.g. If we have class of

“Expensive Cars” it could have objects like Mercedes, BMW, Toyota, etc.

- They can have constructors and destructors.

- It can use inheritance to inherit properties from base class.

- The ‘protected’ access modifier can be used with the data members defined inside the class.

(ii) Structure :
- The ‘struct’ keyword is used to define a structure.
- When the value of one data member is changed, it doesn’t affect other data members in structure.
- It helps to initialize multiple members at once.
- Total size of the structure is equivalent to the sum of the size of every data member.
- It is used to store various data types.
- It takes memory for every member which is present within the structure.
- It doesn’t support inheritance.
- It doesn’t have a constructor or destructor.

Q 38. Explain the default access modifier in a class?

- As the name suggests access modifiers in Java helps to restrict the scope of a class, constructor, variable, method, or data member.

- There are four types of access modifiers available in java :

(a) Default – (No keyword required) 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.

(b) Private - The access level of a private modifier is only within the class.
It cannot be accessed from outside the class.

(c) Protected - The access level of a protected modifier is within the package and outside the package through child class.
If you do not make the child class, it cannot be accessed from outside the package.

(d) Public - The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

- There are also many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc.

Q 39 . Explain a pure virtual function?

- A virtual function for which we are not required implementation is considered as Pure virtual function.
  For example, Abstract method in Java is a pure virtual function.
- A pure virtual function is a "do nothing" function. Here "do nothing" means that it just provides the template,
  and derived class implements the function.
- It can be considered as an empty function means that the pure virtual function does not have any definition relative to the base class.
- Programmers need to redefine the pure virtual function in the derived class as it has no definition in the base class.
- A class having pure virtual function cannot be used to create direct objects of its own. It means that the class is containing
  any pure virtual function then we cannot create the object of that class. This type of class is known as an abstract class.

Q40. Explain dynamic or run time polymorphism?

- The word polymorphism is a combination of two words i.e. ploy and morphs.
- The word poly means many and morphs means different forms.
- Dynamic polymorphism is a process or mechanism in which a call to an overridden method is resolved at runtime
  rather than compile-time. It is also known as runtime polymorphism or dynamic method dispatch.
  We can achieve dynamic polymorphism by using the method overriding.

- Properties of Dynamic Polymorphism :
  > It decides which method is to execute at runtime.
  > It can be achieved through dynamic binding.
  > It happens between different classes.
  > Inheritance is involved in dynamic polymorphism.

Q41 . Do we require a parameter for constructors?

- A constructor is a method that is called at runtime during the object creation by using the new operator.

The JVM calls it automatically when we create an object. When we do not define a constructor in the class,

the default constructor is always invisibly present in the class.

- we use the constructor to initialize the instance variable of the class.

- The parameterized constructor is used if we want to dynamically initialize the instance variables with the specified values

at the time of instantiation.

- A constructor does not have to use everything that is passed to it, but the purpose of a constructor is to "set up" your object,

so not using a parameter seems pointless.

Q42 . Explain static and dynamic binding?

- The static binding happens at the compile-time, and dynamic binding happens at the runtime.

- Hence, they are also called early and late binding, respectively. In static binding, the function definition and the function call

are linked during the compile-time, whereas in dynamic binding, the function calls are not resolved until runtime.

Q43. How many instances can be created for an abstract class?

- The answer to the question of how many instances of an abstract class can be created is zero.
- That is, we cannot create an instance of an abstract class as it does not have any complete implementation.
- An abstract class acts like a template or an empty structure.
- Instead you can create instance of all other classes extending that abstract class.

Q44 . Explain the default access specifiers in a class definition?

- When no access modifier is specified for a class, method, or data member – It is said to be having the default access modifier by default.
- The data members, class or methods which are not declared using any access modifiers i.e. having default access modifier
  are accessible only within the same package.
- No keyword is required here.

Q45. Which OOPS concept is used as reuse mechanism?

- Inheritance is the concept that can be used as resuse mechanism.

- when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our

  new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

- The class whose features are inherited is known as superclass (also known as base or parent class).

- The class that inherits the other class is known as subclass (also known as derived or extended or child class).

- The subclass can add its own fields and methods in addition to the superclass fields and methods.

Q46. Define the Benefits of Object Oriented Programming?

Answer = >

1. Troubleshooting is easier with the OOP language
2. Code Reusability
3. Productivity
4. Data Redundancy
5. Code Flexibility
6. Solving problems
7. Security

1. Troubleshooting is easier with the OOP language

Suppose the user has no idea where the bug lies if there is an error within the code. Also, the user has no idea where to look into the code to fix the error. This is quite difficult for standard programming languages. However, when Object-Oriented Programming is applied, the user knows exactly where to look into the code whenever there is an error. There is no need to check other code sections as the error will show where the trouble lies.

Let's understand it by this example, your mobile you are using your phone and suddenly the screen turns white and then turned off. You infer that the reason behind this must be a problem with the motherboard of your phone. Encapsulation is the reason behind this, there is no need to go after every code in order to understand a problem.

It is mainly encapsulation that makes the objects are self-contained. This further helps in troubleshooting and easier collaborative development.

2. Code Reusability

One of two important concepts that are provided by Object-Oriented Programming is the concept of inheritance. Through inheritance, the same attributes of a class are not required to be written repeatedly. This avoids the issues where the same code has still to be written multiple times in a code. With the introduction of the concept of classes, the code section can be used as many times as required in the program. Through the inheritance approach, a child class is created that inherits the fields and methods of the parent class. The methods and values that are present in the parent class can be easily overridden. Through inheritance, the features of one class can be inherited by another class by

extending the class. Therefore, inheritance is vital for providing code reusability and also multilevel inheritance.
One of the benefits of oop in c++ is the productivity of a code. For example, under the pen class object, one person needs a blue pen class, and the other wants a black pen class. But despite these pen class objects being of different types but they do share a commonality. The commonality for them is that they both are different colour types of pens. This is one of the biggest advantages of oop in c++. Thus, Object-Oriented Programming offers the feature of class reusability where the class that is once created can be used again. In doing so, time is saved, and the need for extra coding is eliminated as similar features can be inherited.

