Java

Java: It is a programming language purely object oriented language. It is an open-source and secured, Robust.

* Gossling Basic goals of Java:
1. simple, object oriented familier
2. Robust and secure
3. Architecture neutral portable
4. execute with high-performance
5. interpreted and dynamics - (threading)

It is also called two state language - compiled interpreted both.

* why is Java important?
1. Trouble with C & C++.
2. They are not portable and platform independent.
3. emergency of worldwide web which demanded portable language.

Java Development kit (JDK)
Java Runtime Environment (JRE)
Java virtual machine (JVM)

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

Introduction to Java

Java is a popular high level language which was developed by and released in 1995 currently owned by oracle and more than 3 billion devices run Java.. Java runs on variety of platform such as windows, macos and various version of unix. Java is used to develop various types of software application desktop app, mobile apps, web apps & home it is a general purpose programming language, write once run anywhere. This means compiled Java code can run on all platforms that supported Java without need to compile.

History of Java: Green team → Java
Development of Java → 1991
official release → 1995
Java gain popularity on web - late 90's
Introduction to Java 2 platform extended addition → early
Released Java as open source language → 2006
oracle aquire sun microsystem and becomes of Java
2010
introduction of Java 2 which significant language enhancement
- 2014
Release of Java 9 with modularity features 2017
Java 10 - 2018
Java 22 or JDK 22 → 19 march 2024

Features of Java (Java Buzz world)

(i) Simple and Easy to learn: Java syntax is based on C++, so it is very easy to learn, syntax is simple and easy to learn or understand. Java removed pointers and operator overloading that makes Java more easier than other programming language.

(ii) Platform independent: Java Application can run on any platform with Java byte code and there is only one condition JVM is installed in that system.

(iii) object-oriented: Java is totally based on object-oriented programming principles, principles like inheritance, encapsulation, Polymorphism and abstraction.

(iv) Automatic Memory management: Java's garbage collection automatically manages memory by freeing up unused objects.

(v) Multithreading: Java has built in support for multi-threading, allowing concurrent execution of multiple threads.

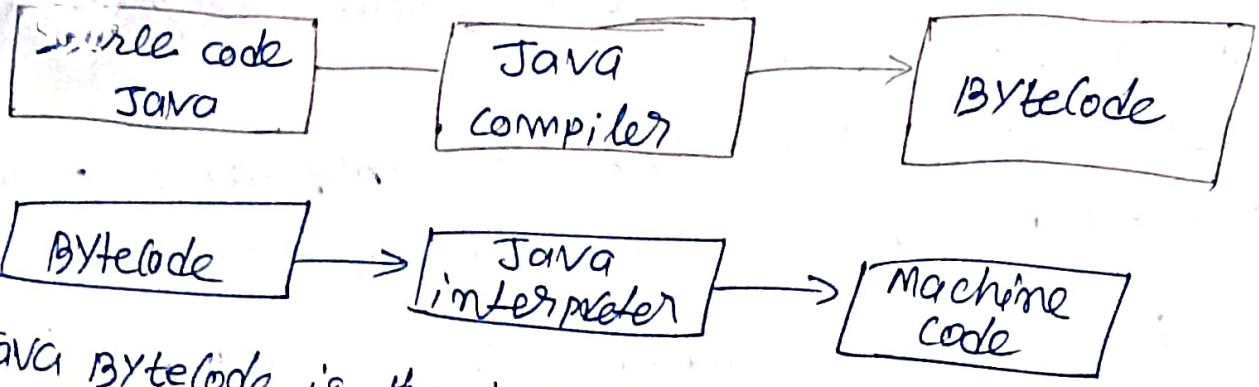
(vi) Secure: Java provides several security features such as bytecode varification, garbage collection and there no extra overhead for pointers that makes Java more secured language than other languages.

* what is Byte code in java?

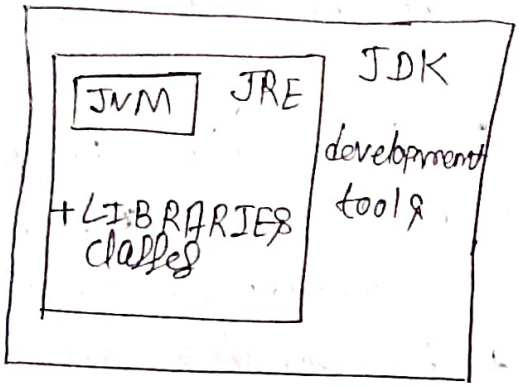
ans=> Byte code is intermediate representation of java code that is generated by java compiler, it is platform independent. it is low level set of instructions that is executed by Java virtual machine. Byte code can be run on any other platform provided JVM is there.

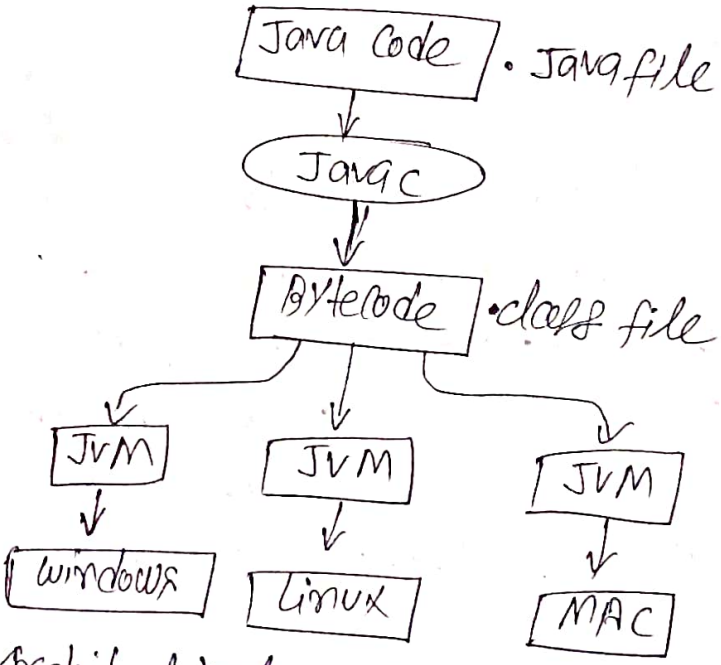
* what is Bytecode ?

ans:-



Java Bytecode is the instruction set of Java virtual machine the language to which Java and other JVM compatible source code is combined, each instruction is represented by single byte hence named bytecode making it a compact form of data.





Architectural Neutral Diagram -

Java Development Kit (JDK) - that provides environment to a developer and executes a Java program. It includes two things so that Development and JRE (Java runtime environment)

JRE: Java Runtime is an installation package that provide enviornment to only run Java program on to your machine.

JRE are only used by those who only wont to run Java program that are end users of your system.

JVM: Java virtual machine is a very important part of both JDK & JRE because it is combined & inbuilt in both.

whatever Java program you run using JRE & JDK goes into JVM and JVM is responsible for executing Java program line by line. Hence it is also called interpreter.

features of Java (Buzz word): Simple: Java syntax is based on C++ it is very easy to learn, syntax is simple and easy to understand. It has removed many compilation and rarely used features like pointer and operator overloading. There is no need to remove unreferenced object because there is a concept of automatically garbage collection.

(3) platform independent:

(4) secure & simple:

(5) Robust → It uses strong memory management. There is lack of pointer that avoid security problems.

(6) Portable

(7) Multithreading

(8) Dynamic

Data Types

- Primitive
  - Numeric
    - integer (4)
      - Byte (1)
      - Short (2)
      - int (4)
      - long (8)
    - Boolean (4)
    - double (8)
  - Non-Numeric
    - Boolean (1)
    - char (2)
- Non-primitive
  - class
  - interface
  - String
  - Array.

* Two types of typecasting

① Widening : A low datatype is transform into higher one by process known as widening. casting down and implicit casting are some name for it. It occur naturally, since there is no chance of data loss. It is secure it occurs when

- the target type must be larger then source type.
- both datatypes must be compatible with each-other

② Narrowing : the process of down sizing a bigger datatype into smaller one is known as Narrowing type, casting are other names. It does not first happen by itself if it do not explicit do that a compile time error occur. data loss might happen due to lower datatype smaller range of permitted value

Note - A cast operator assist in the process of explicit casting.

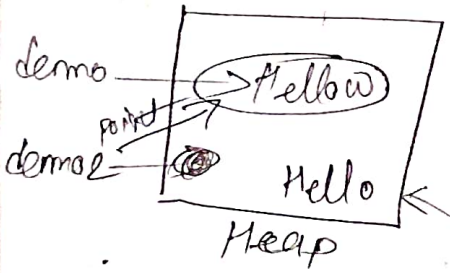
* String in Java: Java strings are object that allow us to store sequence of character which may contain alphanumeric values and enclosed in double quote (" "). It act the same as an array of character.

Note: • Strings are immutable in Java
• It contains methods that can perform certain operations on string.

① Concate ② equal ③ Split ④ length ⑤ replace ⑥ substring

There are two ways to create string

① String literal to make Java more memory efficient because no new object are created if it exist already in string constant.




```
String demo = "Hello";
demo = concate("world");
String s = new String("Hello");
```

② Using new keyword: In such case JVM will create a new string object in normal heap memory

Single quote → \'
double quote → \"
Backslash → \\

methods of string: ① tolowercase → to lowercase
② touppercase → to uppercase (s.touppercase())
③ length fun →
④ indexof →
⑤ lastindexof →

* Concatination Method:

```
class A
{ psvm
  {
  String demo = "Hello";
     sop(demo
  String demo2 = "Hello";
  demo.concat("world");
     sop(demo);
  demo = demo..concat("world");
     sop(demo);
  }
}
```

* literals: The data items which never change their value through a program run are called literals, it is of four type
(i) Number literal (ii) character literal
(iii) string literal (iv) Boolean literal

(i) Number literal → int
→ float

* variables in JAVA: variables are containers for storing value in java. In java there are diff types of variables.

Syntax: datatype var_name = value;

* General rules for naming variable :

1. (i) Names can contains letters, digits, _ and doller $ sign.
   (ii) Name must be begain with letter.
   (iii) Name should start with lowercase letter cannot contain white space.
   (iv) Names are case-sensitive
   (v) Reserved keyword are not used.

* keyword : Yield, record, sealed, non-sealed, permits

* Scanner class : It is used to take input from user.

```
import java.util.Scanner
import java.util.Scanner
class userInput
{ psvm
  {
  Scanner S = new Scanner (System.in);
  sop ("Enter gender");
  char gender = S.next().charAt(0);
  S.next()
  S.nextLine()
  S.nextInt()
  S.nextLong()
  S.nextBoolean();
```

* Types of variable.

(i) local variable (ii) Instance variable
(iii) Static variable.

(i) local variable : A variable which is declared in the the method or method parameter is called local variable.

(2) Instance variable : A variable which is declared inside the class but outside of all the methods is called instance variable.

(3) Static variable : variable which is declared with static keyword.

* classes : classes is a collection of object It does not take any memory space It is also called as blue print, logical entity, there are two types of classes in Java first is predefined and second one is user defined.

(i) Predefined : scanner class, system class, string, class.
(2) User defined : Any name for ex → demo.

* object : object is instance of class that execute the class once the object is created It takes sub-space like other variables in memory

syntax : class-Name → dynamic mem alloc
Scanner sc = new Scanner();
object reference

* Method in Java: Method is a group of block of code which takes input from user and process it to give output. method run only if called.

Ex → println(), sout(), sqrt()

In Java functions are called method because we have to create inside class they are used with class name and have no body.

```
class A
{
  psvm {
  A r= new A();
    r.disp()
}
void disp()
{
sop("methods in Java");
}
```

① Input → ② process → ③ output

```
import java.util.Scanner
Public class calculator {
  Private int n1, n2;
  Private int a1, b1;
  Public Static void input() {
  Scanner s = new Scanner(System.in);
  SOP("Enter the first number:");
    n1 = Scanner.nextInt();
  S.O.P("Enter the second number:");
  n2 = Scanner.nextInt();
}
public static void process() {
      a1 = n1 + n2;
      b1 = n1 * n2;
    }
  Public static void display() {
    S.O.P("sum of " + n1 + " and " + n2 + " is:" + a1);
    SOP("product of " + n1 + " and " + n1 + " is:" + b1);
  }
public static void main(String[], args) {
Calculator cal = new calculator();
calculator.input();
calculator.process();
calculator.display();
}
```

* Method Arguements.

```
Public class A
{ public static void Method(int x)
    {
      Sop("The value of x is" + x);
    }
    psvm.
      {
        Method(8);
      }
}
```

Topic → String, int & function overloading

* Method overloading:

* public class overloading

```
{ public static int addMethod(int x, int y)
    { return x+y;
    }
  public static double addMethod(double x, double y)
    { return x+y;
    }
PSVM {
  int ans1 = addMethod(5,5);
  double ans2 = addMethod(5.5, 4.5);
  Sop("int addition is:" + ans1);
  Sop("double addition is:" + ans2);
}
}
```

If class has multiple methods having same but different parameter is known as method overloading.

*Advantage of method overloading.

① Increase the readability of program.

② there are two ways to overload method in java

(a) by changing no of arguments and

(b) by changing the data type.

* Constructor: In java constructor is a block of code similar to the method. It is called when an instance of a class is created the time of calling constructor memory for the object is allocated in the memory, it is a special type of method which is used to initialized the object every time an object is created using new keyword at least one constructor is called, it calls a default constructor if there is not constructor available in the class. there are two type of constructor

(i) No argument constructor

(ii) parameterized

Rules for creating constructor.

① constructor name must be same as class name.

② A constructor must have no explicit return type.

③ A Java constructor cannot be abstract, static, final & synchronized.

④ In JAVA constructor just like a method but with not return type.

(5) Constructor can also overloaded like Java method.

difference b/w constructor and method in java

Constructor	method
(1) constructor is used to initialized the state of an object	(1) A method is used to expose the behaviour of an object
(2) A constructor must have no Return type	(2) A Method must have a return type
(3) A constructor is invoked explicitely.	A method is invoked explicitly.
(4) The Java compiler Provide a default constructor if you don't have any constructor in a class.	A Method is not provided by compiler in any way.
(5) The constructor name must be same as class name	The method name may or may not be.

diff b/w final, finally, finilized (PYQ/imp)

final	finally	finilize
(i) Final is a keyword and access Modifier which is used to apply restriction on a class method or variable.	(i) Finally is a block in Java exception handling to execute the important code whether the exception occurs or not.	(i) finilize is a method in java which is used to perform cleanup processing just before object is garbage collected.
(ii) Final keyword used with class, Method and variable	finally block is always related to try and catch block in exception handling.	(ii) finilize method is used with objects.
(iii) Once declared final, variable becomes constant and cannot be modified.	finally block runs the important code even if exception occurs or not.	It Performs cleaning activities with respect to object before its destruction.
(iv) final method cannot be over-written and cannot be inherited.	finally block clean up all resources used in try block.	
(v) Final Method is executed only when we call it.	Its execution is not dependent on the exception.	finalize method is executed just before the object is destroyed.

This keyword : This is a reference variable which refers to the current object use of this keyword

(i) This can be used to reference current class instance variable

(ii) This can be used to invoked current class method

(iii) (this) method can be used to invoked current class constructor

(iv) this can be passed as an argument in the method called.

(v) this can be passed as argument in calling

(vi) this can be used to return the current class instance from the method.

Garbage collection: Garbage collection means unreferenced objects, garbage collection is process of reclaiming the runtime unused memory automatically, in other words, it is a way to destroy unused objects.

to do so they were using free function in c language and delete function in c++ but in java it is performed automatically

advantages of garbage collection: It makes Java memory efficient because garbage collector remove the unreferenced object from heap memory, it is automatically done by a part of JVM

How can an object be unreferenced: (i) By nulling the reference (ii) by assigning a reference to another. (iii) by anonymous method (iv) finalized method

Q Make a process class and its instance, roll no, name, display method print both variable.

```
public class A {
    int Roll_No;
    String Name;

    A() {
        Roll_No = 101;
        Name = "Raghu";
    }

    static void display(String[] args)
    {
        Sop("Name is " + Name);
        Sop("Roll_No is" + Roll_No);
    }

    Psvm ( ) {
    A obj = new A();
        obj.Display()
} }
```

constructor

{ final, finally, finalize }

One-D Array :

Declaration and initialization in one line:

```
int numbers[] = {1,2,3,4,5};
```

```
// Declaration
int numbers[];
// initialization.
numbers = new int[5];
```

* 2-D Array :

```
// Declaration
  int matrix[][];
// initialization
matrix = new int[3][4];
```

* Declaration and initialization in one line

```
int matrix[][] = {
                  {1,2,3,4}, {5,6,7,8}, {9,10,11,12}.
                  };
```

Command Line Arguments:

Ans: A command line argument is the information that directly follows the program name on the command line when it is executed to access inside a Java program it is very easy, they are stored as String array passed to the arguments of main.

```
Class cmdLine {
    Psvm {
        for (int i=0; i<args.length; i++)
        {
            Sop("args[" + i + "] + args[i]
        }
    }
}
```

Q1. WAP that takes two variable and test if both between 0 and 1 and false otherwise

Q2. write a java method to display the middle characters of a string

Q3. write a java method to count all the words in a string

Q. WAP to overloading constructor.

Assign

Q1 → Explain Specifier in java

Q2 → Diff b/w final and finalize

Q → write a recursion function to print factorial and fibonacci series.

Inheritance: It is a mechanism in which one object aquires all the properties of parent the idea behind inheritance in JAVA is that you can create a new class that are built upon from existing class, we can use methods and fields of parent class moreover we can add new methods and fields in your current class.

Q Method overloading vs method overriding with example.

Advantage of inheritance: (i) for reusability
(ii) for method overriding.

Note: A subclass contain all the features of superclass so we should create the objects of subclass.

* Terms used in inheritance:
(i) class (ii) subclass (iii) superclass (iv) parent class
(vi) reusability

* Types of inheritance
(i) Single inheritance
(ii) Multilevel (more than one class).
(iii) Hierarchical inheritance

* Method overriding: If subclass has a same method as declared in the parent class it is known as Method overriding.

* Rules for Method overriding:
The Method overriding name must have the same name as parent class, and the parameters must be same.

* Diff b/w Method overloading and method overriding

Method overloading	Method Overriding.
Method overloading is a compile time polymorphism	Method run time polymorphism
(ii) It helps to increase readability of program.	It is used to grant the specific implementation of the method which is already provided by parent class or Superclass.
(iii) It occurs within the class.	It is performed in two class with inheritance relationship.
(iv) Method name must have Same and different Signature.	Method name must have same name, same signature in this dynamic binding.
(v) Static binding is being used for overloaded method	
(vi) The Argument list should different.	The Argument list should be same.

* super keyword, Abstract class,
* super vs super()

* super : The super keyword in java is a referen variable which is used to refer, ~~~~ immediate parent object, ~~~~

Uses of super keyword:

(i) super can be used to refer immediate parent class ~~~~ instance variable.

(ii) Invoke parent method.

(iii) Invoke parent constructor.

```
class A
{
  public void show()
  { sop("It is a object");
  }}

class B extends A
{ public void show()
  { sop();
  }}

class demo
{ psvm
  { obj = new B();
    obj.show();
  }}
```

upcasting

```
A obj = new A();
obj.show();
```

obj refereme

upcasting

* Abstraction : Abstraction is the process of hiding the implementation details and showing only functionality to the user, A class which is declared with abstract keyword is known as abstract class. It can have abstract or not abstract method, It can also have constructor and static method. If a class contain at least one abstract method then it is compulsory, you should declare abstract class. we cannot create object of Abstract class directly.

* Dynamic method dispatch mechanism by which a call to an overriden method is resolved at run-time rather than compile-time

(i) At run-time it depends on the type of object refered to (not the type of reference variable") that determine which version of overriden method will be execute.

(2) A super class reference variable can refer to a sub-class object. This is also called upcasting.

* Advantages of dynamic method dispatch

(i) It allows Java to support overriding of method which is center for a runtime polymorphism.

(ii) It allows a class to specify methods that will be common to all derived by allowing subclasses to define specific implementation some or all of those methods.

* Static V/s dynamic binding

features	Method overloading	Method overriding
Definition	Multiple Method with the same name but different signature.	Method name same as well as signature must be same.
Polymorphism	Compile time polymorphism	runtime polymorphism.
parameters	Must be different	must be same
Inheritance	No need of inheritance	inheritance is required
Return Type	can have different return type	Must have same return type
class	It occures within a same class	It occurs in two different with the help of inheritance
Access Modifiers	Can have any access Modifiers.	Must have accessible Modifiers.
Notation	No any Special Notation is required	@override notation is often used (optional but recommended).

Pg 8 Difference b/w recursion and iteration.

Features	recursion	iteration
Definition	A method that calls itself to slove a problem.	A method that uses loops to repeat a block of code
Approach	Breaking down a problem into smaller tasks of the same problem.	Repeats a block of code until a condition is met.
Memory usage	Uses more memory due to call stacks	Uses less memory as it involves simple loops.
Time usage	May takes more time	It takes less time than recurtion.
Base case	requires a base case.	No need of base case.
Function calls.	Involves multiple function calls	It does not involve function call
Example	Factorial, fibonacci	factorial, fibonacci, iterate over array.
Stack over-flow	It can cause stack over flow	No any concept of stack here.

Q difference b/w procedural programming and object-oriented programming.

features	Procedural programming	object-oriented programming
Focous	Focous on functions or procedures.	focous on objects and classes.
Approach	Top-Down approach	Bottom-up approach
division	Code is divided into functions or procedures.	Code is divided into classes and objects.
Inheritance	Not supported	supported
polymorphism	Limited to function overloading	Supports method overloading and operator overloading, overriding
Example lang	C, Pascal	Java, C++, Python
Encapsulation	Limited encapsulation.	Strong encapsulation.

* This keyword: In java, the this keyword is a reference to the current object, used primarily in object-oriented programming to avoid ambiguity between class attributes and parameters.

(i) Distinguishing between instance variables and parameters.

Eg⇒
```
public class Example {
    int num;
    Example(int num)
    {
        this.num = num;
    }
}
```

(ii) Calling one constructor from another (constructor chaining)

The this() keyword can be used to call another constructor in the same class.

```
public class Example
{   int num;
    String str;
    Example()
    {   this(0, "Rauthem");
    }
    Example(int num, String str)
    {   this.num = num;
        this.str = str;
    }
}
```

(3) Referring to the current object: this can be used to pass the current object as an argument to another method.

```
public class Example
{   public void print()
    {   System.out.println(this);
    }
    psvm()
    {   Example e = new Example();
        e.print()
    }
}
```

o/p → Example@3e41c6

this refers to the current obj basically its address.

④ Returning the current object: (object of return type)

```
Public class Example {
    int num;
    Public Example SetValue(int val)
    { this.num = val;
      return this; // Returning the current object
    }
}
```

⑤ Calling methods of the current class

```
public class Example {
  public void method1()
  { sop("method 1");
  }
  public void method2()
  { this.method1(); // Calls method1() of the current object
  }
}
```

*Command Line Arguments: In Java command line argument are passed to the main method of Java program which is defined as Public Static void main(String[] args). The args parameter is an array of String objects that contains the arguments passed to the program.

```
Public class CommandLineExample {
    Public Static void main(String[] args)
    { if(args.length>0)
      { for(int i=0; i<args.length; i++)
        { sop("Argument" + i + ":" + args[i]);
        }
      }
```

else { sop("No Command is passed"); }

}

* super keyword: super is a keyword in Java that is used for several reasons.

(i) Accessing parent class methods:

```
class parent {
  void print()
  { sop("parent class"); }
}
class child extends parent
{ void print()
  { super.print();
    sop("child class");
  }
psvm { child c = new child();
       c.print();
     }
}
```

op: parent class
child class

(ii) Accessing parent class constructors:

```
class parent {
  parent(){
    sop("parent const");
  }
}
class child extends parent {
    child()
  { super();
    sop("childd const");
  }
  psvm {
    child c = new child();
  }
}
```

op: parent const
child const

(iii) Accessing parent class variable:

```
class parent
{ int val = 10;
}
class child extends parent
{ int val = 20;
  void show()
  { sop(val);      // 20
    sop(super.val); // 10
  }
  psvm {
   child c = new child();
    c.show();
}}
```

* Dynamic Method Dispatch: Dynamic method Dispatch mechanism implements runtime polymorphism, it occurs when a method is overridden in a subclass. when a method is called on a parent class, which refers to a child class object, then overridden method in the child class is called at runtime. and the execution of program at runtime called dynamic method Dispatch.

Ex ->

```
class parent {
  void show()
  { sop("parent");
  }
}
class child extends parent
{  @Override
   void show()
  { sop("child");
  }
  psvm {
  parent obj = new child(); // upcasting
  obj.show(); }}
```

* Abstraction:

```
abstract class Animal
{
    abstract void Sound(); // dog not have a body
    // regular method
    void eat()
    { Sop("eating...");
    }
}
class Dog extends Animal
{   @Override
    void sound()
    { Sop("woof woof");
    }
    public static void main(String[] args)
    { Animal dog = new Dog();
      dog.Sound(); // o/p : woof woof
      dog.eat(); // Eating...
    }
}
```

* Final keyword : (i) Final classes : A class marked as final cannot be inherited.

Ex →

```
final class FinalClass{
        public void display()
        { Sop("This is final");
        } // Note!: cause a compile time error
}
```

(ii) final method : A method marked as final cannot be overridden by subclasses.

```
class parent{
    public final void show()
    { Sop("final");
    }
}
```

```
class child extends parent
{ // @override
// public void show() // cause a compile time error
// { sop("in child");
// }
}
```

3. final variable: A final variable acts like, ~~same~~ a constant. Once initialized, it cannot be changed.

Ex→
```
class parent{
    public final int constant = 10;
    public void display()
    { sop("constant=" + constant);
    }
}
class child extends parent
{ public void change()
  { // constant = 200; // cause a compile time error
  }
```

Subclass in Java: Java programming language allows you to define class inside another class such class is called nested class, Nested classes are divided into two category.

(i) static Nested class

(ii) non static Nested class (inner class):

(i) static Nested class: A static nested class can access the static methods of the outer class, but it cannot directly access non-static members.

Example:
```
class Outer {
    static int data = 100;
    static class NestedStaticclass {
        void disp(){
        sop("Nested class data" + data); } } }
```

```
Public class main{
public static void main(String[] args)
{
  Outer.NestedStaticClass nested = new Outer.NestedStaticCl
  nested.disp();        // Access static nested class without
}                       // creating an instance of outer.
}
```

output ⇒ nested class data : 100

(2) Non-static Nested class (inner class): A non-static inner class (also called an inner class) has access to all members of the outer class, including private ones.

Example:

```
class Outer{
    private String msg = "Hello";
    class Inner {
     void disp()
     { sop(msg);
     }
}
public class main{
public static void main(String[] args)
{ Outer outer = new Outer();
  Outer.Inner inner = outer.new Inner();
  //creating an instance of inner class
  inner.disp(); // Hello.
}}
```

* Need of nested class.

(i) It is a way of logically grouping classes that are only used in one place.

(ii) It increases encapsulation

Ex ⇒ Consider two classes A and B. B need to access members A, that would otherwise declare private.