: Java :

Date: Page No.

★ What is Java ?

- Java is popular programming language created in 1995. sun microsystem but now it is owned by oracle.
- Java is a highlevel programming language which is based on object oriented programming games Gousling known as the father of Java

★ Why use Java ?

1) Java works on different platfrom (windows, mac, linux, Raspbery Pi, etc.)
2) It is one of the most popular programming language in the world
3) It has a large demand in the current job market.
4) It is easy to learn and simple to use.
5) It is open-source and free
6) It is secure, fast and poweful
7) It has huge community support (tens of millions of developers)
8) Java is an object orented language

★ Feature of Java ÷

1) simple ÷ Java is easy to learn and its

Date: / / Page No.

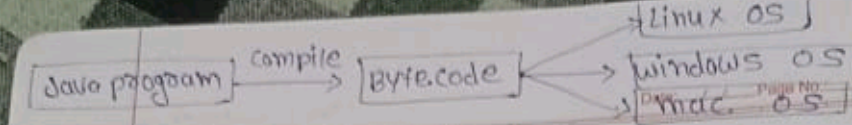
Syntax is quite simple, clean and easy to Understand. The confusing and ambiguous concept of c++ are either left out in Java or they have been re-implemented in a cleaner way.

2) Object oriented :- In Java everything is an object which has some data and behaviour Java can be easily extended as it is based on object modal. following are some basic concept of OOP's.

1) Object
2) Class
3) Inheritance
4) Polymorphism
5) Abstraction
6) Encapsulation

3) Platform Independent :- Java is platform Independent because it is different from other language like c, c++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

There are two types of platform software-based and hard-ware. Java provides a software-based platform.



4) Secured :- when it comes to security, Java is always the first choice. with Java secure feature it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

5) Robust :- The English mining of Robust is strong. Java is robust because.
1) It uses strong memory management.
2) There is a ~~black~~ lack of pointers that avoids security problems.
3) Java provids automatic garbage collection which runs on the Java virtual machine to get rid of objects which are not being used by a Java application anymore.

6) Architecture - neutral :- Java is architecture ~~neu~~ neutral because there are no implementation dependent feature for example, the size of primitive types is fixed.
In computer generates bytecodes which have nothing to do with a particular computer architecture hence a Java program is easy to interepret on any machine.

Date: / / Page No:

7) Portable :- Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

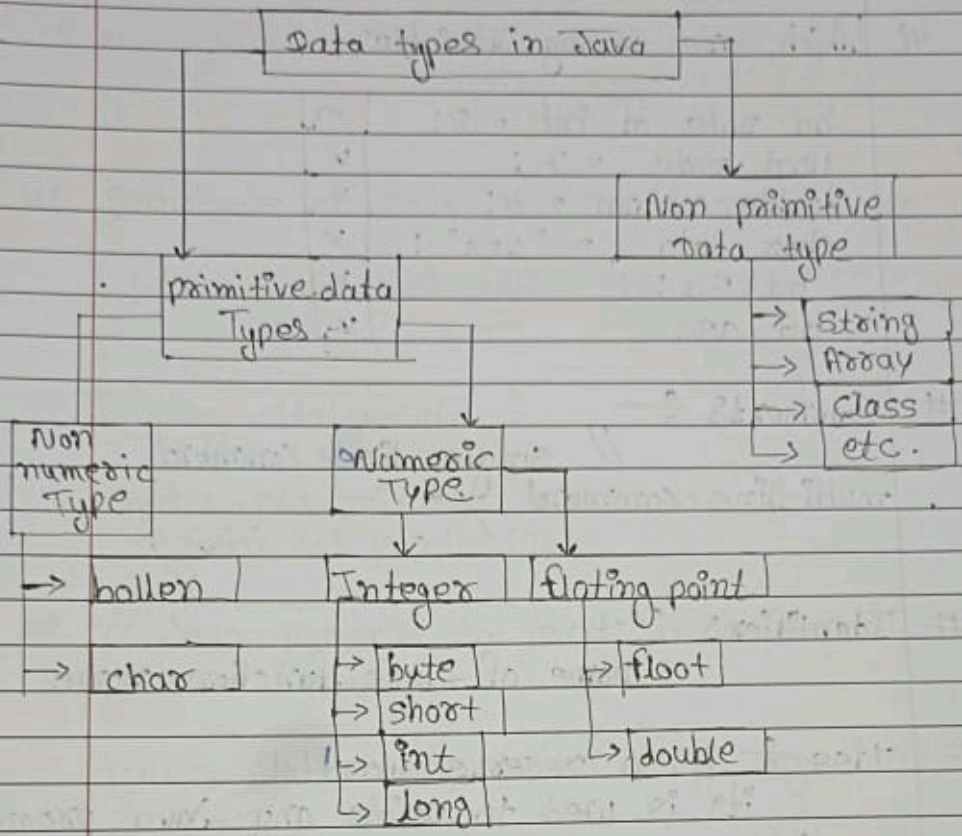
8) High-performance :- Java is an interpreted language, so it will never be as fast as a compiled language like C or C++ But Java enable high performance with the use of just-in-time compiler.

9) Distributed :- Java is also a distributed language programs can be designed to run on computer networks. Java has a special class library for communicating using TCP/IP protocols creating network connections is very much easy in Java as compared to C/C++.

