OOPS Concepts:-

1. Encapsulation :- Wrapping data members & its functions in to a single unit. Those are called with object of a class.

Creating a Class with Private variables & these Private Variables can be Accessed using Public setters & getters methods

Setter methods are used to set values for private variables(Class Level) of a class.

Setter method Return Type will be always void.

Getter Method return type will be always the return type of its variable

Q)What are OOPS Concepts? How did u achieved in u r project?

A) in java we have 4 types of oops concepts

Q) What is encapsulation? How did u achieved in u r project?

Encapsulation :- Wrapping data members & its functions in to a single unit. We achieve in java using Class. In class we have methods and variables, these are referred by object of a class.

1. Abstraction :-

Hiding the Implementation of the functionality. It Tells how it is works & but it doesn’t tell how it was Implemented.

Ex;- clicking on login button(we don’t know how it is validating the Authentication)

Ex:- clicking on Submit button(we don’t know how all the data is saved & submitted in the backend)

This is Achieved by Interfaces & Abstract Classes.

100% Abstraction can be Achieved with Interface

100% Abstraction cannot be achieved with Abstract Class because it might/can contain defined methods because of which 100% Abstraction is not possible

|  |  |
| --- | --- |
| Abstract Class | Interface |
| It Contains Abstract Methods & Defined Methods & Constants | It contains only Abstract Methods & Constants(public static final) |
| Abstract Keyword has to applied at class level & Method Level for Abstract methods | Abstract Keyword is optional at Interface level & method level because JVM will anyways provide . |
| Abstract class can contain Constructor but it will be used by its subclasses | Interface cannot contain Constructor because it will give compilation error if we add Constructor |
| Class &Abstract Class will not participate in Multiple Inheritance | Interface can be used in multiple Inheritance. |
|  |  |
|  |  |
|  |  |

Similarity

object Cannot be created for Both Abstract class & Interface.

Abstract class objects can be create by using its sub classes.

Interface methods are accessed by using its sub class objects.

Q) What is abstraction? How did u achieved in u r project?

Hiding the Implementation of the functionality. It Tells how it is works & but it doesn’t tell how it was Implemented. We use interfaces and abstract classes with abstract methods . by looking at this abstract methods we don’t know how it was implemented, because these methods are overridden in it’s sub classes.

Q) What is difference b/w interfaces and abstract classes?

Refer the above table.

Q) how can u solve diamond problem in inheritance?

Using interfaces. see the hybrid inheritance in diagram .

Inheritance :- Inheritang the Properties and behaviours(variables and methods) from Parent to its children

1. InHeritance :- Inheriting the variables& methods from SuperClass to its SubClasses(Parent to Child)

Private methods & private variables cannot be accessed/Visible in SubClasses.

Inheritance can be achieved through **extends** keyword between classes(A,B)& Interfaces(I1,I2)

Ex:- **Class A extends B**

Interface I1 extends I2

Class A can Implement Interface I1

Ex:- Class A Implements I1

Interface I1 cannot Implement or Extend Class A

Interface I1 implements class A (Not Possible)

**Single Inheritance:-**

We can Create objects of SuperClass with SuperClass Reference so that we can access all the Public methods in SuperClass.

We can Create objects of SubClass with SubClass Reference so that we can access all the Public methods in SubClass & SuperClass

If we Create objects of SubClass with SuperClass Reference then we can access all the Public methods of SuperClass But we will not be able to access methods from Sub Class.

**Multiple inheritance**  in the below diagram A and B are interfaces C is sub class of A and B.

**Note**: If a class (C) implements an interface(I) it has to override all the methods of interface.

if we don’t override any method of the interface It will gives compilation error.

If an abstract class (C) implements an interface(I) overriding the methods of interface is optional.

**Hybrid inheritance** is the combination of more than one inheritances

In Hybrid Inheritance, A,B,C should be Interfaces because if A is Class B & C cannot implement class A.

If a Class A extends ClassB and implements Interfaces I1,I2 we should write the syntax as below:-

Class A extends B implements I1,I2

Class A implements I1 extends B ------not possible throws compilation error

Q) What is inheritance? How did u achieved in u r project?