3. Productivity

The productivity of two codes increases through the use of Object-Oriented Programming. This is because the OOP has provided so many libraries that new programs have become more accessible. Also, as it provides the facility of code reusability, the length of a code is decreased, further enhancing the faster development of newer codes and programs.
One of the advantages of using OOP in C++ is the productivity of a code as it not only saves time but also the possibility of errors. When a code is being put into a library, the steps for manual programming are not required. The codes can be easily accessed in these libraries.

4. Data Redundancy

By the term data redundancy, it means that the data is repeated twice. This means that the same data is present more than one time. In Object-Oriented Deprogramming the data redundancy is considered to be an advantage. For example, the user wants to have a functionality that is similar to almost all the classes. In such cases, the user can create classes with similar functionaries and inherit them wherever required.
Although redundancy by the term is not appealing here it is considered as one of the advantages of object oriented programming, the main reason is that it reduces the repetition of a mundane task. If some data is required to be used again then the data from a similar functionary can be utilised. And the efforts can go into doing those tasks which require more attention.

5. Code Flexibility

The flexibility is offered through the concept of Polymorphism. A scenario can be considered for a better understanding of the concept. A person can behave differently whenever the surroundings change. For example, if the person is in a market, the person will behave like a customer, or the behavior might get changed to a student when the person is in a school or any institution.
In this example, it can be observed that different behaviors are shown by the same person whenever the surroundings around the person get changed. This could explain the concept of Polymorphism and its flexibility. The developers benefit through Polymorphism in the following ways: simplicity and extensibility. Polymorphism is one of the benefits of oop as it gives scope to a code to be in more than one form.

6. Solving problems

Problems can be efficiently solved by breaking down the problem into smaller pieces and this makes as one of the big advantages of object-oriented programming. If a complex problem is broken down into smaller pieces or components, it becomes a good programming practice. Considering this fact, OOPS utilizes this feature where it breaks down the code of the software into smaller pieces of the object into bite-size pieces that are created one at a time. Once the problem is broken down, these broken pieces can be used again to solve other problems. Also, the more minor codes can get replaced through the modules with the same interface having the implementation details.

7. Security

Because of the concept of data abstraction in OOPS, only a limited amount of data is shown to the user which makes good benefits of oop. The rest data is not exposed while exposing only the required amount of data. Therefore, it allows the maintenance of security. Another set of benefits of oop in java concept of abstraction is used to hide the complexity from other users and demonstrate the element's information as per the requirements. It also helps in avoiding repetitive code. Another concept provided in OOPS is the feature of encapsulation that allows the protection of the data in the classes from getting accessed by the system. All the internal contents in the class can be safeguarded. In Java, encapsulation is mainly used for restricting access to the class fields directly while setting all the fields of the class to private.

The code in the OOPS is easy maintenance coding due to the presence of a coding base in a central way. Therefore, it is easy to create procedure code that can be easily maintained.

Q47. Explain method overloading?

Answer = >

1. If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.
2. Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters, or a mixture of both.
3. Method overloading is also known as Compile-time Polymorphism, Static Polymorphism, or Early binding in Java. In Method overloading compared to parent argument, child argument will get the highest priority.

Different Ways of Method Overloading in Java

1. Changing the Number of Parameters.

Method overloading can be achieved by changing the number of parameters while passing to different methods.

Example: public int multiply (int a, int b)

public int multiply (int a, int b, int c)

2. Changing Data Types of the Arguments.

In many cases, methods can be considered Overloaded if they have the same name but have different parameter types, methods are considered to be overloaded.

Example: public int Prod (int a, int b, int c)

public double Prod (double a, double b, double c)

3. Changing the Order of the Parameters of Methods

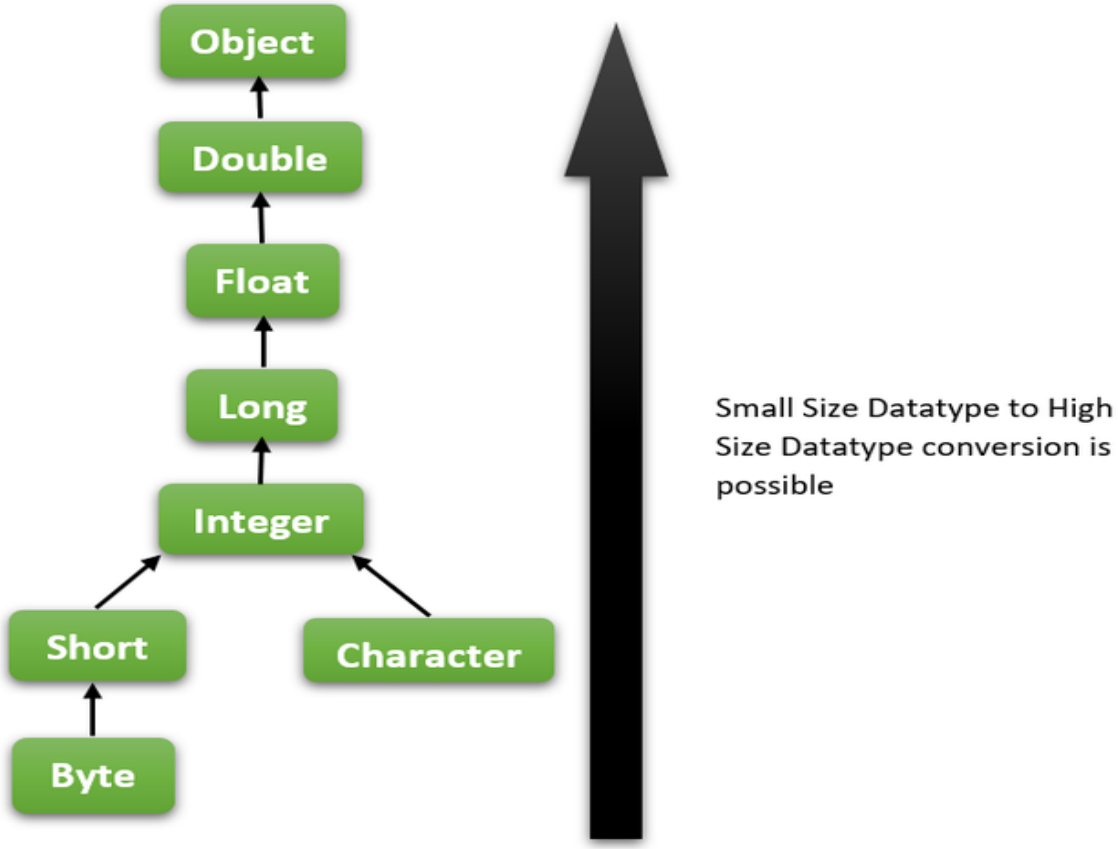
Method overloading can also be implemented by rearranging the parameters of two or more overloaded methods. For example, if the parameters of method 1 are (String name, int roll_no) and the other method is (int roll_no, String name) but both have the same name, then these 2 methods are considered to be overloaded with different sequences of parameters.

