DATE 31/1/2022
PAGE 1

1) What is OOP?

Ans OOP Stands for Object Oriented Programming it Represent a programming Methodology based on object.

2) what is Encapsulation ?

Ans Gathering all related attributes and Methods with in a Class is Called Encapsulation.

3) What is Expert Class ?

Ans Encapsulated Class is Called Expert Class.

4) What is Expert object ?

Ans Object Containing Related attributes and method is Called Expert object.

5) What is Polymorphism ?

Ans Polymorphism is the ability of an object to behave differently in different State. The most Common Use of polymorphism in OOP occurs when an object of parent class holds the Reference of child Class object.

- Polymorphism Can be achieved by
1) Method Overloading.
2) Method Overriding.
4) Interface Implementation.

Method Overloading :-

In a Class, it is Posible to define two or more Method of Same name but having a different argument This Concept is Called method overloading. it is Used for Static Polymorphism. It is also Know as early binding. Class Constructor Can also be Over-loaded.

Exp :-

```
Public class MethodOverload
{
    Public Void Show (int num1)
    {
        SOP ("number 1:" + num1);
    }
    Public Void Show (int num1, int num2)
    {
        SOP ("number1 :" + num1 + "number2:"
            + num2);
    }
```

```
Public Static Void main (String[] args) {

    MethodOverload mo = new MethodOverload ();

        mo.show (3);
        mo.show (4.5);
                                    }
```

Method Overriding :-

If chid Class has a Same Method as declared in Parent Class then it is Called Method Overriding. It is Used for dynamic Polymorphysm. It is also known as late binding.

exmp :-

```
Public Class Animal
    {
        Public Void Move ()
            {
                SOP ("Animal Can Move");
            }
    }

Public class Dog extends Animal
    {
        Public Void Move ()
            {
                SOP ("Dog Can Move");
```

```
Public class MOTest {

    Public static Void main (String[] args)
    {
        Animal a = New Animal ();
        Animal b = New Dog ();

        a.move ();
        b.move ();
    }
}
```

Interface :-

Interface Contains all the abstract method it defined with Interface Keyword and It is used achieved 100% abstraction in Java.

There are two type of Polymorphysm :-

1) Compile Time Polymorphism.

2) Run Time Polymorphism.

Compile-Time Polymorphism :-

Compile Time Polymorphism is a Static Polymorphism is achieved by Method Overloading.

It is a process in which a Call an Overloading method is Resolved at Compile time.

Run-time Polymorphism :

Run-Time Polymorphism is a Dynamic Polymorphism is achieved by Method Overriding. It is a Process in which a Call an Overriding method is Resolved at Run-time.

6) Key Concept of OOP ?

Ans There are three Key Concept of OOP

1) Encapsulation : Creates Expert classes.

2) Inheritance : Creates Specialized classes

inheritance is a Mechanism where a new Class is derived from an existing class.

inheritance is Used to provide Specialized behavior. It makes Relationship between Parent class and child class is Reffered as Is-A Relationship.

3) Polymorphism : Provides dynamic behavior.

7) What is difference b/w Object and Instance?

Ans You Can use encapsulated class to declare an object just like declare a variable of primitive data type

```
Shape S = null;  Declare object S
```

You can allocate meomery to member attribute using Keyword new. Allocating meomery to member attributes is called instantiation and after meomery allocation object can be Referred as instance

```
Shape S = null;  // Declaration
S = new Shape();  // Instantiation
```

Member attributes : Variable / Fields Contained By Class are Cto Called "Member attributes" and function Contained by a class are Called "members methods". member attributes and member ~~function~~ method both are Collectively Called "members of classes".

→ If you Can ct call S as a object in statement 1#
→ After meomery allocation Same S can be refferred as instance in statement #2

8) Types of Inheritance ?

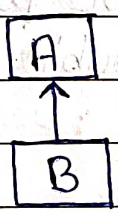
Ans Java Support following type of Inheritance

1) Single Inheritance.

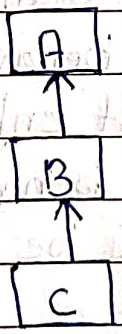
2) Multilevel Inheritance.

3) Hierarchical Inheritance.

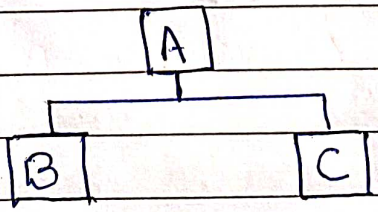
Single Inheritance



Multilevel Inheritance



Hierarchical Inheritance



9) What is Type Propogation.?

Ans When Calling a Method, One type of Parameter is Promoted to another implicitly if no Matching data type is found.

-> byte can be promoted to short, int, long, float or double

-> short can be promoted to int, long, float or double

-> char data type can be promoted to int, ~~fto~~ long, float or double and so on.

10) What is Abstract class ?

Ans Abstract class is Used, when Parent class need to provide their default behavior along with abstract method and it enforces child class to implements Specialized behavior of abstract method. It can never be instantiated.

Abstract class is declared by abstract Keyword.

11) What is Data abstraction ?

Ans Data abstraction is a process to hiding implimentation details and showing only functionality.

Data abstraction in Java is achieved by interfaces and abstract classes.

12) What is abstract Method ?

Ans Method that is declared without any body within an abstract class is called abstract Method it has only Signature not body. The body of method is defined by child classes.

13) What is Data hiding ?

Ans Data hiding is an aspect of OOP that allows developer to protect private data and implimetation details.

14) What is Marker Interface ?

Ans Interface that does not containg any attributes or Methods is called marker interface

1) Serializable (Java.io). 2) cloneable (Java.lang).
3) Remote, (Java.rmi).

DATE
PAGE 16

15) What is Constructor ? types of Constructor

Ans A Constructor is an instance method that have a same name as the class name. Constructor are used to initialize fields of an Object. Constructor do not have any return type.

Types of Constructor. ?

1) no Argument Constructor Like Default Constructor

```
Public class Person
{
Public Person ()
{ }
}
```

2) Parameterized Constructor

```
Public class Person {
String name ;
Public Person (String name)
{
Super ();
this. name = name
}
}
```

16) What is Constructor Overloading

Ans One class May have Multiple Constructor Called Constructor Overloading.

17) What is this and Super Keykord ?

Ans The this Keyword refers to the Current object in a method or Constructor

Super Keyword is Used to access method of the parent class.

18) What is Super() and this() in Java ?

Ans - Super() is Use to Call Base class (Parent class) Constructor.

- this() is Use to Call Current class Constructor.

19) Method finalize() ?

Ans Method finalize is defined in Object class. It is called by the garbage Collector when an object is Removed from meomery.

19) What is Specialization ? No Genralization ?

Ans - Specialization :- is Top down approach. in it splits a entity into Multiple New entity and in Specialization size of Schema is increases

- Genralization :- is Bottom-Up approach. multiple entity are Combined to form a new entity.

20) What is UpCasting and DownCasting ?

Ans UpCasting is a Casting from child class to parent class. In Java UpCasting is implicit.

exmp :- class A {M} class B extends A {M}
class C extends B {M}

```
class Test {
    main(String[] args) {
        A obj = new C();
        obj.m //(Print C)
    }
}
```

→ DownCasting : is a Casting from Parent Class to child class. It is not allowed in Java but we can Call it forcefully

exp :- class A {m}
class B extends A {m}
class C extends B {m}

class Test {
main (String [] args) {

B objB = (B) obj ;
objB.m // method calling of C.

2) what is object cloning ? any types ?

Ans The object cloning is a way to Create exact Copy of Content of an Object. Java Use clone() of Object class to clone an object. It is also Known as shallow Copy.

Types of Cloning :-

1) Shallow cloning - Shallow clonning, the object is Copied without its Contained or Sub objects. Shallow clonnig is only Copies the Top lever structure of an object. It Copies only ~~Key~~ References of an Sub object.

It is default in Java.

exmple :-

```
Public class BankAccount extends implement
    cloneable {
double balance = 0;
So Public BankAccount (double balance){
        Super();
        this.balance = balance;
}

Public Object clone throws CloneNotSupportedEx
        ception {
        return Super.clone();
        }

Public static Void main (String[] args) throws
    cloneNotSupported Exception {

BankAccount B1 = New BankAccount (1000)
BankAccount B2 = (BankAccount) B1.clone()

B2.balance = 2000;
SOP(B1.balance); SOP(B2.balance);

}
}
```

2) Deep cloning :- In Deep cloning, Complete Copy of duplicate Copy of the Original object is Created. it Copies Not Only primitive Values but also Copies all it Sub objects as well.

example :-

```
Public class BankAccount impliments Cloneable
    {
        double balance = 0;

    Public BankAccount (double balance)
        {
            Super();
            this.balance = balance;
        }
    Public Object clone() throws CloneNotSupExp
        {
            return Super.clone();
        }
    Public Void SetBalance (double balance)
        {
            this.balance = balance;
        }
    Public String toString () {
    return "BankAccount[balance="+balance+"]";
    }
```

```
Public class Customer implement Cloneable
{
    String name = null;
    BankAccount account = null;

Public Customer (String name)
{
    Super();
    this.name = name;
    account= New BankAccount(1000);
}

Public Object clone() CloneNotSupExp {

    Customer c=(Customer)Super.clone();
    c.account =(BankAccount)account.clone();
    Return c;
}

Public class Test {

    main(String[] args) throws CloneNotSupExp {

    Customer C = New Customer("ansh");
    Customer D = (Customer) C.clone();

    D.name = "shubh";
    D.account.SetBalance(2000);

    SOP (C.name+" "+C.balance);
    SOP (D.name+" "+D.balance); } }
```

22) Write a Program of One to One relationship

```
Public class Person {

    String name ;
    String PId ;

    Public Person (String name, String PId)
        {
            Super() ;
            this.name = name
            this.PId = PId ;
        }

Public class Passport extends Person {

    Public Passport (String name, String PId)
        {
            Super (name, PId) ;
        }

Public class Test {

    main (string[] args) {

        Passport p = new Passport ("ram", "101");

        SOP (P.name) //
        SOP (P.PId) //
                }
                }
```

23) Write a Program of One to Many Relationship?

Ans

```
Public class Books {

    String title ;
    String author ;

    Public Books (String title, String author)
    {
        Super () ;
        this.title = title
        this.author = author
    }

Public class Library {

    List<Books> book ;

    Public Library (List<Books book) {

        Super () ;
        this.book = book ;
    }

Public List<Books> getBook () {

    return book ;

    }
```

```
Public class Test {

    main (String[] args) {

        Books b1 = new Books ("Java", "ansh");
        Books b2 = new Books ("Python", "Shubh");

        List<Books> book = New ArrayList<> ();
        book.add (b1);
        book.add (b2);

        Library Lib = New Library (book);
        List <Books> bks = Lib.getBook;

        for (Books bk : bks)
            {
                Sop (bk.title + " " + bk.author);
            }
    }
}
```

24 How we Use Polymorphism ?

By Using Arrays, Return type, method argument.