Inheriting the variables& methods from SuperClass to its SubClasses(Parent to Child)

In our project generally we extends some abstract classes or interfaces



4.Polymorphism :-

Same Action Can be performed in multiple ways

We can achieve this feature in java using methods

1. **Compile time Polymorphism** :-polymorphism which occurs at compile time is called Compile time Polymorphism .This can be Achieved by Method Over Loading Concept.

Method Over Loading Concept :- It is performed in a Class

Methods Name are Same but Signature should be different.

Method Signature:- Number of arguments, data types of Arguments, order of the Arguments

If two methods have same name with same signature, Access Specifiers & return types doesn’t make a difference.

1. **Run time Polymorphism**:- polymorphism which occurs at run time is called run time Polymorphism . This can be Achieved by Method Over Riding Concept.

Method Over Riding:- It can be Achieved using Inheritance concept

Methods Name are Same & Signature are also Same . Along with these method return types & Access Specifies should be also the same.

Q) What is Polymorphism ? How did u achieved in u r project?

A) Same Action Can be performed in multiple ways. we have 2 types polymorphism

In our project we have overloaded methods and overridden methods

**Constructor**:-

Constructor is used for Allocating Memory for the Class.

Rules for Constructor:-

1. Constructor name should be exactly same as ClassName(CaseSensitive)
2. Constructor will not have return type(like void, int, String)

Default Constructor( 0 Arguments Constructor) is optional which is provided by JVM

Class can Have Default Constructor & OverLoaded Constructors.

For OverLoaded Constructor, Signature should be different.(Number of arguments, data types of Arguments, order of the Arguments)

In a class if we override overloaded constructor(1 or 2 arg constructor) and if we try to call default constructor(creating object with default constructor) it will give compilation error because jvm will not provide default constructor in this case

We can add return statement inside a constructor it does not give compile time or runtime error

Class A

A(){

………………….

return;

}

***Static* :-** Static is a Keyword which is used at Variable level & method level but not at the Class Level. Which means class can contain static variables ,Static methods & static blocks also.

JVM Allocates memory for static variables Static methods & static blocks at compile time so that we can access these without creating objects(using Class name, we can access these.)

If we try to access these with objects name it does not give any compile time or run time error. It just gives Warning.

Static methods cannot be Overridden in its subclasses because the methods are at class level but they can be accessed in Subclasses.

A Class can contain instance variables, static variables ,constants, instance methods, static methods, constructors, static blocks & instance blocks.

If a Class contains all the above details then it will be executed in the following order:-

1. All the Static Blocks in the order
2. When we create a object of the class all the instance blocks in the order are executed first
3. If constructor is overridden in class that will be executed.
4. Methods will be executed if they are called.

Abstract class can contain static method , abstract method ,define method (instance method).

If we give static for an abstract method it will throw compilation error. Because abstract method memory is allocated at runtime but static method memory loads at compile time.

**Final** is a Keyword which is used at Variable level & method level & Class Level . In a Class if a variable is declared as Final the value will not be changed once it is assigned.

We can not assign final variable multiple times as below

Class A {

Final String s = null;//the value is assigned as null,can not be changed any where.

Final int I;//this is compilation error

Final Methods cannot be overridden in its subclasses but they can be accessed in Subclasses.

Final Classes cannot participate in inheritance that means we cannot extend final class.

**this :** this isa keyword which is used to refer current class object . we can access current class methods and variables using **this**. This is used at variable level and method level and **this()** will be treated as default constructor.

Generally **this()** is used as first statement in a constructor otherwise it will give compile time error.

**super** super is a keyword which is used to refer super class object. we can access super class methods and variables using **super.**

Generally **super()** is used as first statement in a constructor otherwise it will give compile time error

**Note** :in constructor we can add either this() or super() but not both.

**Access specifiers:** we have4 types of access specifiers in java which is used at variable level method level ,constructor level and class level

**Private**: access only inside the class

**Defau**lt: if we don’t give any access specifier it will be treated as default. This is package level specifier that means we can access any variable or method of a class p1.A(p1 is package ) in any class inside the package p1

**Protected**: we can access a variable of a class p1.A(p1 is package ) in any of the class inside the package p1 , any of its sub packages class(p1.p2.B)and any other package classes(p3.C) with extends keyword. that means class(B or C) has to extend the protected Variable Class(A).