Example: public void StudentId (String name, int roll_no)

public void StudentId (int roll_no, String name)

What if the exact prototype does not match with arguments?
Priority-wise, the compiler takes these steps:

- Type Conversion but to higher type(in terms of range) in the same family.
- Type conversion to the next higher family (suppose if there is no long data type available for an int data type, then it will search for the float data type).



Can we overload java main () method?

Yes, by method overloading. You can have any number of main methods in a class by method overloading. But JVM calls main () method which receives string array as arguments only.

Q48. Explain the difference among early binding and late binding?
Answer = >

Early Binding: The binding which can be resolved at compile time by the compiler is known as static or early binding. Binding of all the static, private and final methods is done at compile-time.

Late binding: In the late binding or dynamic binding, the compiler doesn't decide the method to be called. Overriding is a perfect example of dynamic binding. In overriding both parent and child classes have the same method.

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

49. Explain early binding? Give examples?
Answer = >

Binding refers to the association (linking) between method call and method definition (Body). Binding takes place either at compiler time or run time.

The binding which can be resolved at compile time by compiler is known as static or early binding. The binding of static, private and final methods is compile-time. The reason is that these methods cannot be overridden and the type of the class is determined at the compile time.

Static binding example

Here we have two classes Human and Boy. Both the classes have same method walk () but the method is static, which means it cannot be overridden so even though I have used the object of Boy class while creating object obj, the parent class method is called by it. Because the reference is of Human type (parent class). So whenever a binding of static, private and final methods happen, type of the class is determined by the compiler at compile time and the binding happens then and there.

```
class Human{
  public static void walk()
  {
     System.out.println("Human walks");
  }
}
class Boy extends Human{
  public static void walk(){
     System.out.println("Boy walks");
  }
  public static void main( String args[]) {
     // Reference is of Human type and object is Boy type
      Human obj1 = new Boy();

     // Reference is of HUman type and object is of Human type.
     Human obj2 = new Human();
```

```
    obj1.walk();
    obj2.walk();
  }
}
```

Output:

```
Human walks
Human walks
```

==========

Q50.Explain loose coupling and tight coupling?

Answer =>

A situation where an object can be used by another object is termed as coupling. It is the process of collaborating together and working for each other. It simply means that one object requires another object to complete its assigned task. It is basically the usage of an object by another object, thereby reducing the dependency between the modules. It is called as collaboration if one class calls the logic of another class.

Types of Coupling
Coupling in Java is further divided into two types, namely

Tight coupling :

1. The tightly coupled object is an object that needs to know about other objects and is usually highly dependent on each other's interfaces.
2. Changing one object in a tightly coupled application often requires changes to a number of other objects.
3. 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:

1. 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.
2. 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.
3. When two classes, modules, or components have low dependencies on each other, it is called loose coupling in Java. Loose coupling in Java means that the classes are independent of each other. The only knowledge one class has about the other class is what the other class has exposed through its interfaces in loose coupling. If a situation requires objects to be used from outside, it is termed as a loose coupling situation.

51. Give an example among tight coupling and loose coupling.
Answer =>
Tight coupling example:

```
class Volume
{
    public static void main(String args[])
    {
        Box b = new Box(5,5,5);
        System.out.println(b.volume);
    }
}
class Box
{
    public int volume;
    Box(int length, int width, int height)
    {
        this.volume = length * width * height;
    }
}
```

Output:
125

Explanation: In the above example, there is a strong inter-dependency between both the classes. If there is any change in Box class, then they reflect in the result of Class Volume.

Loose coupling example:

```
class Volume
{
    public static void main(String args[])
    {
        Box b = new Box(5,5,5);
        System.out.println(b.getVolume());
    }
}
final class Box
{
    private int volume;
    Box(int length, int width, int height)
    {
        this.volume = length * width * height;
    }
    public int getVolume()
    {
        return volume;

    }
}
```

Output:

125

Explanation: In the above program, there is no dependency between both the classes. If we change anything in the Box classes, then we don’t have to change anything in Volume class.

52. Write in brief abstract class.

Answer =>

Data abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either abstract classes or interfaces

The abstract keyword is a non-access modifier, used for classes and methods:

Abstract class: is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).

Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

In java, the following some important observations about abstract classes are as follows:

1. An instance of an abstract class cannot be created.
2. Constructors are allowed.
3. We can have an abstract class without any abstract method.
4. There can be a final method in abstract class but any abstract method in class (abstract class) cannot be declared as final or in simpler terms final method cannot be abstract itself as it will yield an error: “Illegal combination of modifiers: abstract and final”
5. We can define static methods in an abstract class
6. We can use the abstract keyword for declaring top-level classes (Outer class) as well as inner classes as abstract
7. If a class contains at least one abstract method then compulsory should declare a class as abstract
8. If the Child class is unable to provide implementation to all abstract methods of the Parent class, then we should declare that Child class as abstract so that the next level Child class should provide implementation to the remaining abstract method.

Program:

```
abstract class Demo {
   abstract void m1();
   abstract void m2();
   abstract void m3();
   void m4() {
     System.out.println("Inside m4");
   }

}
abstract class FirstChild extends Demo {
   public void m1() {
     System.out.println("Inside m1");
   }
}
class SecondChild extends FirstChild {
   public void m2() {
     System.out.println("Inside m2");
   }
   public void m3() {
     System.out.println("Inside m3");
   }
}
class GFG {
   public static void main(String[] args)
   {
      // if we remove the abstract keyword from FirstChild
      // Class and uncommented below obj creation for
      // FirstChild then it will throw
      // compile time error as did't override all the
      // abstract methods

      // FirstChild f=new FirstChild();
      // f.m1();
      SecondChild s = new SecondChild();
      s.m1();
      s.m2();
      s.m3();
      s.m4();
   }
}
```

Output:
Inside m1
Inside m2
Inside m3
Inside m4

Q53. Define the Benefits of oops over pop?

