INTERVIEW QUESTIONS

Diwali Assignment 02

1. Explain OOPS?

The Object-Oriented Programming (or OOPS) is basically a computer programming designphilosophy or methodology that organizes/ models software design around data, or objectsrather than functions and logic.
An object is referred to as a data field that has unique attributes and behavior.
Everything in OOPis grouped as self-sustainable objects.

2. Explain an abstraction? Real life example.

Abstraction is a process of hiding the

implementation detailsand showing only

functionality to the user.

Or ,we can say that it shows only essential things to the user and hides the

internal detailsExamples -ATM machine , working of calls through mobile

etc.

3. Explain encapsulation? Real life example.

Encapsulation is a process of binding data members and member

variables togetherto avoid direct access of variables. This ensures

privacy.

In encapsulation, we declare fields as private

In the class to prevent other classes from accessing

them directly.Examples - How my voice is generated

through my throat ,how the video comes through a

SIM etc.

4. Explain the relationship among abstraction and encapsulation?

Abstraction is hiding the details and implementation of the code.

Whereas, Encapsulation is hiding the data and controlling the visibility of the code.

Abstraction is a design level process ; whereas Encapsulation is an implementation level process.

Abstraction focus is on “what” should be done ; whereas Encapsulation focus is on “How” it shouldbe done.

5. Explain polymorphism?

The ability to exist in different form is called polymorphism.

Ex. a+b Polymorphism in OOP is the ability of an entity to take several forms.

In other words, it refers to the ability of a reference to an object to take different forms.Example - A man at the same time is a father, a husband, an employee.

5. Explain Inheritance?

- The process by which one class acquires the properties (data members) andfunctionalities(methods) of another class is called inheritance.
- The class which inherits properties from another class is called subclass orchild class or derived class.
- The class whose properties are inherited by subclass is called superclass orparent class or base class.
- subclass/child-class can inherit properties of its superclass/parent-class, aswell as it can contain its own properties
-

6. How composition is better than inheritance?

Composition is the architecture strategy for executing a relationship between objects.Java composition is done using instance variables from other objects.

It is better because -

1. Flexibility - If you use Composition, you are flexible enough to replace the betterand updated version of the Composed class implementation.Ex - Comparator class.
2. Testing scope - Unit testing is easy in composition because we know what all methods we areusing from another class.

7. Which OOPS concept is used as a reuse mechanism?

Inheritance

8. **Which OOPS concept exposes only the necessary information to the calling functions?**Encapsulation

9. Explain a class? Create a class.

A class is a blueprint for declaring and creating objects.

Class is a user-defined data type, that holds its own data members and member functions,which can be accessed and used by creating an instance of that class.

Definition - <modifier> class

<ClassName>class Shape

```
{

}
```

10. Using above created class, Write in brief abstraction and encapsulation.class Shape

```
{
        private double
        height;private
        double width;

        void setValues(double height,double width)
        {
                this.height=
                height;
                this.width=w
                idth;
        }
        double getHeight()
        {
                return height;
        }
        double getWidth()
        {
                return width;
        }
}
class Rectangle extends Shape
{
        double area;
```

```
	double getArea()
	{

		this.area =
		this.height*this.width;
		return area;
	}
	void printArea()
	{
		System.out.println("Area of Rectangle is : "+getArea());
	}
}
class Triangle extends Shape
{
	double area;

	double getArea()
	{
		this.area =
		0.5*height*width;
		return area;
	}
	void printArea()
	{
		System.out.println("Area of triangle is : " + getArea());
	}
}
class AreaDemo
{
	public static void main(String args[])
	{
		System.out.println("Enter height of Rectangle");

		Scanner sc = new
```

```
        Scanner(System.in);double h
        =sc.nextDouble();
        System.out.println("Enter width of
        Rectangle");double w
        =sc.nextDouble();
        Rectangle r1 = new
        Rectangle();
        r1.setValues(h,w);
        r1.printArea();
        System.out.println();
        System.out.println("Enter height of Triangle");

        double h1 =sc.nextDouble();
        System.out.println("Enter width of
        Triangle");

        double w1 =sc.nextDouble();
        Triangle t1 = new
        Triangle();
        t1.setValues(h1,w1);
        t1.printArea();

    }
}
```

11. Explain difference among class and object?

Class is a user-defined data type, that holds its own data members and member functions,which can be accessed and used by creating an instance of that class.

It is the blueprint of any object.Object is an instance of a class.

All data members and member functions of the class can be accessed with the help of objects.When a class is defined, no memory is allocated, but memory is allocated when it is instantiated.

12. Define 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:

1.pub

lic

2.prot

ected

3.def

ault

4.priv

ate

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

Object is an instance of a class.

All data members and member functions of the class can be accessed with the

help of objects.When a class is defined, no memory is allocated,

but memory is allocated when it is

instantiated. Creating objecty ->

Triangle t = new Triangle();

14. Give real life examples of object.

Dogs - They all have state and behavior.'