10) Multi Threaded :- Java multithreading feature makes it possible to write program that can do many tasks simultaneously Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time like while typing, grammatical errors are checked along.

11) Dynamic :- Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native language C and C++.

Data types in Java



4 bytes float s = 23000.89;
String name = "Shilpi Jain";
String address = "170 indore 452005";
String nationality = "India";
edu = "MCA";
4 bytes int age = 34;

size Data type Variables = Value

declaration & assigning of variables.

Date: / / Page No.:

Which are wrong Variables

int rate of int = 8;	x
float rate = 7;	✓
double return = 4;	x
char data = "yes";	x
int 3a;	x
int a3;	✓

Comments :-

// single line comment

multi line comment /* ... */

Identifiers :-

name of class, function name

class :- its a reserve word

its is used to write any Java program

its a user defined datatype.

{ } :- It is used to create a block which is set of ms

main() :- is the function which is called by compiler to convert the Java code into bytecode, its the entrance of program from where execution start.

Date: / / Page No:

main function :- public static void main (String args []) { }

Structure of Java program class Example { public static void main (String args []) { } }

Printing statements :-

System.out.println ("hello word");
System.out.println (20);

// Java program to print "hello world"
class Example
{
public static void main (String args [])
{
System.out.println ("hello world"); }
}

Escape sequence :- \n next line
\t tab space
\b Backspace

program to print value of a variable
class Hello word
{

Date: / / Page No:

```
public static void main (String [] args)
{
  int a = 10;
  System.out.println ("hello word");
  System.out.println ("hello word" + " " + "a");
  System.out.println ("hello word " + " " + a);
```

output →
hello word a
hello word 10

Program to print add to two number

```
Class Helloworld {
public static void main (String [] args)
{
  int a = 10;
  int b = 34;
  int c = a+b;
  System.out.println ("first value = " + a);
  System.out.println ("second value = " + b);
  System.out.println ("result = " + c); }
}
```

output → frist value = 10
second value = 34
Result = 44

Program to print Average of 5 Numbers :-

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

```
int a,b,c,d,e;
int total ave;
a=30;
b=45;
c=69;
d=35;
e=56;
total = a+b+c+d+e;
Ave = total/5;
System.out.println("total marks is=" + total);
System.out.println("Average is =" + Ave); }
}
```

larger number of two(2) numbers :-

```
public class larger {
public static void main (String [] args)
{
int a=50;
int b=60;
if(a>b){
System.out.println("larger number =" +a);
}
else { System.out.println("larger number="+b);
}
```

program to print larger among 3 number :-

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

```
int a = 45 ;
int b = 35 ;
int c = 75 ;
if(a>b && a>c) {
System.out.println("a is greter than =" +a);
}
else if (b>a && b>c) {
System.out.println("b is greter than =" +b);
}
else { System.out.println("c is greter =" +c);
}
```

output -> c is greter

Program to print swapping 2 number without using third variable -

```
class swappingTwo {

public static void main (String [] args) {
int a = 40;
int b = 90;
a = a+b;
b = a-b;
a = a-b;
System.out.println("a is =" + a);
System.out.println("b is =" +b); }
}
```

o/p -> a = 90
b = 40

Date: / / Page No:

Swapping 2 numbers using third variable -

```
class swappingthird {
public static void main(String [] args) {
    int a = 10;
    int b = 5;
    int c;
    System.out.println("a = " +a "\n" "b = " +b);
        c = a;
        a = b;
        b = c;
    System.out.println("Value of a is = " +a);
    System.out.println("Value of b is = " + b); }
}
```

O/P — Value of a is = 5
Value of b is = 10

Calculate superbazar bill (bill = qty * rate) and provide for discount to get net bill amount -

```
class SuperbazarBill {
public static void main(String [] args)
{
    int qty, rate, bill, dis, net;
        qty = 60;
        rate = 200;

        bill = qty * rate;
        dis = bill * 10/100;
        net = bill - dis;
```

byte → short > int > long > float > double

Date: / / Page No:

System.out.println("The bill is = " + bill);
System.out.println("Discound on bill is =" + dis);
System.out.println("net value is =" + net);
}
}

o/p → The bill is = 12000
Discount bill is = 1200
Net value is = 10800

Typecasting %

// float value are assigned using f

class Helloworld {
public static void main(String [] args)
{
float f = 23;
float l = 5.67f;
System.out.println("f=" + f);
System.out.println("l=" + l); }
}

Typecasting +

class Helloworld {
public static void main (String [] args)
{
byte b = 10;
byte v = b + (byte)20;
System.out.println("V=" + V);
}
}

- 20 is treated as integer 20, error.
- typecast 20 as integer then resolved.

O/P -

```
byte V = b + (byte)20;
              ↑ error
```

```
class Helloworld {
public static void main(String [] args)
{
  byte b=10;
  byte V = (byte)(20+b);
  System.out.println("V="+V); }
}
```

Write a Java program to print an int a double and a char on screen.

```
Class Helloworld {
public static void main(string [] args)
{
  int a=100;
double b = 20.345;
  char c = 'f';
system.out.println("int value is="+a+"\n
    double is ="+b+"\n char value is="+c);
}
}
```

O/P → int value is = 100
double is = 20.345
char value is = f

Date: / / Page No.:

Write a program to print the area of rectangle of sides 2 and 3 units respectively.

```
Class Area {
  public static void main(String [] args){
    int a,b, area;
    a = 2;
    b = 3;
    area = a*b;
  System.out.println("Area of rectangle is = "+area);
  }
}
```

O/P → Area of rectangle is = 6

Write a program product of the number 8.2 and 6.

```
Class Product {
  Public static Void main (String [] args)
{
  System.out.println("product of the number is="
                    + 8.2* 6);  }
}
```

O/P → Product of the number is = 49.2

Print the Ascii Value of the character 'h'

```
Class Ascii {
  public static Void main(String [] args)
{
```

Date: / / Page No:

System.out.println(" The Ascii value of h is =" +(int)'h');

}

} O/P → The Ascii value of h is = 104

Ascii → American standard code for Information Interchange.