Answer=>

Procedure-oriented Programming(POP) and **Object-oriented programming(OOP)** both are the programming approaches, which uses high-level language for programming. A program can be written in both the languages, but if the task is highly complex, OOP operates well as compared to POP. In POP, the 'data security' is at risk as data freely moves in the program, as well as, 'code reusability' is not achieved which makes the programming lengthy, and hard to understand.

Large programs lead to more bugs, and it increases the time of debugging. All these flaws lead to a new approach, namely "object-oriented programming". In object-oriented programming's primary concern is given on '**data security**'; it binds the data closely to the functions which operate on it.

It also resolves the problem of '**code reusability**', as if a class is created, its multiple instances(objects) can be created which reuses the members and member functions defined by a class. There are some other differences which can be explained with the help of a comparison chart.

Comparison Chart

BASIS FOR COMPARISON	Procedure-oriented Programming(POP)	Object-oriented programming(OOP)
Basic	Procedure/Structure oriented .	Object oriented.
Approach	Top-down.	Bottom-up.
Basis	Main focus is on "how to get the task done" i.e. on the procedure or structure of a program .	Main focus is on 'data security'. Hence, only objects are permitted to access the entities of a class.
Division	Large program is divided into units called functions.	Entire program is divided into objects.
Entity accessing mode	No access specifier observed.	Access specifier are "public", "private", "protected".
Overloading or Polymorphism	Neither it overload functions nor operators.	It overloads functions, constructors, and operators.
Inheritance	There is no provision of inheritance.	Inheritance achieved in three modes public private and protected.
Data hiding & security	There is no proper way of hiding the data, so data is insecure	Data is hidden in three modes public, private, and protected. hence data security increases.
Data sharing	Global data is shared among the functions in the program.	Data is shared among the objects through the member functions.
Virtual classes or virtual function	No concept of virtual classes .	Concept of virtual function appear during inheritance.
Example	C, VB, FORTRAN, Pascal	C++, JAVA, VB.NET, C#.NET.

54. Explain Generalization and Specialization?

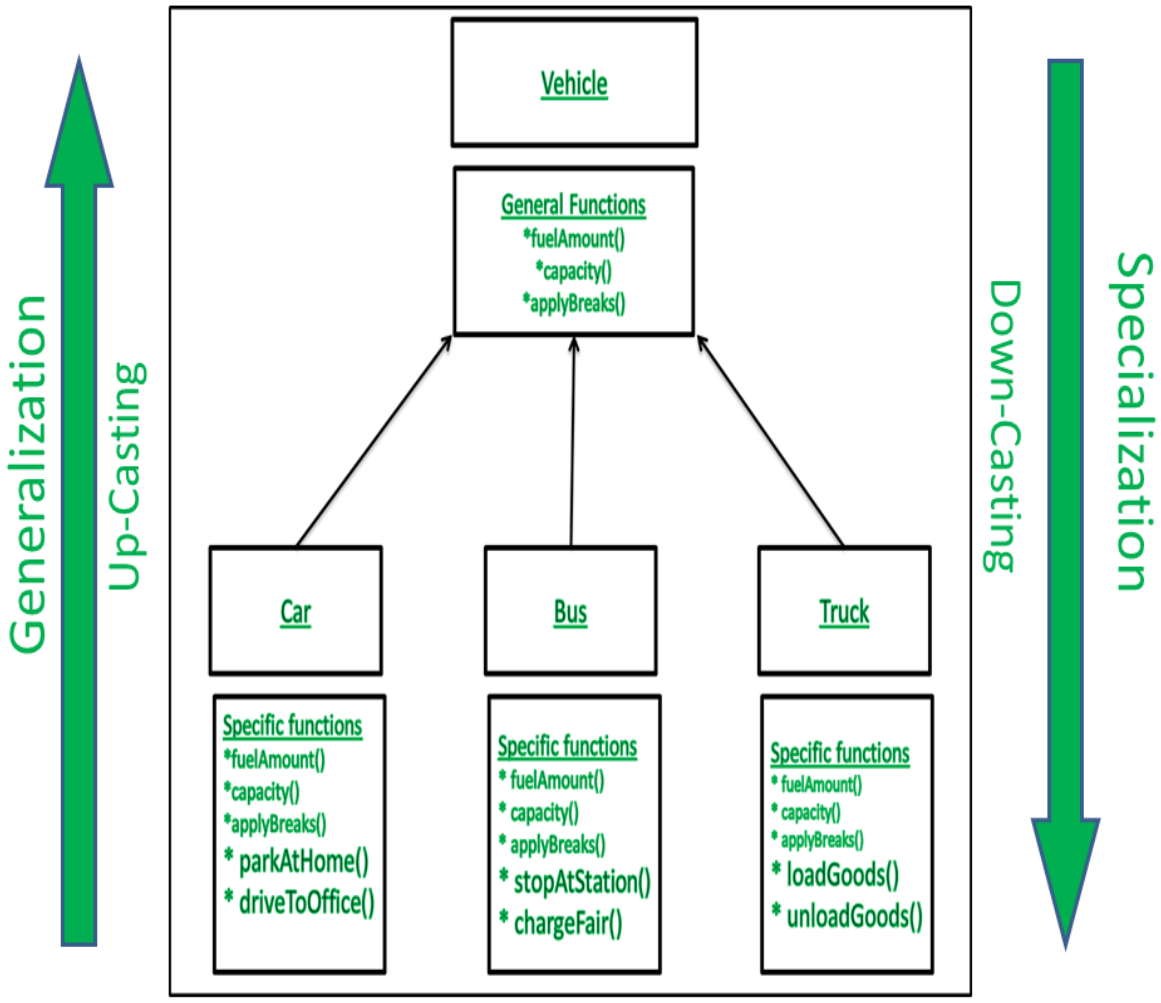
Answer =>

General class:

Loosely speaking, a class which tells the main features but not the specific details. The classes situated at the top of the inheritance hierarchy can be said as General.

Specific class:

A class which is very particular and states the specific details. The classes situated at the bottom of the inheritance hierarchy can be said as Specific.



Therefore,

- References for Vehicle and Bus can be type-casted to each other.
- References for Vehicle and Car can be type-casted to each other.
- References for Vehicle and Truck can be type-casted to each other.
- References for Bus, Car and Truck can’t be type-casted to each other.