Dogs have state (name, color, breed, hungry) and behavior (barking, fetching, waggingtail).Some other examples are - Employee, Shape etc.

15. Explain a Constructor.

A constructor in Java is a special method that is used to

initialize objects.The constructor is called when an object of a

class is created.

It can be used to set initial values for object attributes

16. **Define the various types of constructors**?

There are 2 types of constructors -

1.Java Default Constructor - A constructor is called "Default Constructor" when it doesn't haveany parameter.

It is used to provide the default values to the object.

If we don't create any constructor , compiler provides us a constructor automatically.

2.Java Parameterized Constructor - A constructor which has a specific number of parameters iscalled a parameterized constructor.

The parameterized constructor is used to provide different values to distinct objects.

17. Whether static method can use nonstatic members?

No.

18. Explain Destructor?

A destructor is a member function that is invoked automatically when the object goes out of scopeor is explicitly destroyed by a call to delete.

Destructors in Java also known as finalizers are non-deterministic.

19. Explain an Inline function?

An inline function is one for which the compiler copies the code from the function definition

directly into the code of the calling function rather than creating a separate set of instructions inmemory.

This eliminates call-linkage overhead and can expose significant optimization

opportunities.Java does not provide inline functions, it is typically done by the

JVM at execution time

20. Explain a virtual function?

A virtual function or virtual method in an OOPs language is a function or method used to override the behavior of the function in an inherited class with the same signature to achievethe polymorphism.

21. Explain function overloading?

Function overloading is a feature of object-oriented programming where two or more functions can have the same name but different parameters. When a function name is overloaded with different jobs it is called Function Overloading.

22.Explain a friend function?

In object-oriented programming, a friend function, that is a "friend" of a given class, is a function that is given the same access as methods to private and protected data. A friend function is declared by the class that is granting access, so friend functions are part of theclass interface, like methods.

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

A class that is derived from another class is called a subclass (also a derived class, extendedclass, or child class). The class from which the subclass is derived is called a superclass
(also a base class or a parent class).

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

In Java, all non-static methods are based on the runtime type of the underlying object ratherthan the type of the reference that points to that object. Therefore, it doesn't matter which
type you use in the declaration of the object, the behavior will be the same. 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 arevisible at compile-time.

1. First approach (Referencing using Superclass reference): A reference variable of asuperclass 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 overriddenmethod that will be executed.

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

25.Explain an abstract class?

An abstract class is a template definition of methods and variables of a class (category ofobjects) that contains one or more abstracted methods. Abstract classes are used in all object-oriented programming (OOP) languages, including Java.

26.Explain operator overloading?

Operator overloading is a technique by which operators used in a programming language areimplemented in user-defined types with customized 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 fundamental primitive data types. Operatoroverloading is helpful in cases where the operators used for certain types provide semantics related to the domain context and syntactic support as found in the programming language. It is used for syntactical convenience, readability and maintainability.

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

In the case of Call by Value, when we pass the value of the parameter during the calling ofthe function, it copies them to the function's actual local argument. In the case of Call by Reference, when we pass the parameter's location reference/address, it copies and assigns them to the function's local argument.

28.Explain the super keyword?

The super keyword refers to superclass (parent) objects. It is used to call superclass methods, and to access the superclass constructor. The most common use of the superkeyword is to eliminate the confusion between superclasses and subclasses that have methods with the same name.

29. Explain method overriding?

Method overriding, in object-oriented programming, is a language feature that allows a subclass or child class to provide a specific implementation of a method that is alreadyprovided by one of its super class or parent class.

30. Difference among overloading and overriding?

Overriding	Overloading
Implements “runtime polymorphism”	Implements “compile time polymorphism”
The method call is determined at runtime based on the object type	The method call is determined at compile time
Occurs between superclass and subclass	Occurs between the methods in the same class
Have the same signature (name and methodarguments)	Have the same name, but the parameters aredifferent
On error, the effect will be visible at runtime	On error, it can be caught at compile time

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

A static method can only access static data members and static methods of another class orsame class but cannot access non-static methods and variables. Also, a static method can rewrite the values of any static data member.

32.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 aderived 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 parentclass.

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

In Java, all non-static methods are based on the runtime type of the underlying object ratherthan the type of the reference that points to that object. Therefore, it doesn't matter which
type you use in the declaration of the object, the behavior will be the same. 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 arevisible at compile-time.

1. First approach (Referencing using Superclass reference): A reference variable of asuperclass 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 overriddenmethod that will be executed.

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

34. 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

35. Explain exception handling?

Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program orsystem crashing, and without this process, exceptions would disrupt the normal operation ofa program.

37 **Explain the difference among structure and a class?**

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

38. Explain the default access modifier in a class?

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.

39. Explain a pure virtual function?

A pure virtual function or pure virtual method is **a virtual function that is required to beimplemented 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?

Dynamic polymorphism is a process or mechanism in which a call to an overridden methodis to resolve at runtime rather than compile-time. It is also known as runtime polymorphismor dynamic method dispatch. We can achieve dynamic polymorphism by using the method overriding.