Write a program to print assign a value of 100.235 to a double variable and then convert it to int.

Class Helloworld {

public static void main (String [] args)

{

double a = 100.235;

System.out.println((int) a);

}

} O/P ⇒ 100

1/5/24

Date: / / Page No.

÷ operator ÷

2 + 3
operands

operator .. "operators are used to perform operations on operand"

Types of operators ÷

① Arithmetic ÷ +, −, ×, /, % (% modulous)

Example. — 3)10(3 → quotient
−9
1 → Remainder

```
class Example {
    public static void main(String [] args)
{
int a = 10;
int b = 3;
System.out.println("modules =" + a%b);
                   ("quotient =" + a/b); }
}
```

② Relational → >, <, >=, <=, !=

③ logical → && || !
and or not

Truth Table (and) *

C1	C2	R
T	T	T
T	F	F
F	T	F
F	F	F

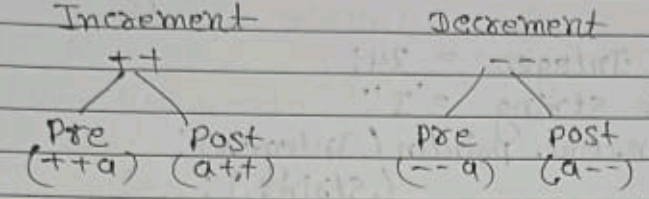
Truth Table (or) +

C1	C2	R
T	T	T
T	F	T
F	T	T
F	F	F

NOT - C R
T F
F T

Date: / / Page No:

(34) Unary Operators →




```
# class postIncrement {
  public static void main (String [] args)
{
   int a = 10;
  System.out.println ("a = " + a++); } 10
}                                     11
```

```
## class preIncrement {
  public static void main (String [] args)
{
   int p = 100;
  System.out.println ("p = " + ++p);   101
```

```
### class Decrement {
  public static void main (String [] args)
{
   int x = 50;
  System.out.println ("x = " + --x);   49
  System.out.println ("x = " + x--);   49
  System.out.println ("x = " + --x);   47
```

Date: / / Page No.

class Example {
public static void main (String [] args)
{
int Integer = 24;
char string = 'J';
System.out.println (Integer);
(string);

"Both are Variable as ~~the~~ we have diclared that both of them with datatypes."

public class solution {
public static void main (String [] args)
{
short x = 10;
x = x * 5;
System.out.println (x); }
}

O/P ->

error: incompatible types: possible lossy conversion from int to short -

① class Helloworld {
public static void main (String [] args)
{
short x = 10;
int y = x * 5;
System.out.println (y);
}
}

Date: / / Page No.

② short x = 10;
x = (short) (x * 5);
System.out.println(x); }

# class Helloworld {
public static void main (String [] args) {
byte x = 127;
x++;
x++;
x = (byte) (x + 125);
System.out.println(x);
}
}

(x+) = -127
(x+127) = 0
(x+126) = +1

Class Example {
public static void main (String [] args)
{
boolean b1 = true;
boolean b2 = false;
boolean b3 = 0;
boolean b4 = 1;
System.out.println(b1);
System.out.println(b2);
—— (b3);
—— (b4);
}
}

op/	
	true
	false
	error
	error

Write a Java program to take two integer value and print there sum, Difference, product, quotient, Remainder.

Date: / / Page No.:

```
Class Arithmetic {
public static void main (String [] args)
{
int a, b, sum, diff, product, quotient, remender;
 a = 100;
 b = 3;
System.out.println(" Answers of all operation
 +" \n sum = "+(a+b)+" \n differenc = "+
(a-b)+" \n product = "+(a*b)+" \n quotient
 ="+(a/b)+" \n remender = "+(a%b);
}
}
```

o/p →

Answer of all operation
Sum = 103
difference = 97
product = 300
quotient = 33
remender = 1

÷ Linux ÷

Q. What is Linux ?

Ans. Linux is a free open source operating system that supports both GUI and CLI interface.
GUI = Graphical user interface
CLI = Command line interface
Linux is considered to be fast and secure.

Commands ÷

① ls → lists all the files and directories under a specified directory.
example → ls

② ls -a commands → The (ls -a) command will enlist the whole list of the current directory including the hidden files.
example → ls -a

③ Pwd → Print working Directory. Pwd displays the the full pathname of the current directory you are in.
example → Pwd

④ mkdir → Creates one or more new directory specified by the directory parameter.
example → mkdir d1 d2

⑤ cd → (chang Directory). It allows you to change your current working directory.

Example → cd dc.

(6) touch → Used to create a new empty file
example → touch dxx.

(7) cd.. → will switch back up one directory from the current directory.
example → cd..