Generalization:

Converting a subclass type into a superclass type is called **Generalization** because 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.

For example, if we say Car is a Vehicle, there will be no objection. Thus Java compiler will not ask for cast operator in generalization. So, in widening or Generalization, **we can access all the superclass methods, but not the subclass methods**.

Specialization:

Converting a super class type into a sub class type is called **Specialization**. Here, we are coming down from more general form to a specific form and hence the scope is narrowed. Hence, this is called **narrowing** or **down-casting**.

Narrowing is **not safe** because the classes will become more and more specific thus giving rise to more and more doubts. For example, if we say Vehicle is a Car we need a proof. Thus, in this case, Java compiler specifically asks for the casting. This is called **explicit casting**.

```
class Father {
	public void work()
	{
		System.out.println("Earning Father");
	}
}

class Son extends Father {
	public void play()
	{
		System.out.println("Enjoying son");
	}
}
```

```
class Main {
        public static void main(String[] args)
        {
                // father is a superclass reference
                Father father;

                // new operator returns a subclass reference
                father = (Father) new Son();

                // which is widened using casting and stored in father variable
                // Though casting is done but it is not needed

                father.work();

                // Uncomment next line to see the error
                // father.play();
 // son is a sub class reference new operator returns a superclass reference
 // which is narrowed using casting and stored in son variable
                Son son;
           // This will throw exception
          son = (Son) new Father();

// Through a narrowed reference of the superclass we can neither access
superclass method and nor the subclass methods
  // Below lines will show an error when uncommented
           // son.work();
           // son.play();
            father = new Son();
           // father is narrowed
            son = (Son)father;
          son.work(); // works well
          son.play(); // works well
        }
}
```

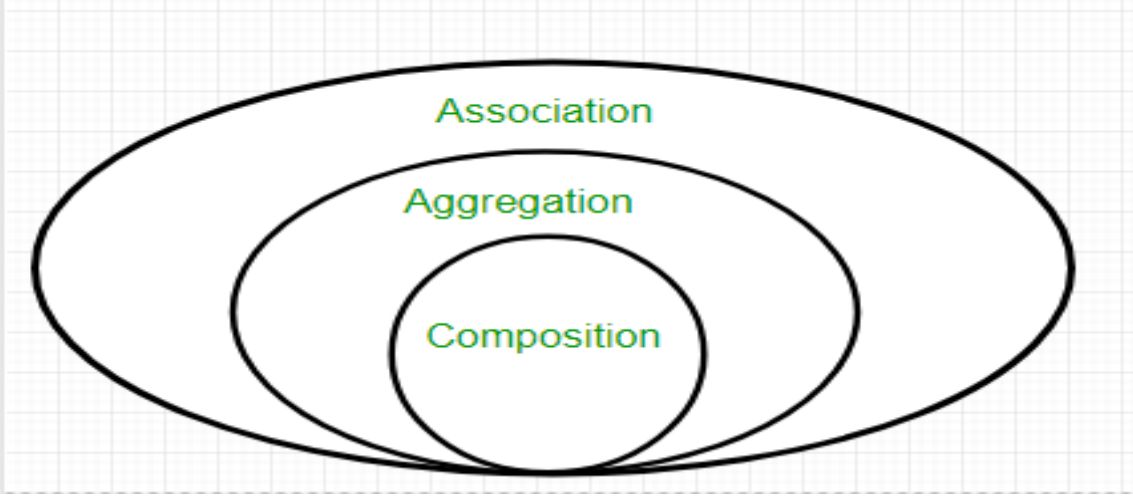
Conclusion:

1. When a superclass reference (referring to superclass object) is narrowed, then using that reference we can access neither methods of subclass nor methods of superclass.
2. When a subclass reference (referring to subclass object) is widened and then again narrowed, then using that reference we can access all the methods of the subclass as well as the superclass. This is the same as simple base class reference referring to base class object where superclass methods have got inherited.

Q55. Write in brief Association, Aggregation and Composition?

Answer=>

Association is a relation between two separate classes which establishes through their Objects. Association can be one-to-one, one-to-many, many-to-one, many-to-many. In Object-Oriented programming, an Object communicates to another object to use functionality and services provided by that object. **Composition** and **Aggregation** are the two forms of association.



Aggregation:

Aggregation is a special case of association. A directional association between objects. When an object ‘has-a’ another object, then you have got an aggregation between them. Direction between them specified which object contains the other object. Aggregation is also called a “Has-a” relationship.

- It represents Has-A’s relationship.
- It is a unidirectional association i.e. a one-way relationship. For example, a department can have students but vice versa is not possible and thus unidirectional in nature.
- In Aggregation, both the entries can survive individually which means ending one entity will not affect the other entity.

Aggregation Example :

Now consider class **Car** and class **Wheel**. Car needs a Wheel object to function. Meaning the Car object owns the Wheel object but we cannot say the Wheel object has no significance without the Car Object. It can very well be used in a Bike, Truck or different Cars Object.

Composition:

Composition is a special case of aggregation. In a more specific manner, a restricted aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition.

- It represents part-of relationship.
- In composition, both entities are dependent on each other.
- When there is a composition between two entities, the composed object cannot exist without the other entity.

Composition Example: Consider the example of a Car and an engine that is very specific to that car (meaning it cannot be used in any other car). This type of relationship between Car and SpecificEngine class is called Composition. An object of the Car class cannot exist without an object of SpecificEngine class and object of SpecificEngine has no significance without Car class. To put in simple words Car class solely "owns" the SpecificEngine class.

Q56. Write in brief Object Composition vs. Inheritance.

Basis of Comparison	**Composition**	**Inheritance**
Relationship	It is a "has-a" kind of circumstance.	It's a case of "is-a."
Functionality	We can evaluate the functioning of the classes that we are using without having to be concerned with determining whether or not they are the parent or the child classes.	It is not possible to test a child class without first testing its parent class.
It is not possible to test a child class without first testing its parent class.	Because of composition, it is possible to reuse the code even in the final classes.	nheritance cannot be used to extend the functionality of the final class.
Significance	Simply declaring a type that we want to use in composition allows it to store several implementations, each of which can have unique behaviours depending on the context of when they are called.	In inheritance, we are responsible for defining the class that will become our "superclass," and this definition is immutable once it has been used.
Class Combining	Composition gives users the flexibility to mix features and capabilities from multiple classes into a single entity.	Java does not support multiple inheritances, which means that several classes cannot be extended.