41. Do we require a parameter for constructors?

Constructors: - A Java constructor is special method that is called when an object is instantiated. In other words, when you use the new keyword. The purpose of a Java constructor is to initializes thenewly created object before it is used.

Type of Constructors:

- Compiler provided Default constructor
- User defined- No-arguments Constructor.
- User defined- Constructors with Arguments.
-

A class can have multiple constructors, as long as their signature (the parameters they take) are not the same. You can define as many constructors as you need. When a Java class contains multiple constructors, we say that the constructor is overloaded (comes in multiple versions). This is what constructor overloading means, that a Java class contains multiple constructors.

Based upon the requirement of code, we have to decide type of constructor to be used.

When there is no requirement of initializing data at the time of creation of object then we can go for either default constructor (compiler provided) or user defined No argument constructor.

if there is requirement to dynamically initialize the instance variables with the specified values atthe time of instantiation parameterized constructor is used

Note: - When there is no constructor defined by user in that case compiler provides a default constructor for every object created of that class. But if any constructor (With or Without Argument constructor) is defined by user then compiler do not provide any default constructor to that class. So, it is better to always provide a No Argument constructor inside class even if we donot need it in our program.

42. Explain static and dynamic binding?

Connecting a method call to the method body is known as binding.

There are two types of binding

a) Static Binding (also known as Early Binding).
b) Dynamic Binding (also known as Late Binding).

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

Why binding of static, final and private methods is always a static binding? Static binding is better performance wise (no extra overhead is required). Compiler knows that all such methods cannot be overridden and will always be accessed by object of local class. Hence compiler doesn't have any difficulty to determine object of class (local class for sure). That's the reason binding for such methods is static.

Dynamic Binding: In Dynamic binding 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 same method.

Important Points

- private, final and static members (methods and variables) use static binding while for virtual methods (In Java methods are virtual by default) binding is done during run time based upon run time object.
- Static binding uses Type information for binding while Dynamic binding uses Objects to resolve binding.
- Overloaded methods are resolved (deciding which method to be called when there are multiple methods with same name) using static binding while overridden methods using dynamic binding, i.e., at run time.

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

Abstract Class can have abstract methods and abstract method does not contain any body or definition inside it. This means that Abstract Class does not seem to be a complete class. It's just atemplate.

So, imagine, you create an object of that abstract class and try to access one of the abstract methods, it will throw error because there is nothing inside the method to perform.

For object to run any method, it must have some body and abstract method does not have any.Hence, we cannot create the object of any abstract class.

Following error will be seen if one tries to create an object of an abstract class

Student –ERROR: **Class_name is abstract; cannot be instantiated.**

Class_name object_reference = new Class_name();

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

When we define class, data members, and member function without specifying any access modifier, it has a default access modifier by default. The class, data members, and member functions that we defined without using access modifiers can be accessed only within the same package. Outside of the current package, we cannot use the classes and methods.

The meaning of both the access specifiers and the access modifiers is the same. There are no differences between the specifiers and modifiers, and the use of both is the same. The access modifier is an official term and the new term that we use instead of modifier is specifier. So, default, public, protected, and private access modifiers can also be referred to as default, public, protected, and private access specifiers.

45. Which OOPS concept is used as reuse mechanism?

Inheritance is the object-oriented programming concept where an object is based on another object.

Inheritance is the mechanism of code reuse. The object that is getting inherited is called thesuperclass and the object that inherits the superclass is called a subclass.

46. Define the Benefits of Object-Oriented Programming?

OOP has a lot of benefits over procedural programming. Here are just a few:

OOP allows you to better represent the real world in your programs. Just about everything in the world can be described as a set of properties and actions, which is exactly how objects are organized. By better representing the world, OOP programs can better solve real problems. OOP programs are easier to read and understand. Because you don't have to write out the properties and actions for each character or sprite, your programs will be much shorter. This makes them easier to read and understand.

It can be faster to program in OOP. OOP makes it easy to reuse objects in other programs. Instead of creating code from scratch, you can use an existing object or method, and change it to fit your program.

It's easier to create large programs. Because OOP helps you eliminate unnecessary code, it's easierto create larger, more complex programs with OOP.

OOP programs are easier to modify and maintain. With OOP, it's easy to make changes to existing objects. You can also use existing objects to create new objects with minor modifications.

Because objects are self-contained, OOP programs are easier to debug.

OOP makes your programs easier to write, maintain, and reuse because of things like objects, classes, properties, and methods.

47. Explain method overloading?

Method overloading is a concept that allows to declare multiple methods with same name but different parameters in the same class.

Java supports method overloading and always occur in the same class (unlike method overriding).

Method overloading is one of the ways through which java supports polymorphism. Polymorphism is a concept of object-oriented programming that deal with multiple forms. We will cover polymorphism in separate topics later on.

Method overloading can be done by changing number of arguments or by changing the data typeof arguments.

If two or more method have same name and same parameter list but differs in return type cannot be overloaded.