If we access protected variables in outside of the class with object name it will give compilation error.

**Public:** we can access public variables or methods in any other class of any package

**Wrapper Classes:** in java we have primitive data types and wrapper Classes also .java supports user defined variables .

We can convert primitive data types to wrapper classes and vice versa

Using wrapper classes we can achieve 100% object orientation in java.

Examples: int ,char , boolean ,float etc.. are primitives

Integer,Character ,Boolean,Float etc….are WrapperClasses

Class A{}

A a1;// here a1 is user defined variable

By default primitive data types contain some default value

Int (0) , boolean (false) ,float (0.0f)

By default all wrapper class, String and user defined class values are “null”.

**Note:** in java Class if we want to exit at any time we use **System.exit(0).** When code execute this statement control will come out of jvm.

**Autoboxing and Unboxing:**

The automatic conversion of primitive data types(eg int) into its equivalent Wrapper type (eg Integer) is known as boxing and opposite operation is Unboxing.

**Conditional Clauses:**

1. If
2. If-else
3. Nested- if-else(if-elseif-elseif…..else)
4. Switch

If: if is a conditional statement it executes when the condition is true.

In if condition it is optional to put “{ }” if we have only one statement. Otherwise we have to put” { }” to execute the block of statements.

If a method has only if conditions all the if blocks will be checked even if condition is satisfied at the beginning. but in nested if else, blocks will be checked until the condition is true. It means once he condition is true remaining conditions in else if or else block will not be executed.

**Break:** generallybreak is used inside the loops or conditional statements. Once break is executed, control will come out of the loop.

For(){

For(){

If(condition){

Break;

}

}

Block of statements

}

In the above syntax, once if block is executed, control will come out of inner for loop, but outer for loop will be executed as it is.

**Return :** generally return is used in side methods. it is used for returning a value from the method. it means once return statement is executed in side a method at any place, control will come out of the method.

Void m1(){

for(){

If(condition){

Return;

}

}

//some stmt to be executed

}

In the above syntax when if condition satisfied, return statement will be executed. Control will come out of the method.

**Loops :** if we want to execute some statements repeatedly we use loops.

1. **while** : in while loop statement will be executed inside the loop until the condition is false

syntax:

< initialization of variable>

While(<condition>){

Statements to be executed

<increment / decrement of variable >

Statements to be executed

}

1. **do while:** in do while the loop will be executed once without checking condition. The condition will be checked at the end of first execution. it means loop will be executed from second iteration only if the condition in while() is true.

Syntax

< initialization of variable>

Do{

Statements to be executed

<increment / decrement of variable >

Statements to be executed

}while(<condition>)**;**

Deference b/w while and do while

In while loop initially it checks the condition. if condition is true then only while loop is executed.

In do while loop initially loop will be executed once without checking condition.

note : **” ;”** ismandatory in do while loop.

1. For loop:

in formal for loop initialization , condition ,increment /decrement will be present in one line.

Syntax:

For(initialization ; condition ;increment /decrement){

Statements to be executed

}

Or

<initialization>

For(;condition ;){

Statements to be executed

<increment / decrement of variable >

Statements to be executed

}

For loop execution

In for loop,

Step1 : variable will be initialized,

Step 2 :condition will be checked.

Step 3: If condition is true, loop will be executed.

Step 4 : After loop is executed, increment/decrement operation performed.

Step 5: in step4 operation, variable holds the increment/decrement value. With this value step2,step3,step4 will be performed until step 2 fails(condition is false)

**For-each loop:** generally for-each loop is used to execute array or collection of values.

Syntax:

Int a[]={1,2,3,4};

For(int I:a){

Syso(i) ;

}

In the above syntax, “i” refers each element in the array(a).

**Java.util.Scanner:** it is a class which is used to read all types of values from console at runtime.

**Exception Handling:** in javawe have 2 types of exceptions

1. Java.lang.Error : we cannot handle this exceptions in java.
2. Java.lang.Exception: we can handle this exceptions in java.

For all the exceptions java.lang.Throwable is the super class

Java.lang.Exception is categorized into 2 types

1. Compiletime errors(java.lang.Exception).
2. Runtime errors(java.lang.RuntimeException)

Here RuntimeException is the subclass of java.lang.Exception