Q 57. Explain cohesion?

Cohesion is a measure of the degree to which the elements of the module are functionally related. It is the degree to which all elements directed towards performing a single task are contained in the component. Basically, cohesion is the internal glue that keeps the module together. A good software design will have high cohesion.

Q 58. Explain “black-box-reuse” and “white-box-reuse”?

Black-Box reuse means that you use component without knowing it's internals. All you have is a component interface. White-box reuse means that you know how component is implemented. Usually White-box reuse means class inheritance.

Q59. Explain “this”.

The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

Usage of java this keyword.

1.this can be used to refer current class instance variable.

2.this can be used to invoke current class method (implicitly).

3.this() can be used to invoke current class constructor.

4.this can be passed as an argument in the method call.

5.this can be passed as argument in the constructor call.

6.this can be used to return the current class instance from the method.

Q60. Write in brief static member and member functions.

Static data members are class members that are declared using static keywords. A static member has certain special characteristics. These are: Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created.

Q61. How will you relate unrelated classes or how will you achieve polymorphism without using the base class?

we can achieve polymorphism without inheritance using composition. Composition is an OOP topic, when we create an object of a class in an other class we called it composition.

Q62. Explain the Diamond problem?

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 and sub-class. When we call the method, the compiler gets confused and cannot determine which class method to be called and even on calling which class method gets the priority.

Q63. 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.

Q64. Explain the need of abstract class?

Abstract Class in Java does the process of hiding the intricate code implementation details from the user and just provides the user with the necessary information. This phenomenon is called Data Abstraction in Object-Oriented Programming

Q65. Why can’t we instantiate abstract class?

Abstract classes cannot be instantiated, but they can be subclassed. When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract .

Q.66.Can abstract class have constructors?

Ans.Yes, abstract class have constructor.

Q.67.How many instances can be created for an abstract class?

Ans. The answer to the question of how many instances of an abstract class can be created is zero. That is, we cannot create an instance of an abstract class as it does not have any complete implementation. An abstract class acts like a template or an empty structure.

Q.68.Which keyword can be used for overloading?

Ans.Super keyword is used for overloading. If both parent & child classes have the same method, then the child class would override the method available in its parent class.

Q.69.Explain the default access specifiers in a class definition?

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.

Example of default access modifier

In this example, we have created two packages pack and mypack. We are accessing the A class from outside its package, since A class is not public, so it cannot be accessed from outside the package.

```
1. //save by A.java
2. Package pack;
3. class A{
4.   void msg(){System.out.println("Hello");}
5. }
```

```
1. //save by B.java
2. package mypack;
3. import pack.*;
4. class B{
5.   public static void main(String args[]){
6.    A obj = new A();//Compile Time Error
7.    obj.msg();//Compile Time Error
8.   }
9. }
```

In the above example, the scope of class A and its method msg() is default so it cannot be accessed from outside the package.

Q.70.Define all the operators that cannot be overloaded?

Ans. Unlike C++, Java doesn't support operator overloading. Java doesn't provide freedom to programmers, to overload the standard arithmetic operators e.g. +, -, * and / etc. If you have worked previously in C++, then you know that Java has left a lot of features supported in C++ e.g. Java doesn’t provide multiple inheritence, no pointers in Java, and no pass-by-reference .

Q.71. Explain the difference among structure and a class?

Ans.

Features	Structure	Class
Definition	A structure is a grouping of variables of various data types referenced by the same name.	In C++, a class is defined as a collection of related variables and functions contained within a single structure.
Basic	If no access specifier is specified, all members are set to 'public'.	If no access specifier is defined, all members are set to 'private'.

Declaration	struct structure_name{
	type struct_member 1;
	type struct_member 2;
	type struct_member 3;
	.
	type struct_memberN;
	};
		class class_name{
		data member;
		member function;
		};

Instance	Structure instance is called the 'structure variable'.	A class instance is called 'object'.
Inheritance	It does not support inheritance.	It supports inheritance.
Memory Allocated	Memory is allocated on the stack.	Memory is allocated on the heap
Nature	Value Type	Reference Type
Purpose	Grouping of data	Data abstraction and further inheritance.
Usage	It is used for smaller amounts of data.	It is used for a huge amount of data
Null values	Not possible.	It may have null values

Q.72. Explain the default access modifier in a class?

Ans. 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. The fields in an interface are implicitly public static final and the methods in an interface are by default public.

Example

Variables and methods can be declared without any modifiers, as in the following examples -

```
String version = "1.5.1";
boolean processOrder() {
   return true;
}
```

Q.73.Can you list out the different types of constructors?

Ans.

- Default Constructor.
- Parameterized Constructor.
- Copy Constructor.
- Static Constructor.
- Private Constructor.

Q.74.Explaina ternary operator?

Ans. The meaning of **ternary** is composed of three parts. The **ternary operator (? :)** consists of three operands. It is used to evaluate Boolean expressions. The operator decides which value will be assigned to the variable. It is the only conditional operator that accepts three operands. It can be used instead of the if-else statement. It makes the code much more easy, readable, and shorter.

Syntax:

1. variable = (condition) ? expression1 : expression2

Q.75 Do We Require Parameter For Constructors?

Ans. Default Constructor – A constructor that accepts no parameter is called Default Constructor. It is not necessary to have a constructor block in your class definition. If you don't explicitly write a constructor, the compiler automatically inserts one for you.

Q76. Explain Sealed Modifiers

The sealed modifier is used to prevent derivation from a class. An error occurs if a sealed class is specified as the base class of another class. Sealed classes are an enhancement to the language included with LTS version 17 (included since version 16) of Java. Sealed classes or interfaces allow us to restrict what other classes and interfaces can extend from them.

Q77. Explain The Difference Between New And Override override:

overrides the functionality of a virtual method in a base class, providing different functionality. new: hides the original method (which doesn't have to be virtual), providing different functionality. This should only be used where it is absolutely necessary. When you hide a method, you can still access the original method by up casting to the base class. This is useful in some scenarios, but dangerous.

Q78. How Can We Call The Base Method Without Creating An Instance?

It is possible if it’s a static method. It is possible by inheriting from that class also.It is possible from derived classes using base keyword.