(8) cat → we can write any message or parcal in the file.
Example → cat>f → कुछ लिखने के लिए
cat f → massage पढ़ने के लिए file का
cat >>f → लिखे massage में और लिखना add करने के लिए

(9) rmdir → This command is used to delete a directory.
Example → rmdir d1

(10) rm → for remove files and directoryes
Example → rm dx.

(11) rm -f → its used the -f option to force the deletion of such directories along with all their contents.
example → rm -f f

(12) mv → Used to move file and rename file.

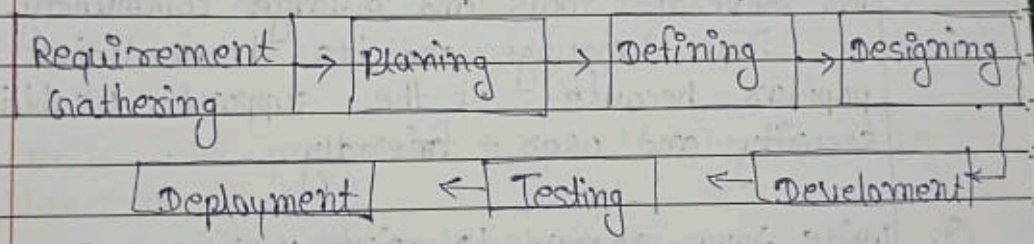
Date: / / Page No:

Example → mv f → rename
mv f ~/d2 → move file
Path

(13) clear → clear the terminal screen in Linux using the clear command.
example → clear

(14) history → check to all command.
example → history → see all command.

Q. SDLC → Software Development life cycle is a ~~the~~ structured process that enables the production of high-quality low-cost software in the shortest possible production time.



Q. what is GitHub ?

Github is a developer platform that allows developers to create, store, mange and share their code.
files and collaborate with fellow developers on open-source projects.

Date: / / Page No:

Q Explain JDK and JRE ?

JDK (Java Development Kit) is used to develop Java application. JDK also contains numerous development tools like compilers, debuggers etc.

JRE (Java Runtime Environment) is the implementation of JVM (Java virtual machine) and it is specially designed to execute Java programs.

Q. Why Java is popular and better than other language?

Ans. Java is an object-oriented, open-source and easy computer language for beginners. The advance programmar demands Java language for several tools and a large community.

Java language and its technologies are popular because of the support, usability, security and user-friendly.

Q. Write Java program to print your name and hobby?

Ans.
```
class Name {
    public static void main (String [] args)
    {
        System.out.println("my name is shivani meena");
        System.out.println("my hobby is are Dancing and leatening music.");
    }
}
```

Date: / / Page No:

Q. Write a Java program to print to perform addition and substraction.

Ans.
```
class Operations {
    public static void main(String [] args)
{
    int a=30;
    int b=10;
    int c = a+b;
    int d= a-b;
System.out.println("Addition of two number="+c);
- - - - - - - ("Substraction of two number="+d);
}
}
```

Q. What is used of sudo →

Ans. Sudo is a command in Linux that allows user to run commands with privileges that only root user have. (temporary).

Q Write Java programs to print hello world 10 times →

Ans
```
class Printhelloworld {
public static void main(String [] args)
{
for(int i=1 ; i<=10; i++)
{
System.out.println("Hello world");
}
}
```

Q. What is front-end and what technology can be used.

Ans. Front end technologies are an essential part of web development that enhance the user interface and overall experience.

Here are some of the benifits of front end technologies create a modern, responsive website with an improved user experience. with the latest css3 standards and HTML5 tags : create a great website.

Q. what is public static void main() ?

public static void main (String [] args);

Access modifire — Object return type — name of function

नहीं बनाना

public →

The access modifier of the main method needs to be public so that the JRE can access and execute this method.

Static → There is no need to create an object

Void → is used to specify that a method doesall return anything.

Void is return type of main method.

main → It is the name of the Java main

Date: / / Page No:

method. It's not a keyword.

String [] args → It store Java command-line arguments.

4 Types of Number system -
Binary, Decimal, Octal, Hexadecimal.

Binary → 2 symbol - 0, 1
Decimal → 10 symbol (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
Octal → 8 symbol
Hexadecimal → 16 symbol

Decimal to Binary →

2	21 → 1		2	16 → 0	
2	10 → 0		2	8 → 0	
2	5 → 1		2	4 → 0	
2	2 → 0		2	2 → 0	
	1			1	

Ans- 10101 Ans- 10000

Binary to Decimal →

10101 → ~~multiplication~~
43210 → Power
$1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 * 1\times2^0$
16 + 0 + 4 + 0 + 1
→ 21

10011
43210
$1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$
→ 16 + 0 + 0 + 2 + 1
→ 19

19/05/24

Date: / / Page No.

Golden Rules :-

n = 4961

→ To Remove last digit
⇒ n%10

→ To get remaining number after removing last digit
⇒ n/10

→ To shift and append number
⇒ n*10+0

n = 4961
10|4961
Ans (1) - rem
Ans = 496
→ 4961/10
⇒ 496

Oops :-

Object oriented programming or Oops refers to language that use objects in programming they use objects as a primary source to implement what is to happen in the code.

object oriented programming aim to implement real world entities like inheritence hiding polymorphism etc.

in Java Object-oriented programming is fundamental to the language's design. Here's why oop is important in Java.