Note: Overloaded method can have different access modifiers and it does not have any significancein method overloading.

There are two different ways of method overloading.

- Different datatype of arguments
- Different number of arguments

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

Early Binding	Late Binding
It is a compile-time process	It is a run-time process
The method definition and method call arelinked during the compile time.	The method definition and method call arelinked 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

50. Explain loose coupling and tight coupling?

Coupling is nothing but the dependency of one class on the other. If one object in a code uses the other object in the program, it is called loose coupling in Java. In coupling, two classes or objects collaborate and work with each other to complete a pre-defined task. It simply means that one element requires another element to complete a function.

1) Loose Coupling in Java:
   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.
   Here, the parent object is rarely using the object, and the object can be easily changed from external sources. The loose coupling in Java has the edge over tight coupling as it reduces code maintenance and efforts. A change in one class does not require changes in the other class, and two classes can work independently.
2) Tight Coupling:
   When two classes are highly dependent on each other, it is called tight coupling. It occurs when a class takes too many responsibilities or where a change in one class requires changes in the other class. In tight coupling, an object (parent object) creates another object (child object) for its usage. If the parent object knows more about how the child object was implemented, we can say that the parent and child object are tightly coupled.

49. Explain early binding? Give examples?

Binding generally refers to a mapping of one thing to another. In the context of compiled programming languages, binding is the association of a method call with the method definition. In simpler terms, when a method is called in Java, the program control binds to the memory addresswhere that method is defined.

- In early binding, the method definition and the method call are linked during the compile time. This happens when all information needed to call a method is available at the compiletime.
- The normal method calls and overloaded method calls are examples of early binding.
- The binding of private, static, and final methods happens at the compile-time as theycannot be overridden.
- Since all information needed to call a method is available before run time, early bindingresults in faster execution of a program.

```
// Java program to illustrate the concept of early and late
binding

class Vehicle
{
   void  start() {
      System.out.println("Vehicle Started");
   }

   static void stop() {
      System.out.println("Vehicle Stopped");
   }
}

class Car extends Vehicle
{
   //Override
   void start() {
      System.out.println("Car Started");
   }

   static void stop() {
      System.out.println("Car Stopped");
   }
}

class Main
{
   public static void main(String[] args)
   {

      Vehicle vehicle = new Car();

      vehicle.start();
      vehicle.stop();
   }
}
```

Output:

Car Started

Vehicle Stopped

Consider the given code:

It has two classes – the Vehicle class and the Car class, where the Car extends the Vehicle class. Each class has two methods – start() and stop(). The stop() method is static while the start() method is non-static. Now when we call the start() and stop() methods using parent class reference, the start() method of the child class is called while the stop() method of the parent class is called.

This is because stop() is a static method and hence cannot be overridden. So binding of the stop() method happens at the compile time itself (early binding). On the other hand, start() is a non-static method overridden in the child class. So, the information about the type of object is available at the run time only (late binding), and hence the start() method of the Car class is called.

51. Give an example among tight coupling and loose coupling.

```
// tight coupling concept

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 concept

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.

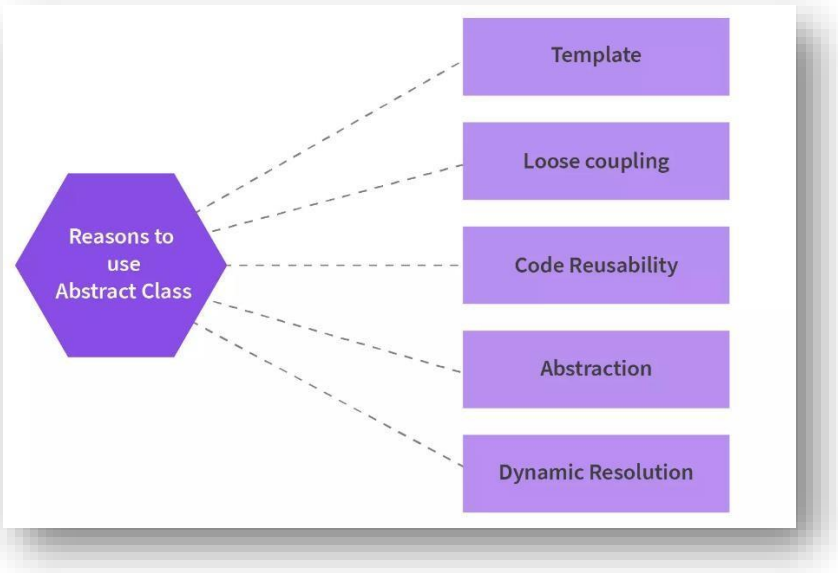
Hiding internal details and showing only the functionality is known as abstraction. Abstraction is one of the key concepts in Object Oriented Programming Paradigm.

In order to implement Abstraction, we use Abstract Classes or Interfaces. Abstract class in Java is a collection of abstract and non-abstract methods. Abstract methods do not have a body or implementation. The implementation for the abstract methods is provided by the child classes which extend the abstract class.

Basically, an Abstract Class is nothing but a blueprint for the child class. Abstract classesjustify the layout of your idea and does not really implement them.

Syntax:

```
abstract class ClassName{
        // Abstract and Non-abstract methods
}
```



53. Define the Benefits of oops over pop?

- OOP makes development and maintenance easier.
- OOP provides data hiding.
- OOP provide ability to simulate real-world event much more effectively.
- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the code DRY "Don't Repeat Yourself".
- OOP makes it possible to create full reusable applications with less code and shorterdevelopment time.
- Modularity for easier troubleshooting.
- OOP gives flexibility through polymorphism.
- OOP gives better productivity.

54. Explain Generalization and Specialization?

The process of converting a class type into another class type having the relationship between themthrough inheritance is called class casting in java.

If classes have some relationship between them through inheritance, it is also possible to converta class type into another class type through type casting.

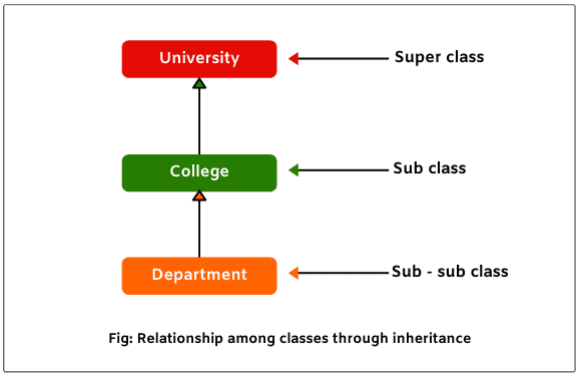


Fig: Relationship among classes through inheritance

since a College class is derived from University class through inheritance, we can convert a College class into a University class through type casting.

Similarly, we can also convert a Department class into College class, since Department is subclass of College class. These are the examples of class casting in java.

the figure which shows the classes Department, College, and University have a relationship among them through inheritance.

Generalization:

The process of converting subclass type into superclass type is called generalization in java. This is because we are making the subclass to become more general so that its scope can be more widening.
This conversion is also called widening or upcasting in referenced data types. Let's understand generalization by taking a real-time example.

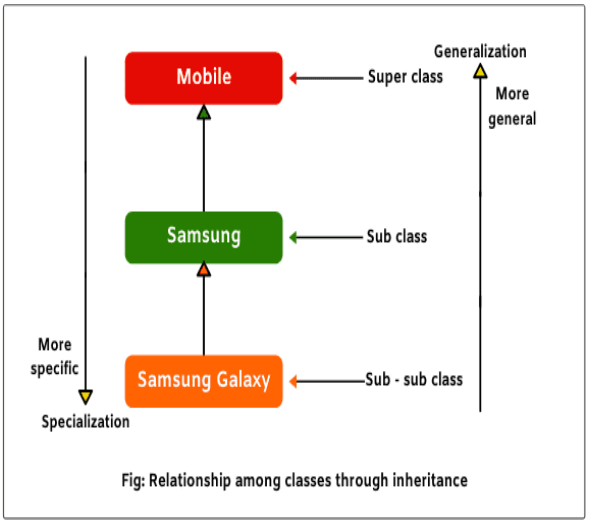


Fig: Relationship among classes through inheritance

When we talk about a mobile, In general, it may represent any kind of mobile. So, here, the scope is widened.

Now, suppose we talk about Samsung mobile, then we come down one step in the hierarchy of inheritance and we have eliminated any other kind of mobiles.

Thus, we are becoming more specific. When we still come down to Samsung Galaxy, we are pointing only Samsung Galaxy mobile and not any other Samsung mobile.

Thus, this is very specific. This means that when we move down from superclass to subclasses, we are becoming more and more specific.

When we go back from subclasses to superclass, we are becoming more general. This is calledgeneralization in java. Generalization is safe because classes will be more general.

Hence, we do not need a cast operator to perform generalization. Java compiler will do theimplicit casting in generalization.

Specialization

The conversion of a superclass type into subclass type is called specialization in java. Specialization means going down from a more general form to a more specific form. Thus, itsscope will be narrowed. Hence, this conversion is also called narrowing or down-casting in referenced data types.

Specialization is not safe because classes will be more and more specific. In this case, we willneed cast operator. Java compiler will do explicit casting in the specialization.

55. Write in brief Association, Aggregation and Composition?

Association in Java is a connection or relation between two separate classes that are set up through their objects. Association relationship indicates how objects know each other and howthey are using each other's functionality. It can be one-to-one, one-to-many, many-to-one and many-to-many.

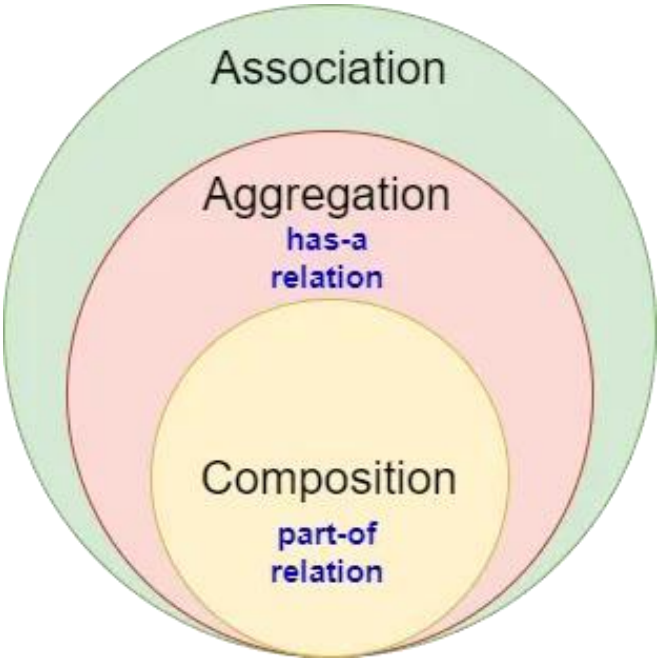
- For example, a person can have only one passport. That is a "one-to-one" relationship.
- If we talk about the association between a Bank and Employee, a bank can have manyemployees, So it is a "one-to-many" relationship.
- Similarly, every city exists in exactly one state, but a state can have many cities, which isa "many-to-one" relationship.
- Lastly, if we talk about the association between a teacher and a student, multiple studentscan be associated with a single teacher and a single student can also be associated with multiple teachers but both can be created or deleted independently. This is a "many-to- many" relationship.

Composition It is a "belongs-to" type of association. It simply means that one of the objects is alogically larger structure, which contains the other object. In other words, it's part or member ofthe larger object. Alternatively, it is often called a "has-a" relationship (as opposed to an "is-a" relationship, which is inheritance).

For example, a building has a room, or in other words, a room belongs to a building. Composition is a strong kind of "has-a" relationship because the objects' lifecycles are tied. It means that if we destroy the owner object, its members also will be destroyed with it. For example, if the building is destroyed the room is destroyed as well in our previous example. But,note that doesn't mean, that the containing object can't exist without any of its parts. For example, if we tear down all the rooms inside a building, the building will still exist.

Aggregation is also a "has-a" relationship, but what distinguishes it from composition, is that thelifecycles of the objects are not tied. Both the entries can survive individually which means ending one entity will not affect the other entity. Both of them can exist independently of each other. Therefore, it is often referred to as week association.

Let's take the example of a player and a team. A player who is a part of the team can exist even when the team ceases to exist. The main reason why you need Aggregation is to maintain code reusability.



Q 56. Write in brief Object Composition vs. Inheritance

Inheritance:

When we want to create a new class and there is already a class that includes some ofthe code that we want, we can derive our new class from the existing class. In doing this, we can reuse the fields and methods of the existing class without having to writethem ourself.

A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, butthe constructor of the superclass can be invoked from the subclass.

Types of Inheritance are:

1. Single inheritance
2. Multi-level inheritance
3. Multiple inheritances
4. Hybrid inheritance
5. Hierarchical inheritance

Composition:

The composition also provides code reusability but the difference here is we do notextend the class for this.

Example of Composition:

Let us take an example of the Library.

57. Explain cohesion ?

. 57Explain cohesion ?

Cohesion in Java is the Object-Oriented principle most closely associated with making sure that a class is designed with a single, well-focused

purpose. In object-oriented design, cohesion refers to how a single class is designed. Note: The more focused a class is, the more is the cohesiveness of that class.

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

White box reuse is normally done through inheritance and black box is done through composition.

White Box Reuse

Pro: You can customize the module to fit the specific situation, this allows reuse in more situations
Con: You now own the customized result, so it adds to your code complexity.

Black Box Reuse

Pro: Simplicity and Cleanliness
Con: Many times it is just not possible

59. Explain "this" KEYWORD

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).

60. Write in brief static member and member functions.

We can define class members static using static keyword. When we declare a member of a class as static it means no matter how many objects of the class are created, there is only one copy of the static member.

A static member is shared by all objects of the class. All static data is initialized to zero when the first object is created, if no other initialization is present. We can't put it in the class definition but it can be initialized outside the class as done in the following example by redeclaring the static variable, using the scope resolution operator :: to identify which class it belongs to.

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

Java, like many other OOP languages, allows you to implement multiple methods within the same class that use the same name. But, Java uses a different set of parameters called method overloading and represents a static form of polymorphism.

The parameter sets have to differ in at least one of the following three criteria:

- They need to have a different number of parameters, one method accepting 2 and another one accepting 3 parameters
- The types of the parameters need to be different, one method accepting a *String* and another one accepting a *Long*
- They need to expect the parameters in a different order. For example, one method accepts a *String* and a *Long* and another one accepts a *Long* and a *String*. This kind of overloading is not recommended because it makes the API difficult to understand

In most cases, each of these overloaded methods provides a different but very similar functionality.

Due to the different sets of parameters, each method has a different signature. That signature allows the compiler to identify which method to call and binds it to the method call. This approach is called static binding or static polymorphism.

62. Explain the Diamond problem?

In Java, the diamond problem is related to multiple inheritance. Sometimes it is also known as the deadly diamond problem or deadly diamond of death. In this section, we will learn what is the demand problem in Java and what is the solution to the diamond problem.

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 different interfaces. It allows us to implement these two interfaces, from a class. We must override the default methods explicitly with its interface name.

64. **Explain the need of abstract**

An abstract class is a class that has been defined using the abstract keyword. The abstract keyword is used to indicate that the class or method has an incomplete implementation. This means that another class, a subclass of the abstract class has to provide the missing implementation of the abstract class.

According to the C# reference, *'the abstract modifier can be used with classes, methods, properties, indexers, and events*'.

If abstract keyword is applied to a class, then the class cannot be instantiated. This means that you cannot directly create a new object of an abstract class. To use an abstract class, you need to create a subclass that derives the abstract class.

65. **Why can't we instantiate abstract**

We can't instantiate an abstract class because the motive of abstract class is to provide a common definition of base class that multiple derived classes can share.
Abstract class is complete Virtual class which contain only abstract members which have noting to do by itself so to avoid the situation we are restricted to instantiate abstract class but if its is need then create its subclass which allows you to create object or instantiate it.

66. **Can abstract class have constructors?**

Yes, abstract class have constructors in java. But it is not used to instantiate abstract class. It is used in constructor chaining or to initialize abstract class common variables.

67. **How many instances can be created for an abstract**

Constructor is always called by its class name in a class itself. A constructor is used to initialize an object not to build the object. As we all know abstract classes also do have a constructor. So if we do not define any constructor inside the abstract class then JVM (Java Virtual Machine) will give a default constructor to the abstract class. If we want to know how to define user define constructors like constructors with argument or any kind of constructor inside the abstract class then you should follow the given procedure.

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

super keyword for Method Overloading in Java.

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

the default access specifier is package.Classes can access the members of other classes in the same package.but outside the package it appears as private

70. Define all the operators that cannot be overloaded

In C++ we can overload some operators like +, -, [], -> etc. But we cannot overload any operators in it. Some of the operators cannot be overloaded. These operators are like below

These operators cannot be overloaded because if we overload them it will make serious programming issues.

For an example the sizeof operator returns the size of the object or datatype as an operand. This is evaluated by the compiler. It cannot be evaluated during runtime. So we cannot overload it.

71. Explain the difference among structure and a class?

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.

75. Do We Require Parameter For Constructors?

To create a parameterized constructor, it is needed to just add parameters as a value to the object as the way we pass a value to a function. Somewhat similar scenario we do by passing the parametrized values to the object created with the class. Parameters are used to initialize the objects which are defined in the constructor's body.

72. 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:

1. Default – No keyword required
2. Private
3. Protected
4. Public

- Default: 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.

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

Types of Java constructors

There are two types of constructors in Java:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

A constructor is called "Default Constructor" when it doesn't have any parameter.

A constructor which has a specific number of parameters is called a parameterized constructor.

Why use the parameterized constructor?

The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.

74. Explain a ternary operator

The ternary operator take three arguments:

1. The first is a comparison argument
2. The second is the result upon a true comparison
3. The third is the result upon a false comparison
4. It helps to think of the ternary operator as a shorthand way or writing an if-else statement.

76. 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.

77. 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.

78. 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.

79. Define The Various Types Of Constructors

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

80. 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 (<<) and extraction (>>) 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.

81. Can you give some examples of tokens?

```
public class Demo
{
    public static void main(String args[])
    {
        System.out.println("Hello");
    }
}
```

In the above code snippet, **public, class, Demo, {, static, void, main, (, String, args,[,],), System, ., out, println, Hello**, etc. are the Java tokens.

The Java compiler translates these tokens into Java bytecode. Further, these bytecodesare executed inside the interpreted Java environment.

82. 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'tsupport 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#.....

The structured program mainly consists of three types of elements:

- Selection Statements
- Sequence Statements
- Iteration Statements

The structured program consists of well structured and separated modules. But the entry and exit in a Structured program is a single-time event. It means that the program uses single-entry and single-exit elements. Therefore a structured programis well maintained, neat and clean program. This is the reason why the Structured Programming Approach is well accepted in the programming world.

Disadvantages of Structured Programming Approach:

1. Since it is Machine-Independent, So it takes time to convert into machine code.
2. The converted machine code is not the same as for assembly language.
3. The program depends upon changeable factors like data-types. Therefore it needs to be updated with the need on the go.
4. 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.

83. When to use interface over abstract class.

- If the functionality you are creating will be useful across a wide range of disparate objects, use an interface. Abstract classes should be used primarily forobjects that are closely related, whereas interfaces are best suited for providing common functionality to unrelated classes.
- If you are designing small, concise bits of functionality, use interfaces. If you aredesigning large functional units, use an abstract class.

84. 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 **constructor** 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. In this section, we will discuss **private constructors in Java, rules for creating a private constructor, and its use cases.** Also, we will see its implementation.

Java allows us to declare a constructor as private. We can declare a constructor private by using the **private** access specifier. Note that if a constructor is declared private, we are not able to create an object of the class. Instead, we can use this private constructorin **Singleton Design Pattern**.

Use Cases of Private Constructor

The main purpose of using a private constructor is **to restrict object creation**. We also use private constructors to implement the singleton design pattern. The use-cases of theprivate constructor are as follows:

- It can be used with static members-only classes.
- It can be used with static utility or constant classes.
- It can also be used to create singleton classes.
- It can be used to assign a name, for instance, creation by utilizing factorymethods.
- It is also used to avoid sub-classing.
- It incorporates the factory methods.

85. 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 tothe class in which they are declared.

The following rules keep in mind while dealing with private constructors.

- It does not allow a class to be sub-classed.
- It does not allow to create an object outside the class.
- If a class has a private constructor and when we try to extend the class, acompile-time error occurs.
- We cannot access a private constructor from any other class.
- If all the constant methods are there in our class, we can use a private constructor.
- If all the methods are static then we can use a private constructor.
- We can use a public function to call the private constructor if an object is notinitialized.
- We can return only the instance of that object if an object is already initialized.

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

Use 'private' access specified to prevent a method from being overridden in thederived class.

87. Why can't you specify accessibility modifiers for methods inside interface?

- All methods declared inside interfaces are implicitly public and abstract, evenif we don't use public or abstract keyword.
- Methods inside interface must not be static, final, native or strictfp.
- However, from Java 8, interface can have default and static methods and fromJava 9, it can have private methods as well.

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

The error **non static variable cannot be referenced from a static context in Java** is mostly faced by the beginners at the time of compilation of Java program. The reason to occur this error is that they use a non-static member variable in the main() method. Because the main() method in Java is a static method and it is invoked automatically, we need not to create an object to invoke it.

89. Define different ways a method can be overloaded?

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 increasesthe 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 otherprogrammers to understand the behavior of the method because its name differs.

Benefits of using Method Overloading

- Method overloading increases the readability of the program.
- This provides flexibility to programmers so that they can call the same method fordifferent types of data.
- This makes the code look clean.
- This reduces the execution time because the binding is done in compilation time itself.
- Method overloading minimizes the complexity of the code.
- With this, we can use the code again, which saves memory.

There are two ways to overload the method in java

- By changing number of arguments
- By changing the data type

1. By changing number of arguments

```
class SumOverloding
{
 //adding two integer numbers
 int add(int a, int b)
 {
  int sum = a+b;
  return sum;
 }
 //adding three integer numbers
 int add(int a, int b, int c)
 {
  int sum = a+b+c;
  return sum;
 }
}
class MethodOverloading
{
 public static void main(String args[])
 {
  SumOverloding obj = new SumOverloding();
  System.out.println(obj.add(10, 20));
  System.out.println(obj.add(10, 20, 30));
 }
}

OUTPUT:
30
60
```

2. By changing the data type

```
class SumOverloding
{
 //two int parameters
 public int add(int a, int b)
 {
  int sum = a+b;
  return sum;
 }
 //two float parameters
 public float add(float a, float b)
 {
  float sum = a+b;
  return sum;
 }
}

class MethodOverloadingDeiffDataType
{
 public static void main(String args[])
 {
  SumOverloding obj = new SumOverloding();
  //This will call the method add with two int params
  System.out.println(obj.add(5, 15));

  //This will call the method add with two float params
  System.out.println(obj.add(5.5f, 2.5f));
 }
}
OUTPUT:
20
8.0
```

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

An abstract class is a class that is declared abstract —**it may or may not include abstract methods**. 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

91. Explain the default access modifier of a class?

default: **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

92. **Can function overriding be explained in same** class?While overriding −

- Both methods should be in two different classes and, these classes must be inan inheritance relation.
- Both methods must have the same name, same parameters and, same returntype else they both will be treated as different methods.
- The method in the child class must not have higher access restrictions than theone 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 canjust overload them.

93. Does function overloading depends on Return Type?

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: ifthere are two functions: int sum() and float sum(), these two will generate a compile- time error as function overloading is not possible here.

94. Can abstract class have a constructor?

We can declare constructors of abstract classes. The Java Virtual Machine (JVM) isresponsible to generate a default constructor for every class in Java Program if thereare no User Defined Constructor present. This same applies to Abstract classes as well.

You can also create a user-defined parameterized constructor for your abstract class aswell. Since an abstract classes are meant to be inherited from, the child classes can instantiate the parent class as per its need using the super() keyword.

95. Define rules of Function overloading and function overriding?

Function Signature: **Overloaded functions must differ in function signature i.e. either number of parameters or type of parameters should differ. In overriding,function signatures must be same**. Scope of functions: Overridden functions are in different scopes; whereas overloaded functions are in same scope.