Q79. Define The Various Types Of Constructors

1. Do Nothing Constructor
2. Default Constructor
3. Parameterized Constructor
4. Copy Constructor

Q80. Define Manipulators

Manipulators are helping functions that can modify the input/output stream. It does not mean that we change the value of a variable, it only modifies the I/O stream using insertion (<>) operators.

• Manipulators are special functions that can be included in the I/O statement to alter the format parameters of a stream.

• Manipulators are operators that are used to format the data display.

• To access manipulators, the file iomanip.h should be included in the program.

Q81.

What are tokens

Java compiler breaks the line of code into text (words) is called Java tokens. These are the smallest element of the ~~For~~ Java program. The Java compiler identified these words as tokens. These tokens are separated by the delimiters.

Eg public class Demo{

public static void main (String [] args)
{
System.out.println ("Java");
}
}

Here public, class, Demo, {, static, void main, (, String, args, [,],), System, ., out, println, Java, are the Java tokens.

Q82. Explain structured programming and its disadvantage?

Structured Programming Approach, as the word suggests, can be defined as a programming approach in which the program is made as a single structure. It means that the code will execute the instruction by instruction one after the other. It doesn't support the possibility of jumping from one instruction to some other with the help of any statement like GOTO, etc. Therefore, the instructions in this approach will be executed in a serial and structured manner. The languages that support Structured programming approach are:

C

C++

Java

C#

Disadvantages of Structured Programming Approach:

Since it is Machine-Independent, So it takes time to convert into machine code.

The converted machine code is not the same as for assembly language.

The program depends upon changeable factors like data-types. Therefore it needs to be updated with the need on the go.

Usually the development in this approach takes longer time as it is language-dependent. Whereas in the case of assembly language, the development takes lesser time as it is fixed for the machine

Q83.

When to use interface over abstract class.

* If the functionality we are creating will be useful across a wide range of disparate objects use an interface.
* Interface is a good choice when we think that the API will not change for a while.
* We can use interface when we want something similar to multiple inheritance.
* If we are designing small, concise bits of functionality, use interface. If we are designing large functional units, use an abstract class.

Q84. Explain a private constructor? Where will you use it?

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 declared class. You cannot access its objects from outside the constructor class.

Private constructor uses.

- You can use it with static members - only classes.
- You can use it with static utility or constant classes.
- You can use it to serve singleton classes
- You can use it to assign a name, for instance, creation by utilising factory methods.
- You can use it to prevent subclassing

Q85.

Ques Can we override private virtual methods?

If a class (derived) extends a class (Base) then we cannot override private methods of Base class in derived class

Eg
```
Class Base{
    Private void fun(){}
}
Class derived extends Base{
    Private void fun(){}
}
PSVM(){
    derived d = new derived();
    d.fun();
}
```

Above function will give error

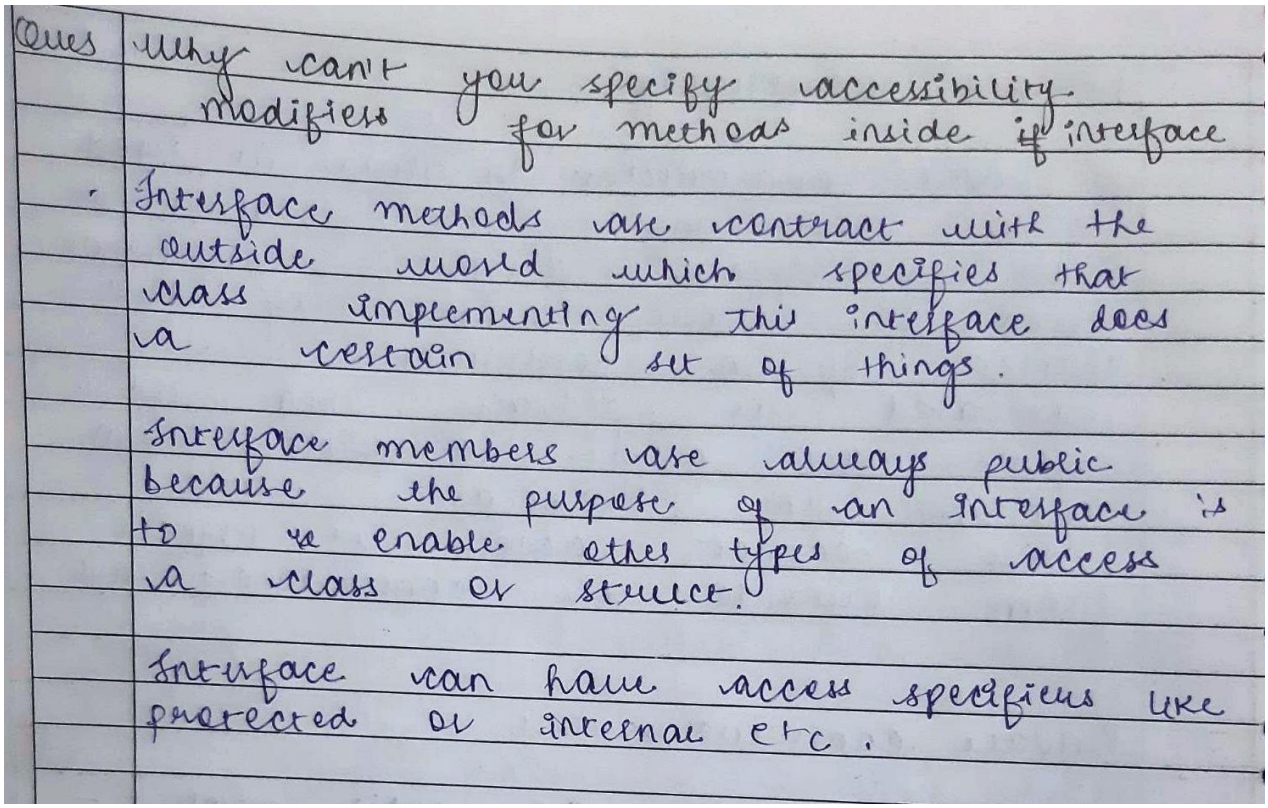
* An inner class can access private members of its outer class.

Q 86. Can you allow class to be inherited, but prevent from being over-ridden?

Ans:-

Yes. By using sealed keyword.

Q87.



Q88.