- Object-oriented programming or Java Oops concept refers to language that use object in programming.
- The main aim of oop is to bind together the data and the functions that operate on

① इसमें program का structure बहुत ही simple होता है जिससे complexity कम होती है।
② इसमें किसी code को एक बार लिखने की जरूरत होती है इसलिए बार-बार use कर सकते हैं।
③ It data redundancy पूरा करता है ④ इसमें debugging आसानी से हो सकती है।
⑤ इसमें code को आसानी से Maintain किया जा सकता है। Date: 5/1 Page No:
⑥ object-oriented programming में data hiding और abstraction का सुविधा होता है जिससे security बढ़ती है।

them so that no other part of the code can access this data except that function.

"Oops" is a common acronym in programming that stands for "Object-oriented programming." Object-oriented programming is a programming paradigm that revolves around the concept of objects which can contain data and code.

Oops concepts :-

- class
- Objects
- Inheritance
- polymorphism
- Abstraction
- Encapsulation

Class :-

- A class in Java is a ~~get~~ set of object which shares common characteristics / behavior and common properties / attributes.
- It is a user-defined blueprint or prototype from which objects are created for example, Student is a class while a particular student named Sonam is an Object.

properties of Java classes -

- Classe is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.
- Class is a user-defined data type.

OOPS एक object-oriented programming है जो object पर आधारित है। यह सॉफ्टवेयर डेवलपमेंट और मेंटेनेंस को सरल बनाने के लिए कुछ कॉन्सेप्ट प्रदान करता है जिससे प्रोग्राम को आसानी से बना सकें और कॉम्प्लेक्स प्रॉब्लम solve कर सकें। object-oriented programming में सभी चीजों को object के रूप में मानकर programming की जाती है। OOPS से पहले pro. procedural language में लिखे जाते थे। OOPS abstraction, encapsulation, inheritance और polymorphism जैसे सिद्धांतों पर कार्य करता है।

- Class does not occupy memory.
- Class is a group of variables of different data types and a group of methods.
- A class in Java can contain:
  - → Data member
  - → method
  - → Constructor
  - → Nested class

#

Class	objects
fruits	Apple Banana Mango

Ex →

```
Class {
{
     → Data members
}
```

Example →

```
Class Student
{
    String name = "Navya";
    int Rno = 12689;
  public void Showdata()
  {
    System.out.println("sleeping");
    System.out.println(name);
  }
}
```

Date: / / Page No.

real-world-entity

Object :-
- An object in Java is a basic unit of object-oriented programming and represents real-life entities.
- Objects are the instances of a class that are created to use the attributes and methods of a class.
- A typical Java program created many objects which as you know, interact by invoking methods. An object consists of:
  1) state
  2) Behavior
  3) Identity

In Java an object is created from a class object is a variable which datatypes is one another class. It is a Referene variable.

To create an object of student specify the class name followed by the object name and use the keyword new.

keyword for the object creation

```
Student obj = new student();
obj.showdata();
```

constructor

- when a object of a class is created the class is said to be 'instantiated.
- A single class my have any number of instanc
- All the instances share attributes the state are unique for each object.

Date: / / Page No:

Class	Object
1) Class is the blueprint of an object. It is used to create objects.	An objects is an instance of the class
2) No memory is allocated when a class is declared.	memory is allocated as soon as an object is created
3) A class is a group of similar objects.	An objects is a real-world entity such as a book, car, etc.
4) Class is a logical entity	An object is a physical entity.
5) A class can only be declared once.	objects can be created many times as per requirement.
6) An example of class can be a car	Objects of the class can be BMW, Mercedes Ferrari, etc.

☆ Why Java is not a purely objects oriented language?

- pure object oriented language or complete Object oriented language are fully objects

Wrapper class → Wrapper class provides the mechanism to convert primitive into object and object into primitive. In Java you can use Integer, Float, etc instead of int, float etc.

Small talk is a pure object oriented programming language

Date: Page No.

oriented language that supports or have features that treats everything inside the program as objects.

- It does't support primitive datatype (like int, char, float, bool etc).

1) primitive data type -
2) The static keyword -
3) Wrapper class -

Java Constructors

- Java Constructors or Constructors in Java is a terminology used to construct something in our program.
- A constructors is a special method that is used to initialize objets.
- The constructor is called when an objects of a class is created.
- It can be used to set initial values for object attributes.
- The constructors name is same name of class.

What are constructors in Java?

- In Java a constructor is a block ~~is~~ is a of codes similar to the method.
- It is called when an instance of the class is created.
- At the time of calling the constructor memory for the objects is allocated in the

WORA → Write once Run Anywhere. we can code returns on any platform can run on any platform.

memory.

- Every time an object is created using the new() keyword at least one constructor is called. • constructor is does not return type.

How Java constructors are different from Java method?

- Constructors must have the same name as the class within which is defined it is not necessary for the method in Java.
- Constructors do not return any type while method(s) have the return type or void if does not return any value.
- Constructors are called only once at the time of object creation while method(s) can be called any number of times.

```
class Animal
{
Animal() {
        System.out.print("sonam");
}
```

(object by calling)

two type of constractor :-

1) Default constructor
2) Parameterized constru---
3) Copy constructor,

(1) Default Constractor in Java :- 1) • A constructor that has no parameters is know as default the

class A → default constructor invisible - 31324

```
A()
{
  S.O.P("shivani");
}
```

Constructor.

- A default constructor is invisible.
- And if we write a constructor with no arguments, the compiler does not create a default constructor. It is taken out.
- It is being overloaded and called a parameterized constructor.
- A creat a object default constructor are by default called.
- The default constructor changed into the parameterized constructor.
- But parameterized constructor can't change into the default constructor.
- The default constructor can be implicit explic It.
- If we manually write a constructor the implicit one is overrided.

2) Parameterized Constructor :- A constructor that has parameters is know as parameterized constructor. If we want to initialize fields of the class with our own values then use a parameterized constructor.

constructors know as constructor overloading. Just like methods, we can overload costructors for creating objects in different ways.

The compiler differentiates constructors on the basis of the number of parameters types of parameters and order of the parameters.

Date: Page No.

```
class A
    String Name;
A(String Name) {
        this.Name.Name;
}
P S V.M ( )
A ob = new A("shivi");
    obj.Name;
```

3) Copy constructor :- Unlike other constructors copy constructor is passed with another objects which copies the data available from the passed objects to the newly created object:

```
class person {
 String name;
 int id;
person ( String name, int id) {
 this.name = name;
  this.id = id;
}
person(Person obj2) {
    this.name = obj2.name;
    this.id = obj2.id;
}
}
Class Driver {
P.S.V.m(String args[]):{
```

restricting - सीमित या प्रतिबंधित

Date / / Page No.

```
Person obj = new Person("Vani", 123);
S.o.p ("name" + obj.name + " id " + obj.id);
person obj2 = new person(obj);
S.o.p ("name" + obj2.name + " and id " + p2.id);
```

NOT -> A constructor ~~ca~~ in Java is a special method used to initialize objects.
yes, a constructor can be declared private. A private constructor is used in restricting object creation.

Inheritance in Java :- • Java : Inheritance is an important pillar of OOPs.

- Inheritance in Java is a mechanism is which (one object acquires all the properties and behaviors of a parent ~~object~~ class.).
- one class is allowed to inherit the features of another class.
- In Java Inheritance means creating new classes based on existing ones.
- A class that inherits from another class can reuse the methods and fields of that class.
- In addition you can add new fields and methods to your current class a well.

How to use Inheritance in Java :-

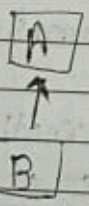
- The extends keyword is used for Inheritance in Java.

Date: / / Page No:

Using the extends keyword inhicates you are derived from an existing class. In other words "extends" refers to increased functionality.

- Single Inheritance.
- Multilevel Inheritance
- Hierarchical Inheritance
- Multiple Inheritance
- Hybrid Inheritance.

1) Single Inheritance :- • In single inheritance, a sub-class is derived from only one super class.
• It inherits the properties and behaviour of a single-parent class.
• Sometimes, it is also known as simple inheritance. In the blow figure 'A' is a parent class and 'B' is a child class.
• The class 'B' inherits all the properties of the class 'A'.



Single inheritance Example →

```
Class Animal {
void eat() {
System.out.println("eating");
}
```

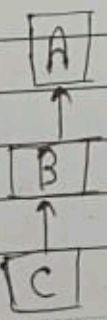
```
// extends -> keyword
class Dog extends Animal
{
  void bark() {
    System.out.println("barking...");
  }
class TestInheritance {
public static void main (String[] args)
{
  Dog d = new Dog();
    d.bark();
    d.eat();
}
}
```

O/P -> barking...
eating

2) Multilevel Inheritance :-

- In multilevel Inheritance, a derived class will be ~~Inheritance~~ Inheriting a base class, and as well as the derived class also acts as the base class (super class) for other classes.
- In the below image, class 'A' serves as a base class for the derived class 'B', which in turn serves as a base class for the derived class 'C'.
- In Java, a class cannot directly access the grandparent's members.

base class
↓
super class



Date: / / Page No:

multilevel inheritance Example ->

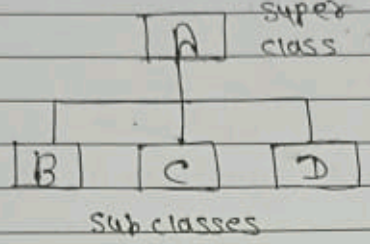
```
class Animal {
void eat() {
System.out.println("eating...");
}
}
class Dog extend Animal {
void bark() {
System.out.println("barking");
}
}
class BabyDog extend Dog {
void weep() {
System.out.println("weeping");
}
}
Class TestInheritence2 {
public static void main(String[] args)
{
BabyDog obj = new BabyDog();
obj.weep();
obj.bark();
obj.eat();
```

O/P ->
weeping
barking
eating

Date: / / Page No:

3) Hierarchical Inheritance :-

In Hierarchical Inheritance, one class serves as a super class (base class) for more than one subclass. In the below image, class A serves as a base class for the derived classes B, C and D.



Hierarchical Inheritance Example :-

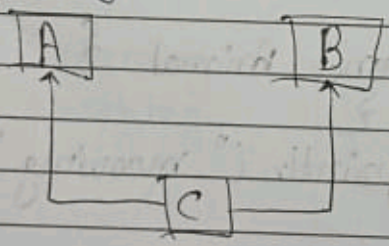
```
Class Animal {
Void eat() {
  System.out.println(" eating ");
}
}

Class Dog extend Animal {
Void bark() {
  System.out.println(" Barking... ");
}
}

Class cat extend Animal {
Void meow() {
  System.out.println(" meowing ");
}
}
```

```
Class TestInheritance3 {
public static void main(String[] args)
{
Dog D = new Dog();
   D.bark();
   D.eat();
Cat C = new Cat();
   C.meow();
   C.eat();
```

op-> Barking
eating
meowing
eating

4) Multiple Inheritance (Through Interfaces):-

- In multiple inheritance, one class can have more than one superclass and inherit feature from all parent classes.
- please not that Java does not support multiple inheritance with classes.
- In Java we can achieve multiple inheritances only through Interface.



Date: / / Page No:

5] Hybrid Inheritance :-

- It is a mix of two or more of the above types of inheritance.
- Since Java doesn't support multiple inheritance with classes, hybrid inheritance involving multiple inheritance is also not possible with classes.
- In Java we can achieve hybrid inheritance only through Interface if we want to involve multiple inheritance to implement Hybrid inheritance.
- It can be achieved through a combination of multilevel inheritance and Hierarchial inheritance with classes, Hierarchical and single inheritance with classes.
- Therefore it is indeed possible to implement Hybrid inheritance using classes alone, without relying on multiple inheritance type.
- Hybrid means consist of more than one. Hybrid inheritance is the combination of two or more types of Inheritance.

Example :-

```
class Grandfather {
    public void show() {
        System.out.println("I am Grandf
                            ather");
    }
}
```

Date: / / Page No.

```
class father extends Grandfather {
    public void show() {
    System.out.println("I am father");
    }
}

class son extends father {
public void show() {
System.out.println("I am son");
}
}

public class Daughter extends father {
    public void show();
System.out.println("I am a daughter");
}
```

```
    Daughter ob = new Daughter();
    ob.show();
}
}
```

O/P → I am daughter.

★ Java is-A type of Raletionship :-

is-A is a way of saying: this object is a type of that object. Let us see how the extends keyword is used to achieve inheritance.

★ What can Be Done in a subclass?

In sub-class we can inherit members as is.

Date: / / Page No:

replace them, hide them, or supplement them with new members :-

- The inherited fields can be used directly, just like any other fields.
- We can declare new fields in the subclass that are not in the superclass.
- The inherited methods can be used directly as they are.
- We can declare new methods in the subclass that are not in the superclass.

★ Advantages of Inheritance in Java :-

- Code Reusability :- Inheritance allows for code reuse and reduces the amount of code that needs to be written. The subclass can reuse the properties and methods of the superclass reducing duplication of code.
- Abstraction :- Inheritance allows for the creation of abstract classes that define a common interface for a group of related classes. This promotes abstraction and encapsulation, making the code easier to maintain and extend.
- Class Hierarchy :- Inheritance allows for the creation of a class hierarchy, which can be used to model real-world object and

their relationships.

- Polymorphism :- Inheritance allows for polymorphism, which is the ability of an object to take on multiple forms. subclasses can override the methods of the superclass, which allows them to change their behaviour in different ways.

* Conclusion :- let us check some important points from the article... are mentioned below:

- Default superclass :- Except objects class, which has no superclass, every class has one and only one direct superclass (single inheritance). in the absence of any other explicit superclass every class is implicitly a subclass of the Object class.

- Superclass can only be one :- A superclass can have any number of subclass. But a subclass can have only one superclass. This is because Java does not suport multiple inheritance with classes. Although with interfaces, multiple inheritance are supported by Java.

- Inheriting Constructors :- 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, but the constructor of the superclass can be invoked from the subclass.

- Private member inheritance :- A subclass does not inherit the private members of its parent class. However, if the superclass has public or protected methods (like getters and setters) for accessing its private fields, these can also be used by the subclass.

Q What is Is-A Relationship in Java ?

Ans. A relation in Java means different relation between two or more classes.

for example if a class Bulb inherits another class Device then we can say that Bulb is having is-a relationship with device which implies Bulb is a device.

In Java we have two type of relationship

(1) Is-A relationship :- whenever one class inherits another class is called an Is-A relationship.

Date: / / Page No.

② Has-A relationship :- whenever an instance of one class is used in another class it is called "HAS-A" relationship.

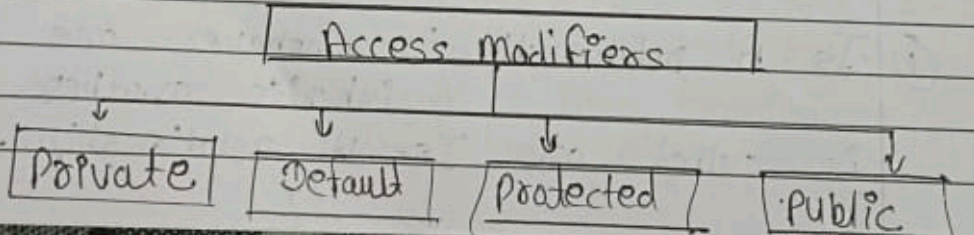
☆ Access Modifiers :-

- In Java, Access modifiers help to restrict the scope of a class, Constructor, Variable, method, or data member.
- It provides security, accessibility, etc. to the user depending upon the access modifier used with the element.
- Let us learn about Java Access Modifiers, their types and the uses of access modifiers in this article.

There are four types of access modifiers available in Java:

- Default - No keyword required
- Private
- Protected
- Public

Access Modifiers in Java



for classes you can use public or default:

Public => The class is accessible by any other class.

default => The class is only accessible by classes in the same package. This is used when you don't specify a modifier.

for attributes methods and constructors you can use the one of the following.

public => The code is accessible for all classes.

Private => The code is only accessible within the declared class.

default => The code is only accessible the same package. This is used when you does't (don't specify a modifier.

Protected => The code is accessible in the same package and subclasses.

1) Default Access Modifier :

- When no access modifiers is specified for a class, method, or data member - It is

said to be having the default access modifier by default.

- The data members, classes, or methods that are not declared using any access modifiers having default access modifiers are accessible only within the same package.
- In this example we will create two package and the classes in the packages will be having the default access modifiers and we will try to access a class from one package from a class of the second package.

2) Private Access Modifier :-

- The private access modifiers is specified using the keyword private.
- The methods or data members declared as private are accesible only within the class in which they are declared.
- Any other class of the same package will not be able to access these members.
- Top-level classes or interfaces can not be declared as private because private means "only visible within the enclosing class".
- In this example, we will create two classes A and B within the same package P1. We will declare a method in class A as private and try to access this method

from class B and see the result.

3) Protected Access Modifier :-

- The protected access modified is specified using the keyword protected.
- protected means" only visible within the enclosing class and any subclass" Top level class cant be protected.
- The methods or data members declared as protected are accessible within the same package or subclasses in different packages.

4) Public Access modifier :-

- The public access modifier is specified using the keyword public.
- The public access modifier has the widset scope among among all other access modifiers.
- classes, methods, or data members that are declared as public are accessible from everywhere in the program.
- There is no restriction on the scope of public data members.

	Default	Private	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non subclass	Yes	No	Yes	Yes
Different package non - subclass	No	NO	NO	Yes

Different package subclass	No	No	yes	yes

When we use this keyword.

This ⇒ 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 constractor parameter).

this can also be used to —

- Invoke current class constructor.
- Invoke current class method.
- Return the current class object
- Pass an argument in the method call
- pass an argument in the constructor call

Ex. — class main {
int model year;
String modelName;
;

public main (int modelyear, String model name) {
this. modelyear = modelyear;
this. modelyeoname = modelname;
}

Public static void main(String [] args)
{
Main obj = new Main(1969, "mustang");
System.out.println(obj.modelyear + "." + modelname);
}
}

O/P => 1969., mustang

Polymorphism :-

Polymorphism in Java is a concept by which we can perform a single action in different ways.

- Polymorphism is derived from 2 Greek words: Poly and morphs.
- The word "poly" means many and "morphs" means forms.
- So polymorphism means many forms.

There are two types of polymorphism -
Compile - time polymorphism
Runtime polymorphism.

- we can perform polymorphism in Java by method overloading and method overriding.
- If you overload a static method in Java it is the example of compile time polymorphism.

Compile time polymorphism ⇒

It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.

Note → But Java dont support the operator overloading.

method overloading → when there are multiple method (functions) with the same name but different parameters, then these functions are said to be overloaded.
- methods can be overloaded by changes in the number of arguments ~~or~~ and a change in the type of arguments.

Example —

```
class Helper {
    static int multiply(int a, int b)
    {
        return a*b;
    }
    static int multiply(int a, int b, int c)
    {
        return a*b*c;
    }
}
class Driver {
    public static void main(String[] args)
    {
        System.out.println(Helper.mulpiply(2,4));
```

System.out.println(Helper multiply(2,7,3));

Output :- 8
42

Note :- ① The Return types of the above methods are not the same.

② Overloaded methods may have the same or different return types but they must different in parameter.

Constructor overloading :- Java supports constructor overloading in addition to overloading methods. In Java overloaded constructor is called based on the parameters specified when a new is executed. Constructor named are same but parameters different.

Example :-
```
class Box {
    Box() {
        System.out.println("Hello");
    }
    Box(int a)
    {
        System.out.println("age");
    }
    Box(int a, int b) {
        System.out.println("number");
    }
}
```

class Driver1 {
{
Public static void main (String [] args)
{
Box ob = new Box (12);
}
}

Output :- age

* Runtime polymorphism :-

It is process in which a method call to the overridden method is resolved at Runtime.

Overriding :- Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes.

"when a method in a subclass has the same name, the same parameters or signature and the same return type as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

note :- Constructor is not support overriding.

Example :- class Animal {
public static void displayInfo() {
System.out.println("I am an Animal");
}
}

class Dog extents Animal {
public void displayInfo() {
System.out.println("I am a dog");
}
}

class main {
public static void main(String[] args)
{
Dog d1 = new Dog();
d1.displayInfo();
}
}

Note :- Both the superclass and the subclass must have the same method name, the same return type and the same parameter list.

Super Keyword :-

- "Super Keyword to refer immidiate parent class instance variable.
- Refers to superclass objects.
- It is used to call methods of the super class that is overridden in the Subclass.

- It is used to access attributes of the superclass if both superclass and subclass have attributes with the same name.
- It is used to access the superclass constructor.
- The super keyword in Java is a reference variable which is used to refer immediate parent class object.
- Whenever you create the instances of subclass an instance of parent class is created implicity which is refered by super keyword.

used to super keywords →

- Super is used to refer immediate parent class instance variable.
- Super can be used to invoke parent class method
- Super is used to invoke parent class constructor.