Can static members use non static members

A static method cannot access non-static variable or method because static ~~met~~ methods can be accessed without instantiating the class.

If the class is not instantiated the variables are not initialized and thus cannot be accessed from a static method.

Also static members are linked to classes and get memory once the class is loaded but non static or instance members get once when an instance is created for that class.

If we call any non-static member from static context, it means that when static member is loaded ~~when~~, it will search for the non-static member which is not in existence.

Q89.

Define different ways of method can be overloaded.

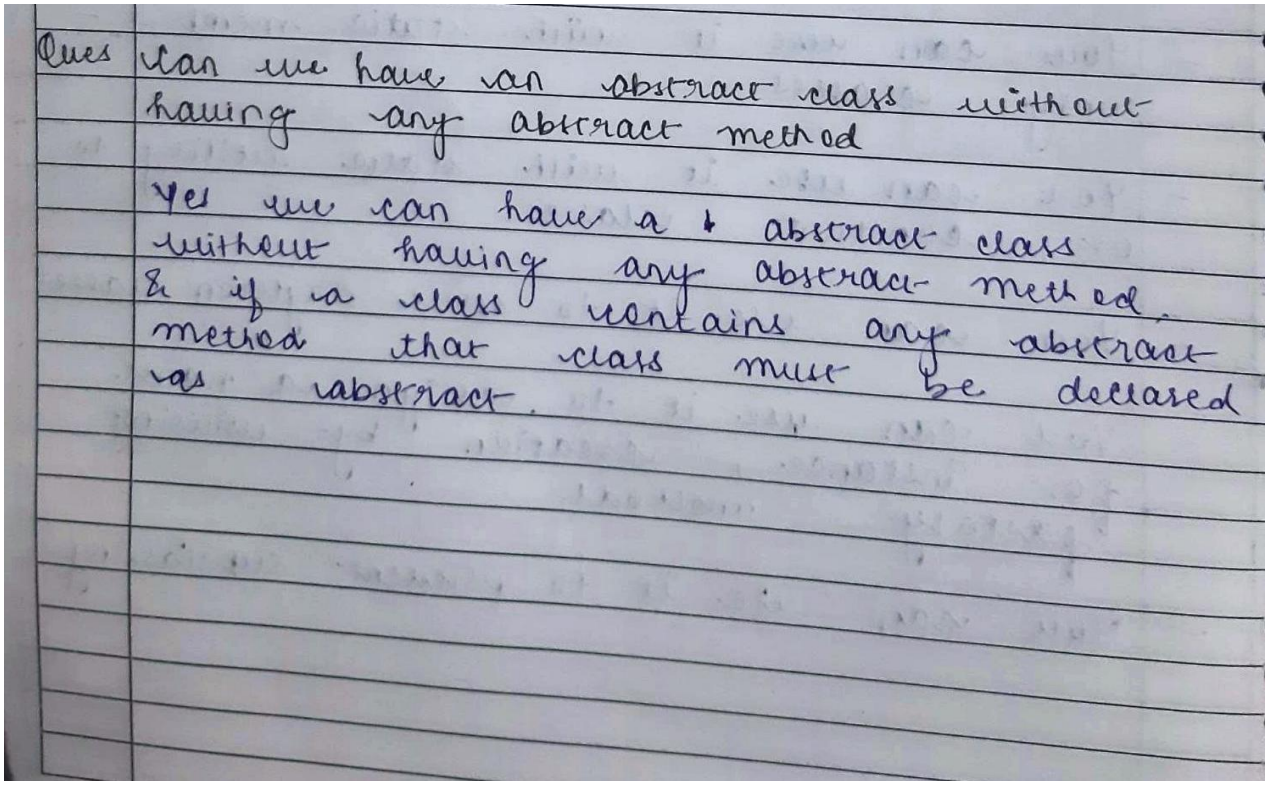
Method overloading can be done in several ways -

(i) By changing no of parameters in 2 methods

(ii) By changing datatypes of the parameters of method.

(iii) By changing Order of the parameters of method.

Q90.



Ques Can we have an abstract class without having any abstract method

Yes we can have a abstract class without having any abstract method. & if a class contains any abstract method that class must be declared as abstract.

Q91. Explain the default access modifier of a class?

- As the name suggests access modifiers in Java helps to restrict the scope of a class, constructor, variable, method, or data member.

- There are four types of access modifiers available in java :

(a) Default – (No keyword required) 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.

(b) Private - The access level of a private modifier is only within the class.
It cannot be accessed from outside the class.

(c) Protected - The access level of a protected modifier is within the package and outside the package through child class.
If you do not make the child class, it cannot be accessed from outside the package.

(d) Public - The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

- There are also many non-access modifiers, such as static, abstract, synchronized, native, volatile, transient, etc.

Q92) 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.

Q93)Why is Function overloading not possible with different return types?

No.

Function overloading comes under the compile-time polymorphism. During compilation, the function signature is checked. So, functions can be overloaded, if the signatures are not the same.

The return type of a function has no effect on function overloading, therefore the same function signature with different return type will not be overloaded.

Example:-

```
public
class Main {
public
    int foo() { return 10; }
public
    char foo() { return 'a'; }
    // compiler error as it is a new declaration of fun()
public
    static void main(String args[]) {}
}
```

Output

```
prog.java:10: error: method foo() is already defined in
class Main
   char foo() { return 'a'; }
      ^
1 error
```

Q94) Can abstract class have a constructor?

Yes,It can have a constructor.

- If you define your own constructor without arguments inside an abstract class but forget to call your own constructor inside its derived class constructor then JVM will call the constructor by default.
- So if you define your single or multi-argument constructor inside the abstract class then make sure to call the constructor inside the derived class constructor with the ‘super’ keyword.

Q95. Define rules of Function overloading and function overriding?

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

1. By changing number of arguments

In this example, we have created two methods, first add() method performs addition of two numbers and second add method performs addition of three numbers.

```
class Adder{
static int add(int a,int b){return a+b;}
static int add(int a,int b,int c){return a+b+c;}
}
class TestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

2. By changing the data type

In this example, we have created two methods that differs in data type. The first add method receives two integer arguments and second add method receives two double arguments.

```
class Adder{
static int add(int a, int b){return a+b;}
static double add(double a, double b){return a+b;}
}
class TestOverloading2{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6));
}}
```

Rules for Java Method Overriding

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).

Example: