Hand written Core Java Notes (Part-1/2)

Santosh Kumar Mishra
SDE @Microsoft

Core Java (J2SE)

* **Software:-** set of programs meant for performin certain types of operations.

In the industry, based on the type of operations we perform, the softwares are classified into 3 types they are,

1. system software
2. Application software.
3. Internet software.

1 System software:-

a) The basic aims of system s/w are developr of drivers (printers, scanners, joystick, etc)

b) Development of language compilers, Interpre loaders, linkers, assemblers, etc)

c) Development of real time operating syste (windows, XP, UNIX, Linux, MAC OS, OS/2, solaris, etc)

According to industry standards to develop the system s/w we use the languages like C and ALP (Assembly langu programming).

These languages becomes popular for develop of system software because They have high capabilities to interact with hardwa devices directly.

GUI Appl^n = Look & feel based appl^n.

* 256 Header files in C lang.
* MNC uses C & ALP developing system s/w for

System software:-

Defination:-

A system s/w is a program running in the background of the system for providing internal services for effective execution of our applications.

2/7/13

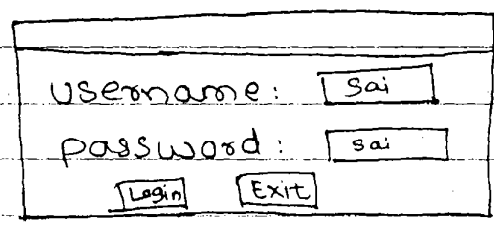
(2) Application software:-

The aim of appl^n s/w is to develop s/w for the organisations

In s/w development to develop the appl^n s/w we use 2 types of technologies.

(I) Front End Technologies:-

The aim of front end technologies is to develop GUI applications (Look and feel based appl^n are called GUI appl^n)



To develop GUI appl^n we have 2 types of front end technologies they are;

(a) Developer 2000 (Oracle corp)
Visual Basic (Microsoft corp)

Middle level lang] compilers used = Fast

Low level lang ⇒ Interpreters used (8086) ⇒ slow.

If we develop any GUI appl^n by using the above technologies then such GUI appl^n runs only on Microsoft provided operating systems (windows xp, NT, etc) but not directly running on Non Microsoft operating systems (UNIX LINUX, MAC OS, Solaris, etc). Hence this type of technologies are called platform dependent ~~tec~~ front end technologies.

(b) AWT (Abstract windowing toolkit)
Applets
swings } → sun microsystem - Java b

If we develop any GUI appl^n with the above concepts then such GUI applications runs on every OS without considering their vendors. Hence these type of technologies are called platform Independent front end Technologies.

(II) Backend Technologies:-

The aim of backend technologies is to store the data permanantly (known as data persistancy).

In Industry data persistancy can be achieved in two ways they are
(a) Through files.
(b) Through database slw.

If we store data permanently in the form of files then data of files can be manipulated by any unauthorized user. because file concept of any programming language is not providing security in the form of username & passwords.

Hence files are not recommended to used in the industry for archiving security.

Industry is always recommended to achieve the data persistancy in the form of database s/w.

so every database s/w is treated as backend technology.

The following table gives database s/w name and corresponding vendor

Database s/w		Vendor
oracle	⇒	oracle corp.
MYSQL	⇒	SUN MICROSYS.
DB2	⇒	IBM.
SQL server	⇒	microsoft.

Q. Define standalone applications and distrib appl^n.

Standalone applications:-
A standalone appl^n is one which runs i the context of local disks and whose results are accessible in the same m but not accessible across the globe.

e.g: All the projects / appl^n of system sl or Appl^n s/w are comes under standal appl^n.

Distributed appl^n:-
Are those which are running in the context of browser / www whose results are accessible across glo

e.g: All websites (www.gaboo.com)
Credit card, Debitcard processing, e

3/7/13

3 Internet ~~Applicati~~ software:-

Internet s/w basic aim is to develop distributed applications.

We know that distributed appl^n are tha which are running in the context of browser / www & whose results are sharab across the globe

To develop the distributed appl^n we have following languages/ Technologies,

(a) JAVA (Developed by SUN microsystem handed over by oracle corp)

(b) .NET (Developed by Microsoft INC, USA)

"Hence JAVA is one of the Internet S/W developed by SUN microsystem and released to the real industry and meant for development of each and every application but it becomes more popular for the development of distributed application."

* Architecture required for development of Distributed Applications:-

According to industry standards, to develop any distributed application we have a well known architecture called Client-Server arch.

Client Server arch. contains 3 sub arch. They are,

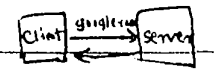
1. 2-tier Arch.
2. 3-tier Arch
3. n-tier Arch.

Every 3 tier arch must be n tier to archieve security.

① 2-tier Architecture:-

This arch contains 2 types of programs they are,

ⓐ set of client side programs.
ⓑ set of server side programs.

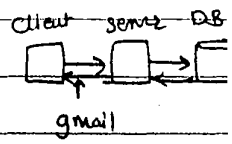


e.g: Google.com

② 3- tier Architecture:-

This arch contains 3 types of programs they are,

ⓐ set of client side programs.
ⓑ set of server side programs.
ⓒ set of Database software.

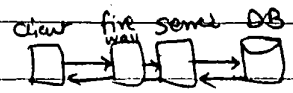


e.g: mail.yahoo.com, gmail.com, ATM based appl^n, etc

③ n-tier architecture:-

This arch contains 4 programs they are

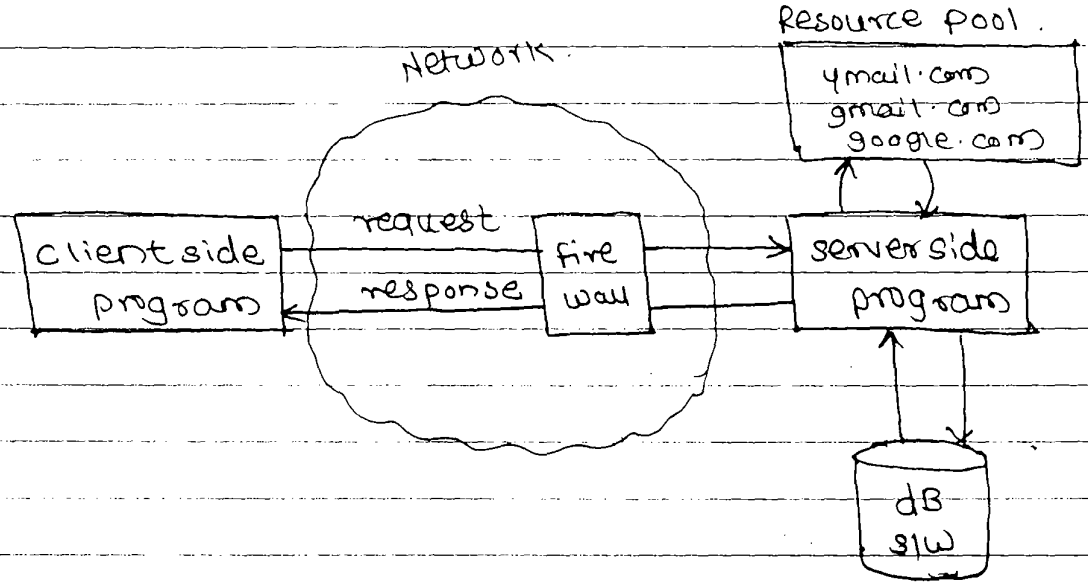
ⓐ set of client side programs.
ⓑ set of firewall programs
ⓒ set of server side programs
ⓓ set of database software.



Hence in the real world projects we have following people to work

ⓐ set of client side programmers
ⓑ set of server side programmers
ⓒ set of firewall developers
ⓓ set of database programmers.

Following diagram gives flow in the client server architecture.



* Definations:-

① Client:- A client is a program which is always used for making request to the server side program for getting the services

② Server:- A server is also a program which will receive client request, process the client request and gives response back to multiple clients concurrently (parallel accessing and ||el processing)

The basic advantage of server s/w in the real world is to provide concurrent access.

The following table gives server s/w names and their vendors.

Server S/W Name	Server S/W vendor Name
Tomcat	Apache Jakartha S/W foundation.
~~Webshare~~ Weblogic	BEA System
Websphere	IBM
Oracle	Oracle Corp.
Glassfish	SUN microsystem
IIS (Internet Information server)	Microsoft.

(3) **Resource pool:-**
Resource pool is one of the layer present in server S/W & in which we place distribut applications.

(4) **Firewall:-**
Firewall is one of the security program which will validate wheather client request is virus oriented or virus free.

(5) **Database:-**
Database is one of the data persistancy layer and it provides effective security to the confidential data and it prevents unauthorized modifications.

⑥ Protocol :-

protocol is set of rules which are used for exchanging data between client & server side applications.

In the current industry internet word is using HTTP protocol.

4/7/13

* History Of Java:-

JAVA is one of the programming lang/Tech. used in industry for each and every appl^n development & it becomes more popular for development of distributed applications.

The Java S/w developed at a sun microsystem under the guidance of James Gosling & others.

The Java S/w started its development in early of the year 1990 & released to the industry in middle of the year 1991 on the name of "OAK" which is one of the original name of Java & scientifically it is one of the "tree name" & OAK has taken 18 months to ~~comp~~ develop.

The S/w OAK was popular for developing s/w for electronic, Mechanical & automobile devices.

In other words the S/W OAK was able to fulfill few requirements of the industry & unable to fulfill some other requirements of industry. This is an industry opinion.

SUN microsystem developers, under the guidance of James Gosling the S/W OAK was revised or reengineered & released to the real industry in the year 1995 on the name of "Java" scientifically Java is one of the coffee seed nar

The Java S/W is available in the real industry in 3 categories They are,

ⓐ J2SE [Java 2 Standard Edition] (client
ⓑ J2EE [Java 2 Enterprise Edition]
ⓒ J2ME [Java 2 micro / Mobile Edition]

In the industry J2SE is used for developing client side application, J2EE is used for developing server side appl^n, and J2ME is used for developing Mobile / wireless based appl^n

The following table gives Java technology version, Java lang. version.

Java Tech. version	Java lang. version
JAVA	JDK 1.0
	JDK 1.1
JAVA2	JDK 1.2
	JDK 1.3
	JDK 1.4

Mostly Industry is using **JDK 1.5** Nowadays.

JAVA5	JDK 1.5 [Tiger Slw]
JAVA6	JDK 1.6
JAVA7	JDK 1.7

In the current Industry J2SE must be called as JSE or JAVASE, J2EE must be called as JEE or JAVA EE, and J2ME must be called as JME or JAVAME.

JAVA is one of the open source code and it can be downloaded freely from www.sun.com or www.oracle.com.

Q. What are differences betn HTTP & FTP.

HTTP	FTP
It is one of the Acknowledgement oriented protocol. (send email ack to sender)	It is one of the non ACK. oriented protocol.
It is one of Stateless protocol (maintains the ~~session~~ identity for certain time)	Stateful protocol. (maintains identity of client forever)
Belongs to TCP	Belongs to UDP

Q. Define stateless & stateful protocol.

Stateless protocol:-

A Stateless protocol is one which maintains an identity of a client for the limitted period of time.
eg: Http. (Best suited for ATM & logins)

Stateful protocol:-

A stateful protocol is one which maintains an identity of client forever.
eg: ftp.

5/7/13

* Features/Buzzwords of JAVA:-

Features of a language are nothing but the services or facilities or functionalities provided by language developers to the industry programmers.

SUN microsys. has provided 13 features in JAVA programming to the industry programmers they are,

1. Simple
2. platform Independent
3. Architecture Neutral.
4. portable
5. Multithreaded
6. Networked
7. Distributed
8. High performance
9. Highly interpreted
10. Robust (strong)
11. secured
12. Dynamic
13. object oriented programming lang. (OOPL)

expert says pointers in C are complex

C → without pointers :- appl^n development time is less.
appl^n execution time is more

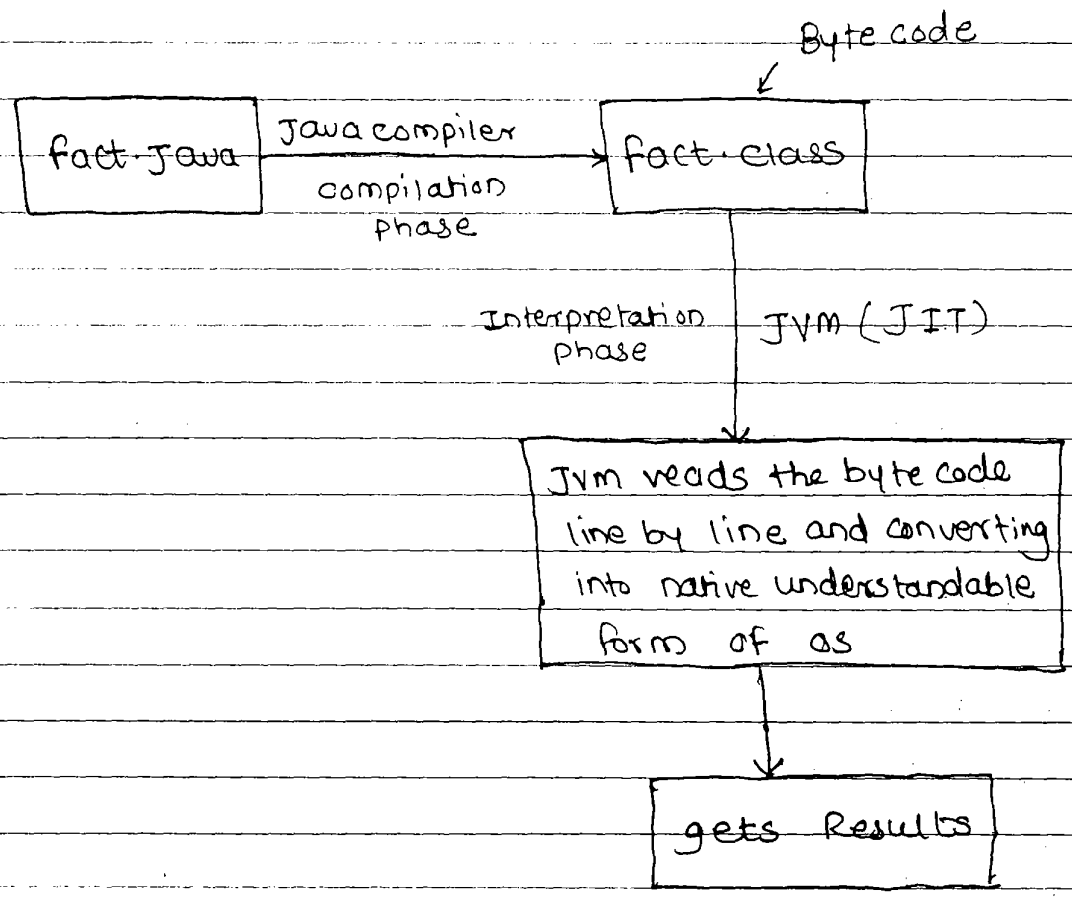
→ with pointers :- appl^n development time is more.
appl^n execution time is less

① Simple :-

Java is one of the simple programming lang because of following 4 factors,

(a) Java programming eliminates complex concept called pointers so that appl^n development time is less and appl^n execution time is less because of magic of <u>byte code</u>

Let us consider the following diagram which gives an awareness about compilation and interpretation phase of Java



* Definations:-

① Byte code:-

Byte code is set of optimized instructions generated by Java compiler during compilation phase and nature of byte code is much powerful than ordinary pointer code of C, C++ languages.

② JVM:- (Java virtual machine)

JVM is the program developed by SUN microsys. & supplied as a part of Java SW (SDK) & whose role is reading the byte code line by line & convertin into native understanding form of OS.

Byte code is OS independent and JVM is OS dependent.

③ JIT:- (Just-In-Time)

JIT is one of the program developed by SUN and added as a part of JVM whose role is to speed up the interpretation phase by reading entire section of bytecode & converting into native understanding form of OS.

Java programming is one of the compiler and interpreted based programming language.

" C is not able to deallocate unused memory.

Java uses Dynamic memory allocation ⇒ memory allⁿ at runtime

C, C++ ⇒ static ⇒ allocate memory at Compile time

16

allⁿ = allocation

C++ can deallocate unused memory by usind destructors.

6/7/13

(b) Java programming contains inbuilt facility called garbage collector for collecting unused memory space which will improve performance of Java application.

Q. Defination:-

Garbage Collector:-

A garbage collector is one of the System background Java program which is running along with our regular Java program for collecting unused/ unreferenced memory space. and it improves performance of every Java application.

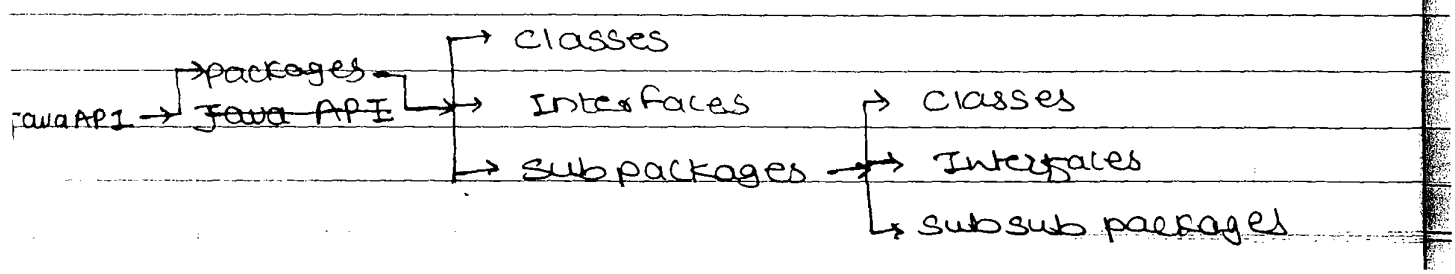
Garbage collector program running internally for regular periodical intervals of time.

(c) Java programming contains rich set of API (appl^n programming Interface).

(Just like header files in C & CPP)

API:-

An API is a collection of packages, a package is a collection of predefined classes, Interfaces & subpackages. A subpackage in turns contains classes, interfaces and subsub packages etc.



C	Int	2	4
	Float	4	8
	⋮		
	Char	1	1

as is ~~state~~ having two times more space than DOS as except char it has same space.

(d)= Java programming language provides user friendly syntaxes which makes user to develop quicker appln development without errors.

② Platform Independent :-

A lang / Tech. based appln are said to be platform Independent if & only if whose appln's runs on each and every as without considering their vendor

In order to say a language is platform independent it has to satisfy following properties.

(1)= The lang. / Tech. related datatypes must take same amount of memory space on each & every as.

(2)= The lang / Tech. must contains some special internal program & whose role is converting native understanding form of one as into native understanding form of another as.

The languages like c, c++, pascal, COBOL, etc (non Java programming lang) are treated as platform dependent. Because they never satisfie above 2 properties.

The language like Java & whose related appln are treated as platform Independent because Java satisfies above 2 properties.

DOS ⇒ MOZARTA (MB)
UNIX ⇒ Embedded La
native understandable forms
format (ELF)

multitasking uses Round Robin algorithm

8/7/13 (Mon)

③ ~~Architectural Neutral~~ :-

③ Multithreading:-

The aim of multithreading concept is to provide concurrent access/execution.

A flow of control is known as thread

The purpose of thread is to execute logic of the Java program which is written in the form of user defined methods

If a Java program is containing multiple flow of controls then it is known as multithreaded.

The languages like C/C++/Pascal COBOL, etc are treated as single Threaded Modelling Languages because their execution enviornment provides single flow of control, They provide sequential execution whose appl^n takes more execution time & does not contain any library for development of multithreading based appl^n.

The languages/Technologies like Java and .Net are treated as multithreaded modeling languages because their execution enviornment provides multiple flow of control provides concurrent execution, whose appl^n

takes less execution time & provides an effecti
Library for development of thread based appl^n.

When we write a Java program there exists 2 types of threads,

ⓐ Foreground / Child Thread :-

A foreground thread is one which is always executing logic of a Java program which is written in the form of user defined methods.

ⓑ Background / Parent Thread :-

A background thread is one which is monito execution status of foreground thread.

The real time Implementation of multithreadin concept is that to develop real world server softwares such as Tomcat, weblogic (BEA systems) websphere, etc. In Other words most of the real world server s/w's developed by third party server vendors in Java language by using multithreading concepts.

Hence multithreading of Java is one of the specialized form of multitasking of operating systems.

Q. How do you Justify "Each & Every Java Program is multithreaded"?

Ans:- When we execute any Java program, the logic of the Java program is executed by

one of the thread known as foreground thread. To monitor the execution status of foreground thread, one more thread is created known as a background thread. So Java program is containing 2 threads hence every Java program is multithreaded.

(4) High Performance:-

Java is one of the high performance language because of following factors,

1) Magic of Byte code (use less execution time)
2) Automatic memory management because of garbage collector. (collects unused memory space for improving performance of Java appl^n).

(5) Highly Interpreted:-

In the initial versions of Java compilation phase is very faster and Interpretation phase is slow which is one of the industry compliments & complaints.

In the later versions of Java SUN developers has developed a program called JIT (Just In Time compiler) and added as part of JVM & whose role is to speed up Interpretation phase by reading the entire section of the bytecode & converting into native understanding form as as.

Conclusion:- In the current versions of Java Interpretation phase is very fast even compared with compilation phase. Due to this Sun developer has populated Java is one of the highly Interpreted programming language.

9/7/13

Robust (Strong) :-

Java is one of the strong programming lang. because It contains a facility called exception handling.

The languages like C, C++, pascal, COBOL, etc & whose applications are treated as weak because these languages does not contain a facility called exception handling.

Whereas Java & whose applications are treated as strong because Java contains a error handling facility called Exception Handling.

Q. Define Exception ? What is meant by Exception Handling ?

Runtime error of Java program is known as an exception.

Every Exception of Java program generates by default system / Technical error messages which are not understandable by application users. Hence industry is not recommended to generate system error messages

But recommended to generate user friendly error messages.

The process of converting system error messages into user friendly error messages is known as Exception Handling.

(7) Dynamic :-

In any programming lang. we have 2 types of memory allocation techniques. They are,

a) Static memory allocⁿ
b) Dynamic memory allocⁿ.

In static memory allocⁿ memory space is created / allocated for input of the program at <u>compile time</u>.

Due to static memory allocation we get the following limitations.

i) Waste of memory space.
ii) Loss of Data.
iii) overlapping of existing data.

To avoid these problems industry is highly recommended to follow dynamic memory allocation.

In Dynamic mem. allocⁿ the memory space is allocated for i/p of program at runtime

Java programming follows only Dynamic mem. allocation but not static memory allocations

Since Java is following Dynamic mem alloc^n, SUN developers has populated that Java is one of the Dynamic programming language.

(8) Secured :-

Security is one of the principle used in the industry projects for protecting Confidential information from unauthorized users.

In non Java programming lang. (C, C++, pascal, etc there is no existing library for security

Non Java lang. programmers needs to write their own code for providing the security & it may be complex hence every non Java programming lang. is unsecured.

In the case of Java programming we have a dedicated API (library) for dealing with a security i.e. in Java projects, a Java programmer need not to write any security program on their own & they can use readily available security program present in the existing API (like Javax.security.*;)

Since it is Containing readily available security API SUN developers has populated Java is one of the secured programming language

10/7/13

(9) Architectural Neutral :- (Irrespective of processor)

A lang / Tech. and whose related applications are said to be Architectural Neutral iff They runs on every processor without considering their architectures & vendors.

To say a lang / Tech. is arch. neutral, it has to satisfy the following property

1) The lang / Tech. must contain some special programs & whose role is converting the factor / key of one processor to factor / key of another processor.

The languages like C, C++, Pascal, COBOL, etc & whose related applications are treated as architectural Dependent because these lang. does not satisfy above property.

The lang. like Java & whose applications are treated as architectural Neutral because Java satisfies above property

format of exe file:-

Binary data	MZ format	16 bit hex number

← different ~~unique~~ for each processor

The binary data of code

MOZARTA format which is native understandable format of OS.

(10) Portable :- (runs on every os & every processor).

A portable Technology & whose application runs on every os and on every processor without considering their architectures & vendors

portability = Platform Independent + Arch. Neutral

Every non Java language & whose appl^ns are treated as non portable. whereas Java lang. & whose appl^ns are said to be portable.

Industry is prefering to use portable language for development of most of the distributed applications & not prefering Non portable languages.

11/7/13

* Data types:-

The purpose of data types is to allocate sufficient amount of memory space for the input of the program in the main memory of the computer either by following static memory allocation or Dynamic memory alloc^n.

In every programming language data types are classified into 3 types. They are,

① fundamental / pre defined datatypes:-
(primary / primitive / core / scalar / built-in)

② Derived data types.

③ user defined / secondary / custom / programmer / Referential data types.

① Fundamental data types are those which are developed by language developers & supplied as a part of their software & whose variables allows us to store single value but they never allows us to store multiple values of same type.

e.g:- int a;

a = 10; // valid.

a = 10, 20, 30; // Invalid.

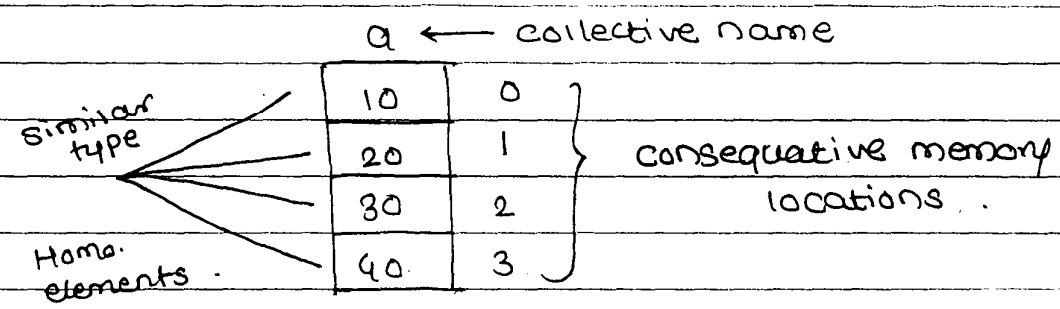
② Derived data types are those whose variables allows us to store multiple values of same type but they never allows us to store multiple values of diff type.

e.g: int a[] = {10, 20, 30}; // valid.
int b[] = {10, 'A', 23.5}; // Invalid.

* In every programming language the concept of Arrays comes under derived datatypes.

Array:-

Defn:- An array is a collective name given to a group of consequtive mem. locations which are all refered by similar | Homogeneous type of elements



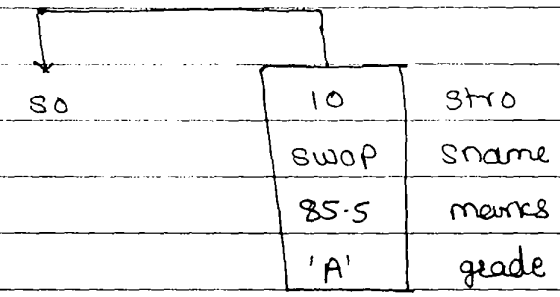
(3) User defined data types are those whose variables allows us to store multiple values either of same or different types or both types by using language facilities.

e.g: In 'c' programming to create user defined datatype we use ~~stuc~~ structures (Struct), Unions (Union), Enumerations (Enum).

simillarily in 'Java' programming we use classes (Class), Interfaces (Interface), Enumerati (Enum), etc. for development of user defined data types.

e.g:

```
Student so = new Student();
```



12/7/13

* Java Fundamental Data Types:-

In Java programming we have 8 fundamental data Types which are classified into 4 categories they are,

1) Integer category data types
2) Float category data types.
3) Character category data types.
4) Boolean category Data types.

① Integer category Data Types:-

The aim of Integer cat. Data types is to store integer data in the main memory of the computer by allocating sufficient amount of memory space.

Integer category contains 4 data types and they are given in the following table.

Integer Cat. Data Type → byte, short, int, long

Sl/No	datatype	Size in byte	Range
1	byte	1	+127 to −128
2	short	2	+32767 to −32768
3	int	4	+ x to −(x+1)
4	long	8	+y to −(y+1)

If any category of data type contains multiple data types then it is mandatory to Java programm to choose an appropriate data type.

For choosing an appropriate Data type we need to calculate the range of a data type. Range makes us to understand how much it can store at maximum side & How much it can store at negative side.

The following formula will be used for calculating range of any data type.

Range of any data type = [The number of bits available in the lang. which is understandable by computer]

[number of bits occupied by a perticular data type]

(2← As computer understands only binary la

At -ve side one bit is more for storing sign bit.

16 bits = 2 × 8

e.g: Range of short data type = $(2)^{16 \text{ bits}}$

= 1 to 65536

= 0 to 65535

= +32767.5 to -32767.5

= +32767 to -32768

15/7/13 (Mon)

② Float category Data types:-

The basic aim of float category data-types is to store real constant values (π), etc) in the main memory of the computer by allocating sufficient amount of memory space.

This category contains 2 types of datatypes They are given in the following table,

datatype	Size (in bytes)	Range	Number of Decimal places.
float	4	+x to -(x+1)	7
double	8	+y to -(y+1)	15

In some books number of decimal places are given 8 & 16 for float & double which are including dot (.) also. The shown above are excluding it.

Float:-

If we store any real constant values in a variable of float data type then such real constant value stored in the main memory in such a way that after the dot (·) it takes 7 decimal places. And the real constant value must be followed by a letter 'f' otherwise we get compile time error.

The following calculations shows (IEEE 754: floating point calculation) how real constant values stored in the main memory in bit representation of float datatype.

```
float f1 = 3.4f;
```

f1

$\boxed{011 \cdot 0110011}$

7 decimal places.

for 0.4:-

0.4	
* 2	
0.8	0
* 2	
1.6	1
* 2	
1.2	1
* 2	
0.4	0

3.4

for 3:

2	3	
2	1	1
	0	1

011

011·0110

Double:-

If we store any real constant value in a variable of double datatype then the double data type stores the real constant value in such away that after the dot (·) it takes 15 decimal places & a real constant value may or may not be followed by a letter 'd'

By default any constant value is treated as high in size data type.

eg: double $d_1 = 3.4d$;

or

double $d_1 = 3.4$;

d_1

$\boxed{011.\underbrace{01100110011001 1}_{\text{15 decimal places.}}}$

In our Java program if we use any real constant value directly then such real constant value is by default treated as highest datatype in float category that is double datatype.

float $f_1 = 3.5$;

⇒ error.

Because 3.5 is by default treated as double & it can not be directly stored in a variable of float type.

NOTE:-

In 'C' language:-

e.g: ①	float a = 3.4;	② float a = 3.5;	This happens due to
	If (a == (3.4)) double	If (a == (3.5)) double	Floating point Calculation
	{ pf("OK"); }	{ pf("OK"); }	3.4:-
	else	else	011.011001 float
	{ pf("CANCEL"); }	{ pf ("CANCEL"); }	011.01100110011001 double
	pf("OVER");	pf ("OVER");	which are not equal.
			3.5:-
			011.1000000
			011.10000000000000
	ANS: CANCEL OVER	ANS: OK OVER	which are equal.
			Except 5 all other 1 to 9 are having diff

To overcome this problem we need to write f for

③ Character category Data Type:-

A character is an identifier which is enclosed within single qoutes.
eg: 'A', 'g', '$', etc.

A collection / sequence of characters enclosed within double qoutes is known as strings.
eg:- "swap", "om sai ram", etc.

To store the character data in the main memory of the computer in c, cpp, Java programmi we use a datatype called 'char'

Char data Type in c, cpp takes 1 byte beco2 c, cpp follows ASCII character set.

In Java programming char datatype takes 2 bytes beco2 Java follows UNICODE character set and Java programming is available in 18 International languages (English, Latin, Greek, etc)

Q. Define ASCII & UNICODE.

ASCII is a character set followed by those programming languages which are available in one International language called English. And whose size is 256 ~~byte~~ ($2^8 = 256$)
1 byte.

Languages like C, CPP, COBOL, follows ASCII character set as they are available in only english language.

ASCII :- American standard code for Information Interchange

UNICODE character set is one which is followed by those programming languages which are available in more than 1 International languages including English & whose size is 65536 characters ($2^{16} = 65536$)
2 bytes.
e.g: Java, .Net, etc.

* NOTE:- (Drawback of Java)

As per as English Java programmer is concerned for storing one characters it requires one byte and rest of one byte is simply wasted which is one of the negligible drawback of Java language.

(4) Boolean Category Data Type :-

The aim of this category data type is to store logical values (feeling that they are existing but in reality they won't take physical memory space in main memory)

e.g: True, false.

In Java programming logical values are stored by using boolean.

boolean datatype of Java implemented by using a general purpose register called flip flop which will store one bit of memory space (1 for True & 0 for false)

boolean data type of Java take 0 bytes of main memory space becoz boolean values are not storing in main memory but they are storing in registers called flip flops.

NOTE:-

Each and every keyword in Java must be written in small letters.

The Default value of Integer Category variables is 0 (zero).

The Default value of float Category variables is 0.0.

The Default value of character Category variables is nothing

The Default value of boolean Category variable is false.

Q. Why character category data types requires 1 byte memory space in C while 2 byte in Java ?

ANS: As C lang. supports only english language as it follows ASCII character set it requires single byte to store.
While Java uses UNICODE character set which supports 18 International languages including english.
Consider in latin A is represented as A ← 1 byte
To store such letters 2 bytes are required.
For english Java programmer 1 byte of memory space gets wasted. which is not collected by garbage collector coz it is single strip of mem. sp

In 'C' length of variable is 8.
In recent versions it is 16.

16/07/13

* Variables:-

* Importance of Variables:-

We know that by using data Types we can allocate sufficient amount of memory space in the main memory of computer and values are also stored in the memory space. To process the values which are stored in the memory space, it is mandatory to the programmer to give or qualify some distinct name to the memory spaces. These distinct names are used for retrieving values from the memory space & these names are called identifiers.

Identifier values are varying from one point of time to another point of time. so that identifiers are also known as Variables.

* Defination of variable:-

A Variable is an identifier whose values are changing during execution of the program.

Hence to write a meaningful program we require to use variables in other words without variables we can not store the data in the main memory of the computer.

* Rules for writing variables:-

out of so many number of rules of variables, the following rules are suggested for writing variables

1) first letter of variable must be an alphabet.

2) In Java programming the length of variable can be upto 32 characters.

3) No special symbols are allowed except '_' (underscore)

4) No keywords to be used as variable Names (if, while, else, final)

* Variable Declaration In Java:-

The process of allocating sufficient memory space in the main memory of the computer and qualifying memory space by variable names is known as variable declaration.

Before using the variable in our Java program they must be declared just before their usage

* Syntax:--

datatype $v_1, v_2, \ldots, v_n$;

datatype represents either fundamental, derived or user defined datatype. $v_1, v_2, \ldots, v_n$ represents Java valid variable names.

Example:-

```
                    a   b   c
int a, b, c;       [10] [20] [30]
a = 10;
b = 20;
c = a+b;
```

-OR:-

```
                    a
int a;             [10]
a = 10;
                    b
int b;             [20]
b = 20;
                    c
int c;             [30]
c = a+b;
```

* ~~Variable Declaration~~:-

* **Variable Initialization:-**

The mechanism of placing our own values without placing default values when the memory space is created for variables.

Whenever we want to give the values which are common for most of the programmer then such common values must be placed in variable during program development, hence every common / fixed value must be initialized.

Syntax:-

datatype $V_1 = Val_1, V_2 = Val_2, \ldots V_n = Val_n$;

Here $Val_1, Val_2, \ldots Val_n$ represents list of values decided to initialize in the variables $V_1, \ldots V_n$ respectively.

Example:- int a=5, b=20;
float PI = 3.1417f;

* Constants in Java:-

A constant is an identifier whose values can not be changed during execution of the program.

In Java programming to make anything as constant we use a keyword called final.

The final keyword playing an important role in Java in 3 places they are,

1> At variable level.
2> At Method level.
3> At Class level.

① Final at variable level :-

If we don't want to change the value of the variable then variable value must be made as final.

In general every ~~final variable~~ common value must be made as final.

syntax:- final datatype $v_1 = val_1, v_2 = val_2, \ldots v_n = val_n$;

Example:- ① final int a = 10; // valid.
a = a+1; // Invalid.

② final float PI = 3.1417f;
PI = PI+2; // Invalid.

Each and every final variable value can be used by every programmer but they are not permitted to change its value i.e. final variable value cannot be change

17/7/13

② final at method level :-

In any programming language to perform any type of operation we use functions/methods

Specially in Java programming functions concept is known as methods.

Syntax for Defining method :-

```
returntype  methodname ( List of formal parameters If any)
{
    Block of Statements . ( having local variables)
}
```

In the above syntax return type represents the type of value returned by the method. If the method is not returning any value then its return type must be void.

Method name represents one of the Java valid variable name. In Java programming If we use any method & if it is containing either one word or more than one word then first word first letter is small & rest of subsequent words first letter must be capital. (This is nothing but Hungerian notation)

e.g: itemStateChanged ()
actionPerformed ()
textValueChanged ()

} Java follows Hungerian notations as shown.

Formal parameters represents list of variables used for holding the values.

Block of stmts represents set of executable statem which will provide solution to the problem & such logic is known as Business Logic.

If we use any variables as a part of method body then it is known as Local Variable.

e.g: write a method for computing $(a+b)^2$.

```
int calculate ( int a, int b)
{
    int c;
    c = a*a + 2*a*b + b*b;
    return (c);
}
```

write a method for calculating area of Circle (π

```
float calCircleArea( float r )
{
    float area = 3.1417f * r * r;
    return (area);
}
```

In s/w development if we develop any method which is common for most of the Java programmer then such common method's defination must be preceded by final.

Hence every final method can be used by every Java programmer but they can not redefined or they can not be override

In other words final methods cannot be overridden.

e.g: Write a method for calculating simple Interest. ($SI = \frac{P*R*T}{100}$)

```
final float simpleInterest (float P, float T, float R)
{
    float si = P*T*R/100;
    return (si);
}
```

(Hence as final is heare noone is able to change logic of this method)

③ final at Class Level :-

If we don't want to give features of base class into derived class then the defination of base class must be made as final.

Once the class is final, whose features can not be inherited into derived classes.

e.g:
```
final class personal
{
    ==
}

class others extends personal
{
    ==.                    // error.
}
```

NOTE:-

1) Final variable values can not be modified.
2) final methods can not be overridden.
3) final classes can not be inherited.

OOPS Principles / Concepts :-

In real Industry to develop any real world project we need to choose a suitable language and it may belongs to either procedure oriented programming lang. or object oriented programming lang.

In other words in s/w development we have 2 types of models of languages they are,
1) procedure oriented prog. lang.
2) Object oriented prog. lang.

① Procedure Oriented programming Languages:-

ALP, C, COBOL, Pascal, VB, Oracle 7.3, Perl (practical expert retrival language) etc.

In Industry procedure oriented prog. languages are always used for developing applications / projects like system s/ws and application s/ws. And Industry is not recommended to use procedure oriented prog. lang for development of Internet based applications like distributed applications.

If we develop any distributed appl^n (website) by using procedure oriented prog. lang. then such distributed

applications will get following limitations.

1) security problems are more.
2) The data is visiting betn client & server side application in the form of plaintext but not in the form of cipher text / encrypted form.
3) The data is visiting betn client & server side applN in the form of byte by byte & it results in poor communication.
4) The data is around functions but not around the objects.

Hence to avoid the above problems of procedure oriented prog. lang. which are developed in connection with distributed applications, Industry always recommends use some programming lang. which will satisfy object orientation principles.

(2) Object oriented Programming language :-

Object oriented principles are classified into 8 types they are

1) Classes
2) Objects.
3) Data Abstraction.
4) Data Encapsulation.
5) Inheritance
6) Polymorphism.
7) Dynamic Binding.
8) Message Passing

In order to say a language is object oriented it has to satisfy object oriented principles.

e.g. SmallTalk, Lisp, Object COBOL, Object Pascal, C++, Java, .Net, Riffil, Simala, etc

All the above OOPS principles are common for all object oriented programming languages but whose implementation or definations are varying from one object oriented prog. lang. to another object oriented prog. lang.

19/07/13

① Class :-

Classes concept is always used for development of user defined / programmer defined data types.

While we are using classes concept for development of user defined datatype we use a keyword called 'class'

Each and every Java program must start with a concept of class. In other words without classes concept there is no Java program

Each and every class name in Java can be treated as user / programmer defined datatype.

Defination:-

The process of Binding Data members and associated methods in a single unit is known as class

:OR:

logical existance :- feels that they are present but actually they doesn't takes memory space.

A Class is a collection of data members and methods.

Whenever we define a class there is no memory space is created for the data members & methods but memory space will be created for the data members & methods when we create an Object with respect to a class. Hence all the classes. Contains logical existance and objects containts physical existance

In Object oriented programming we have 2 types of methods they are,

A] Member Methods.

B] Non-Member methods.

A member method is one which is defined inside the scope of the class and it can access the data of the class.

A Non member method is one which is not available within the scope of the class & it can not access the data of the class.

Java programming allows only member methods but not Non member methods.

Datamembers of a class are also known as properties or fields or attributes. And methods are also known as Behaviours, or Accessories

In real world application development classes can be expressed by using class diagrams & they can be developed by using UML softwares like Rational Rose, Turbo Analyst (IBM products) The structure of class diagram is,

Class name
~~data members~~
~~methods~~

e.g: Draw a class diagram for student.

Student
stno: name: marks: collegename:
int getNumberOfHourStudy() float getTotalMarks() String getGrade()

Syntax for Defining a class in Java:-

```
class classname
{
    variable declaration (~~data Types~~) (datamembers)
    methods defination
}
```

Class is a keyword used for developing user/programmer defined data type.

Classname represents a java valid variable name treated as the name of class. Each & Every classname in java is treated as userdefined datatype. Whenever we define a class there is no memory space is created for the data members and methods of a class but whose memory space is created when we create an object w.r.t. class.

Variable declaration represents data members of a class and they will be selected depends on the name of a class.

Methods defination represents set of methods meant for performing some specific operations & they will be selected depends on the name of the class. Each & Every method in java must be defined inside the class i.e. no method of java to be defined outside the class.

The defination of the class must be incorporated within { and } and the defination of the class may or may not be terminated by (;) semicolon.

NOTE:-

If a class name contains either one word or more than one word then all words first letters must be capital.

e.g. ActionEvent, NumberFormatException, Button, etc

Q. Write a class defination for student.

```
Class Student
{
    int stno;
    String name;
    float m1, m2, m3;

    int getNumberOfHourStudy ()
    {
        return (2);
    }

    float getTotalMarks ()
    {
        float tot = m1+m2+m3;
        return (tot);
    }

    String getGrade ()
    {
        return ("Destinction");
    }
} // student.
```

Q. Write a class defination for computing sum of two numbers.

```
Class sum
{
    int a, b, c;
```

sum
int a;
int b;
int c;
void accept()
void add()
void disp()

```
void accept ()
{
    a = 10;
    b = 20;
}

void add ()
{
    c = a+b;
}

void disp ()
{
    sop (" val of a = " +a);
    sop (" val of b = " +b);
    sop (" sum = " +c);
}
} // sum
```

② * Object :-

We know that when we define a class there is no memory space created for datamembers and methods of class but whose memory space is created when we create an object with respect to class.

In otherwords If we want to enter values of datamembers of the class first we have to allocate memory space for datamembers of class by creating its object.

has class variable
UDT Student s;

To create an object there must exist a class defination. Without object creation data processing is not possible in Java based applications.

Definations of object :-

recommended during interview

① Instance of a class is known as an object (Instance is nothing but allocating sufficient amount of memory space for datamembers & methods of a class)

② Each & Every class variable is known as an Object.

③ Each and Every Grouped item is known as an Object.

not recommended

④ Real world entities are called Objects.

⑤ Blueprint of a class is known as an object.

* Creating An Object In Java:-

Creating an object is nothing but allocating memory space for data members & methods of a class by following dynamic memory allocation by using new operator. & it is known as dynamic memory allocation operator.

functions of new operator:-

① It allocates sufficient space for datamembers and methods of a class.

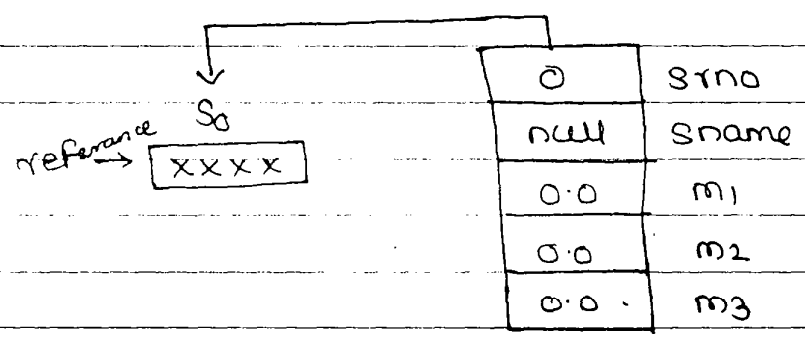
② It takes an address/Referance of a loaded class and place it into L.H.S variable i.e. object name.

To create an object with new operator we have 2 syntaxes.

syntax 1:-

<clsname> objname = new <clsname()>;

e.g: Student So = new Student();



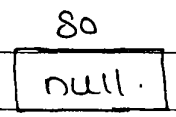
Syntax 2:-

<clsname> objname;
Objname = new <clsname()>;

Here stmt 1 represents object declaration when ever an object is declared, memory space is

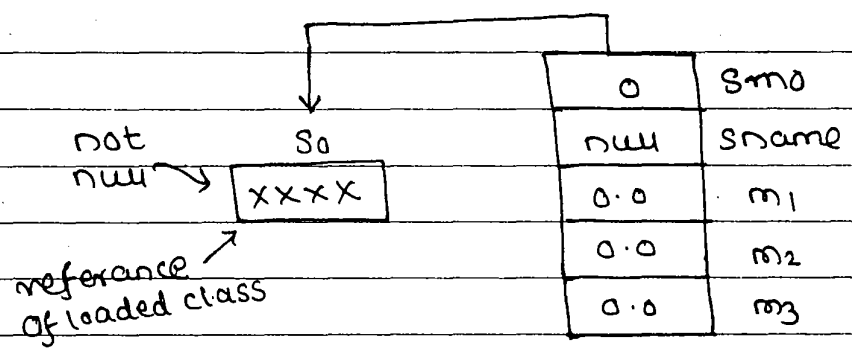
not created for datamembers & methods of a class & whose default value is null.

Student so; // student object declaration.



stmt2 represents object referencing. whenever an object is referenced memory space is allocated for data members of the class & whose default value is not null (which is nothing but referance | Address of loaded class)

so = new Student (); // student object referencing memoryspace



Q. Define a class Loader Subsystem.

It is one of the program developed by SUN microsystem & added as a part of Java SW & whose role is to transfer or load the .class files from secondary memory (Hard disk) into main memory (RAM).

23/07/13

Main memory
① Heap memory - objects.
② Stack memory - peg logic
③ Associative memory - constants values.

S4

Q. What are the differances between classes & objects?

Class	Object
A class is a collection of data members & methods	Instance of class is known as object.
When we define a class no memory space is created for the data members & methods	When we create an object memory space will be created for data members & methods of a class.
All the classes will contain logical existance	All the objects will contain physical existance.
The class definations will exists only once	w.r.t one class defination one can create multiple objects.
When we execute a Java program, the defination of class loaded once in main memory from secondary memory	After loading class def^n in the main memory, later objects will be created w.r.t loaded class.

* NOTE:-

All the objects of Java resides in Heap memory.

All the methods of Java resides & executes in stack memory

All the constant values/figurative constants resides in Associative memory.

* Types of Data members In a class :-

In a class of Java we can write 2 types of data members they are,

1. Instance | Non-Static data members.
2. Static data members.

Instance Data Member	Static Data Member
Instance data members are those whose memory space is created each & every time whenever an Object is created.	Static data members are those whose memory space is created only once when the class is loaded in main memory irrespecti of number of objects are created.
Instance data members are always used for storing specific values.	Static data members are always used for storing common values.
Instance data members declaration should not be preceded by static keyword	Static data members declaration must be preceded by static keyword.
syntax: datatype $v_1, v_2, \ldots v_n$;	syntax: static datatype $v_1, \ldots v_n$;
Each & Every Instance data member must be accessed wrt object name as, Objname. Instancedatamemname	Each & every static data member must be accessed wrt Class name as, classname. staticdatamemname

Instance data member values are not sharable	Static data member values are sharable.
Instance data members are also known as object level data member because they depends upon ~~classnc~~ object name & independent from class name	static data members are also known as class level data member because they depends upon class name & independent of object name.
e.g: int stno = 10; String s1 = "swap";	e.g: static float PI = 3.1417f; static String bname = "SBI";

Each and every final variable value must be static.

e.g:

```
static final float PI = 3.1417f;
```

But A static variable value may or may not be final.

e.g: In one point of time course name for group of student is JSE & in another point of time for some other group of student course name is Adv Java.

```
static String crsname = "JSE";
        :
        :
crsname = "ADV Java";
```

* Types of Methods In Java :-

In Java programming we have 2 types of methods They are,

① Instance / Non static Methods.

② Static Methods.

Instance Methods	Static Methods.
Instance methods are those which are recommended to perform repeated operations such as reading records from file, database, ~~user authentication process~~, etc.	Static methods are those which are recommended to perform one time operations like opening files in read or write mode, obtaining database connection, etc.
Instance methods meant for performing specific operations.	Static methods are meant for performing common operations.
Programmatically Instance methods defination should not be preceded by static	Static methods defn must be preceded by static.

Syntax:-

```
ret.type method name( list of formal parameters if any )
{
    block of stmts;
}
```

```
static rettype methname( list of formal par if any )
{
    block of stmts;
}
```

Each & every instance method must be accessed w.r.t object name objname. InstanceMetName();	Each & every static method must be accessed wrt Class name. Classname.staticmetname();
Instance methods results are not sharable	Static methods results are sharable.
Instance methods are also known as object level methods coz they depends on object name & Independent from class name.	static methods are also known as class level methods coz they depends on class name & Independent from object name.

e.g. int calTotal()
{
int tot = m1+m2+m3;
return (tot);
}
	e.g: public static void main (String_args)
	{
	S1.calTotal();
	S2.calTotal();
	}

We know that a class of Java contains Instance datamembers, static datamembers, Instance methods & static methods, Instance datamembers & Instance methods must be accessed wrt object name whereas static datamembers & static methods must be accessed wrt classname.

* System.out.println (-)

Above stmt is used for Displaying result of the Java program on the console (monitor) Line by line.

System.out.print (-)
Above stmt is used for displaying result of Java program on the console in the same line.

Technical Description:-

println and print are the two predefined ~~method~~ instance methods present in a predefined class called printstream class

To access println and print methods we require an object of printstream class.

An object of printstream class called 'out' created as a static datamember present in another predefined class called System hence println & print methods can be accessed as System.out.println(-
System.out.print(-)

Ex. ①: System.out.println(" Hello Java world");

②: int a=10;
Sop (a); OR
Sop (" val of a" + a);

③ int a=10; b=20;
int c = a+b;
Sop ("sum of" + a + "and" + b + " = " + c);

* Structure of the Java program :-

Structure of a program is nothing but the standard format released by lang. developers to the Industry programmers. SUN Developers has released the following standard format to the Java programmers for developing Java applications.

```
Package details;
Class <Clsname>
{
    Data members

    user defined methods

    public static void main (String S[])
    {
        Block of Stmts;
    }
}
```

Explanation:-

In the above structure,

① We know that a package is a collection of classes, interfaces and subpackages. A subpackage Inturns contains collection of classes Interfaces & subsub pcks If we use any predefined class or interface as a part of our Java program, It is a responsibility of a Java programmer to specify in which package these classes & interfaces presents. out of so many number of packages in Java one of the package called java.lang.* will be imported by default to every Java prog This package is known as default

② Each & every Java program must start with a concept of class. A class is keyword used for developing user/programmer defined datatype.

③ <clsname> represents Java valid variable name treated as name of class. Each & every classnam in Java programmatically treated as user defined data type.

④ Datamembers represents either instance or static and they will be selected depends on the name of the class.

⑤ User defined methods represents either instance or static meant for performing some operations repeatedly or one time respectively.

⑥ Each & Every Java program starts executing from main method hence main method is known as program Driver.

⑦ Since main method of Java is not returning any value hence whose return type must be void

⑧ Since main method of Java executes only once throughout the life of entire Java program & hence whose nature must be static.

⑨ Since main method can be accessed by every Java programmer & whose access specifier must be public. (Access specifier makes us to understand where to access the data & where not to access data)

(10) Each and every main method of Java must take array of objects of String class.

(11) Block of statements represents set of executable stmts which inturns calls user defined methods either wrt object or wrt class depends on the type of method.

(12) The file naming convension in Java programming is that whichever class name is containing main method, that classname must be given as a filename with an extension .java.

Q. Write a Java program which will display a message called "Hello Java World".

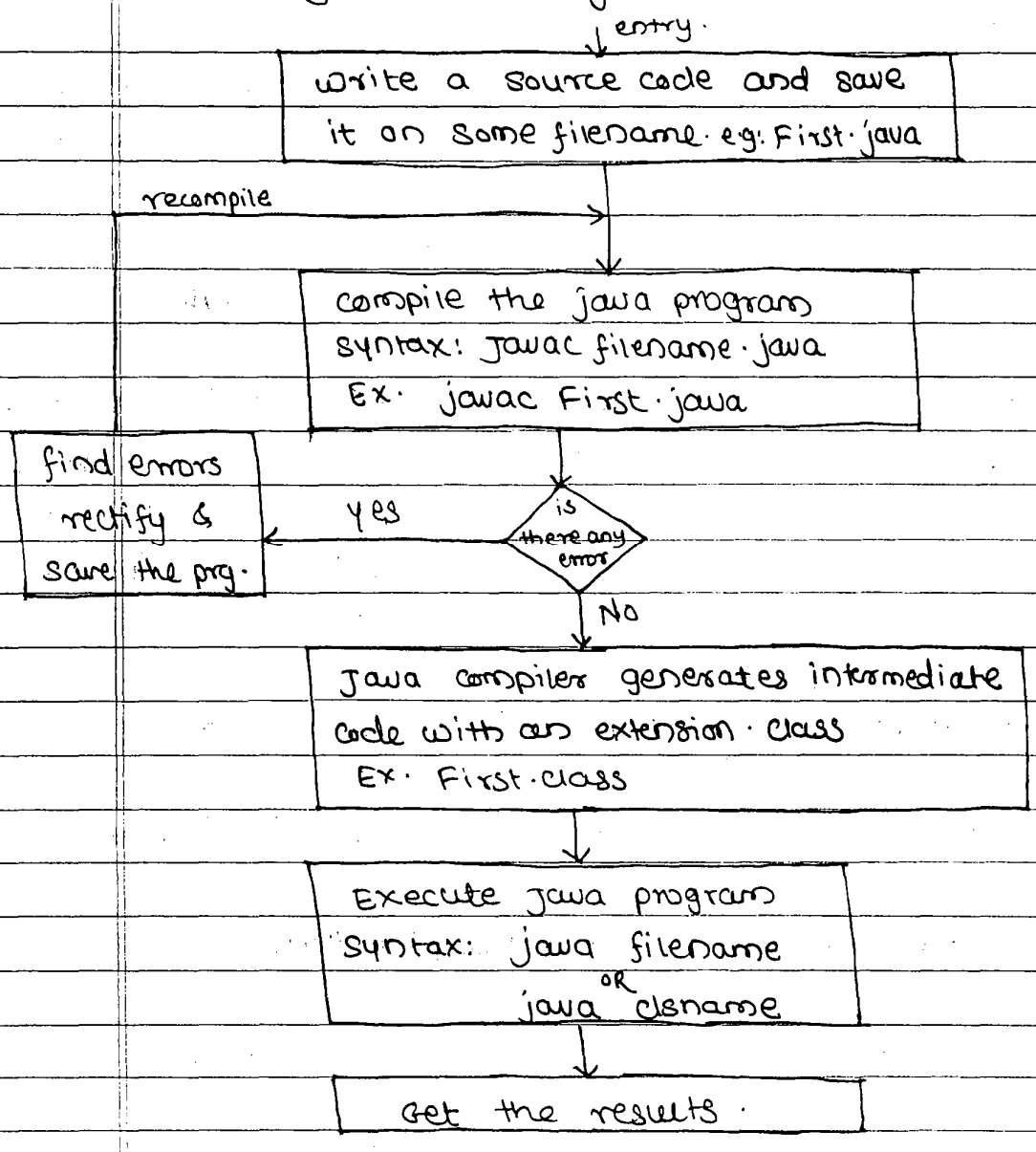
```
// First.java.

Class First
{
  public static void main (String s[])
  {
     System.out.println ("Hello Java World");
  }
}
```

```
cmd> javac First.java.
cmd> java First
```

* Steps for compiling and Executing Java program :-

The following diagram gives sequence of steps for compiling & executing java program.



In the above diagram javac and java are the two exe files or tools or application programs developed by SUN microsys. & supplied as a part of JavaHome/bin folder and they are used for compiling & executing java prgs respectively

* Requirement analysis for Developing Java Application or projects :-

To develop any Java appl^n we require 3 requirements.

1) Java Software requirement: Download jdk1.5, jdk1.6, jdk 1.7 from www.oracle.com or www.sun.com freely

2) OS requirement: Any OS can be taken because Java is one of the platform independent.

3) IDE requirement: (Integrated Development Enviornment) IDE is one of the third party s/w developed by third party vendors and releases to the real Industry for the development of Java applications by getting inbuilt Compatabilities.

various IDES are,

I] Notepad, Dos Editor, wordpad, Notepad++,

II] EditPlus (developed by ES Enterprise)

III] NetBeans, Eclipse, MyEclipse, IntelliJIDEA, etc.

Q. How Java program Executes? Explain.

ANS: Before executing the java program compile the java program.

javac filename.java

Ensure filename.class file must be generated.

Execute the Java program.

Java filename

When the above stmt executes following 4 operations will be taken place internally.

1) Class loader subsystem loads filename.class in the main memory from secondary memory.

2) JVM takes the loaded class (filename.class)

3) JVM looks for main method to start its execution

4) JVM calls the main method cort loaded class as filename.main()

Q. Write a Java program which listed the concept of Instance methods and static methods.

Ans:

```
// Third.java
class Third
{
    void f1();
    {
        f(3) f3();    // Rule ②
        sop("f1() Instance");
    }
    static void f2()
    {
        sop("f2() static");
    }
    void f3()
    {
        Third.f2();    // Rule ④
        sop("f3() Instance");
    }
```

```
psvm( String S[])
{
    sop(" starting of main()");
    Third t1 = new Third();
        t1.f1();                    // Rule (3)
        f2();                       // Rule (1)
    sop(" Ending of main()");
  } // main
} // Third.
```

Rules :-

(1) One static method can call another static method directly provided both static methods belongs to same class.

(2) One instance method can call another instance method directly provided both instance methods belongs to same class.

(3) One static method can call another instance method wrt objectname without considering wheather they belongs to same class or different class.

(4) One instance method can call another static method wrt classname & without considering wheather they belongs to same class or different class.

These rules will establish the commn betn the methods.

Q. Write a Java program which will compute sum of two numbers?

ANS:-

```
// SumDemo.java.
Class Sum
{
   int a,b,c;
   void accept()
   {
      a=10;
      b=20;
   }
   void add()
   {
      c=a+b;
   }
   void disp()
   {
      SOP("a=" +a);
      SOP("b="+b);
      SOP("sum=" +c);
   }
} // Sum.
Class SumDemo
{
   PSVM(String S[])
   { SOP("starting of main");
      Sum So = new Sum();
         So.accept();
         So.add();
         So.disp();
      SOP("ending of main");
   }
} //SumDemo
```

If we run the above program multiple number of times then we get the same result. If we want to get different results then we need to perform the following set of operations.

1) goto the program
2) change the values in the required place.
3) Save the program.
4) compile the program.

If any program satisfies the above properties then such type of programming is called static programming/ hard codded approach. Industry is not recommended to develop hard codded approach based applications & Industry is always recommending to develop dynamic programs / Lexical applications.

Q. Define Hard coded approach and flexible approach.

ANS: In hard coded approach the input of the program is participating during compilation time

In flexible approach the input to the program is participated during execution of the program.

* Accepting the values Dynamically to the Java program :-

In Java programming we have 4 approaches to accept the values dynamically They are,

1. Through command prompt.
2. Through Keyboard.
3. Through properties / Resource Bundle file.
4. Through XML document.

Length:-

Length is one of the implicit variable created by JVM & supplied to each & every Java program for two purposes they are,

1. To find the number of elements in an array (size of array).
2. To treat fundamental arrays (int array, float array Char array, etc) as objects.

Syntax:- arrName.length.

Here arrName represents name of the array.

e.g: 1] int a[] = {10, 20, 30}

```
sop(" Number Of values:" + a.length);
for( int i=0; i< a.length; i++)
{
    sop(a[i]);
}
```

2] String s[] = {"c", "c++", "Java"}

```
sop(" Number Of courses:" + s.length)
for ( int i=0; i< s.length; i++)
{
    sop(s[i]);
}
```

① Accepting the values Dynamically through Command prompt :-

The number of values passing to the Java program from the command prompt are known as command line arguments.

Accepting command line arguments are nothing but accepting the data dynamically to the Java prg.

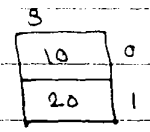
Ex: $\underbrace{\text{E:\textbackslash Swap\textbackslash prgs>}}_{\text{command prompt}}$ Java $\underbrace{\text{SumDemo}}_{\text{Java prg}}$ $\underbrace{\text{10 20}}_{\text{values}}$

Here 10 & 20 are called command line arguments. When the above stmt is in execution following 4 actions will takes place internally.

1) class loader subsystem loads SumDemo class alongwith command line arguments (10,20) into the main memory.

2) JVM takes the loaded class (SumDemo) alongwith the loaded command line arguments (10,20) & also counted its length & placed in length variable. (2)

3) JVM looks for main () method & places command line arguments in a variable of string type. i.e. main (String s[2])



This indicates every command line argument is passing to main method & available in the form of array of objects of String class.

4) JVM calls the main method wrt loaded class as SumDemo.main(-).

e.g: E:\swap\prgs> java Sample 10 10.75 A true

Sample.main(String K[4])

K	
10	0
10.75	1
A	2
true	3

Hence what are all the command line arguments we are passing to the java prg. They are sending to the main method & available in the form of array of objects of string class.

NOTE:-

SUN microsystem developers has withdrawn the decision of providing 8 main methods ~~with~~ by taking 8 fundamental arrays (int array, float array char array, etc) because of following reasons,

a) Writing 8 main methods is complex in writing & lengthy process.

b) Internally they don't want to treat fundamental values as fundamental arrays

In order to provide more simplicity & to treat every command line argument as object, they decided to take single main method by taking array of objects of string class.

Q. Write a Java program which will accept command line arguments & display those values.

```
// DataRead.java
class DataRead
{
    psvm( String S[] )
    {
        SOP(" No of cmd line args:" + S.length);
        for (int i=0; i< S.length; i++)
        {
            SOP( S[i]);
        }
    } // DataRead.
```

cmd> javac DataRead.java.
cmd> java DataRead 10 20 1.5f A swap.

compile the program ensure DataRead.class file is generated. Run Java prg by passing command line arguments.

27/07/13

* Converting String Type data into fundamental type data :-

We know that what are all the command line arguments we are passing to the Java program, They are by default available in the main method in the form of array of Objects of String class. On String data it is not possible to perform any numerical operations. To perform numerical operations on the String data we need to convert String type data into fundamental / numerical type data.

The following table gives conversion of numerical string type data into numerical / fundamental type values.

fundamental data type	wrapper class name	conversion method from numerical string type data into numericals (fundamental datatype)
byte	Byte	public static byte parseByte (String)
short	Short	public static short parseShort (String)
int	Integer	public static int parseInt (String)
long	Long	public static long parseLong (String)
float	Float	public static float parseFloat (String)
double	Double	public static double parseDouble (Str
char	Character	~~ptb~~
boolean	Boolean	public static boolean parseBoolean (Str

In general each & every wrapper class contains the following predefined static method.

wrapper class nam

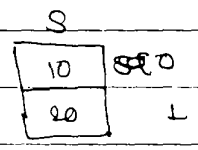
public static xxx parseXxx (String)

Here xxx represents any fundamental data type except char datatype.

When we are using any types of methods in our Java programming then we have to consider 3 important points They are,

1. Decide what type of method we are using? (Instance | Static).
2. What type of parameter it is taking & pass its value?
3. What type of return type it is returning & hold that value in that type of variable.

e.g: 1)
```
String S[] = {"10", "20"};
sop( S[0] + S[1])        // 1020
int x = Integer.parseInt(S[0]);
int y = Integer.parseInt(S[1]);
int z = x+y;
sop(" sum=" +z);         // 30.
sop(" sum=" +x+y);       // 1020
sop ("sum=" + (x+y));    // 30.
```



2)
```
String s= "10.75";
double d1= Double.parseDouble(s);
```

3)
```
String s = "true";
boolean b1= Boolean.parseBoolean(s);
```

Q. Define a wrapper class? What is its purpose?

ANS: For each & every predefined data type there exists a predefined class. such predefined class is known as wrapper class.

The purpose of wrapper classes is to convert numerical string type data into numerical type data.

Q. Write a Java program for swapping of two values:

ANS:

```
// swapDemo.~~class~~ java.
Class Swap
{
    int a,b;
    void accept (int n1, int n2)
    {
        a=n1;
        b=n2;
    }
    void disp
    { sop (" val of a = " + a);
      sop (" val of b = " + b);
    }
    void swapValues ()
    {
        int t = a;                a = a+b;
            a = b;        or      b = a-b;
            b = t;                a = a-b;
    }
}   // swap => Business logic class.

Class SwapDemo
{
    psvm ( String S[])
    {   if ( K.length != 2)
        {  sop (" please enter two values");
        }
        else
        {
            int x = Integer.parseInt (S[0]);
            int y = Integer.parseInt (S[1]);
```

```
        swap so = new swap();
            so.accept (x,y);
        Sop(" originial values:");
            so.disp();
            so.swapvalues();
            sop(" swapped values:");
            so.disp();
    } //else
  } // main
} // SwapDemo => Execution logic.
```

Q. Write a java program which will accept a number from the command prompt & Display its multiplication table.

ANS:

```
// MulDemo.java.
class mul
{   int n;
    void set(int x)
    {
        n=x;
    }
    void table ()
    {   for(int i=0; i<=10; i++)
        {
            sop("n+" * "+i+ "=" + (n*i));
        } //for
    }
} // mul  BLC

class MulDemo
{   psvm (String S[])
    {   if (S.length != 1)
            sop (" enter one value");
```

every class name in Java is a programmer defined datatype.

```
        else
        {
            int n = integer.ParseInt(S[0]);
            if (n<=0)
            Sop("n+" Invalid input try again");
            else
            {   mul mo = new Mul();
                    mo.set(n);
                    mo.Table();
            }
        }
    }  // main
}  // MulDemo ⇒ ELC
```

29/7/13 mon.

Q Write a Java program which will decide wheather the given number is prime or not.

ANS:

```
// PrimeDemo.java.

class Prime
{
    int n;
    void set(int x)
    {
        n=x;
    }

    void disp()
    {
        String res= decide();
        Sop(res);
    }
```

```
String decide ( )
{
    int i;
    for( i=2; i<n; i++)
    {
        int r= n%i;
            if (r==0)
            {
                break;
            } //if
    } //for
    if (i==n)
    return " PRIME";
    else
    return " NOT PRIME";
  } // decide
} // prime      BLC

class PrimeDemo
{
  psvm( String args[])
  {
      if (args.length!=1)
       sop(" Enter only one value);
      else
      {
          int x= Integer.parseInt (args[0]);
            if (x<2)
             sop (x+" is Invalid input try again");
            else
          { Prime P0 = new Prime();
                P0.set (x);
                P0.disp ();
          } //else
```

goto stmt is not there in Java.

NOTE:- A method of Java is not only returning fundamental data types as a return type but also they can return Classnames as a return type. (Every classname of Java can be treated as one type of datatype)

Q. Write a Java program which will convert ordinary numbers into Roman Numbers.

ANS: Logic Analysis:-

10 → X (while)	100 → C (while)	1000 → M (while)
9 → IX	90 → XC	900 → CM
5 → V (if)	50 → L (if)	500 → D (if)
4 → IV	40 → XL	400 → CD
1 → I (while)		

```
// RomanDemo.java.
Class Roman
{
    int n;
    void set (int x)
    {
        n = x;
    }
    void convert()
    {
        if (n<=0)
         sop ("No Roman equivalent");
        else
        {
            while (n>=1000)
            {   sop ("M");
                n = n-1000;
            }
```

```
if (n >= 900)
{ sop ("CM");
  n=n-900;
}
if( n >= 500)
{ sop ("D");
  n=n-500;
}
if (n>= 400)
{ sop ("CD");
  n=n-400;
}
while( n >= 100)
{ sop ("C");
  n=n-100;
}
if( n>= 90)
{ sop(" XC");
  n=n-90;
}
if (n>= 50)
{ sop(" L");
  n=n-50;
}
if( n>= 40)
{ sop( "XL");
  n=n-40;
}
while( n >= 10)
{ sop ("X");
  n=n-10;
}
```

```
            if(n>=9)
            { sop ("IX");
                n=n-9;
            }
            if(n>=5)
            { sop ("V");
                n=n-5;
            }
            if (n>=4)
            { sop("IV");
                n=n-4;
            }
            while (n>=1)
            { sop ("I");
                n=n-1;
            }
        }//else
    } // convert.
} // Roman. => BLC.

Class RomanDemo
{   psvm ( String args[])
    { if ( args.length != 1)
          sop(" Enter only one value");
      else
      { int x = Integer.parseInt(args[0]);
          Roman r = new Roman();
                r.set (x);
                r.convert();
      } //else
    } // main
}   // RomanDemo.
```

Q. Write a Java program for converting Roman number to ordinary number.

ANS:

When we pass MMX from command prompt that will be available in array of objects of String class. { args }

MMX	0

Here split individual characters of args[0] & compare with Roman number & add those equivalent values. (1000 + 1000 + 10)

If we enter 'MCA' then it is Invalid because 'A' is not a Roman number.

30/07/13

* Constructors In Java :-

Defination:-

A constructor is a special member method which will be automatically called by JVM (implicitely) whenever an object is created for placing our own values without placing default values.

The aim of constructors concept is to initialize an object.

Advantages of constructors:-

If we write any Java prg. with the concept of constructors then such appln will get the following advantages,

1. Constructors eliminates in placing default values.
2. Constructors eliminates in calling ordinary methods.

Rules | properties | Characteristics of constructors:-

When we are using constructors in our Java program we follow the following rules.

1. constructors will be called implicitly by JVM whenever an object is created.

2. constructors name must be similar to class name

3. constructors should not return any value even void also. (If we write any written type then it is by default treated as ordinary method).

④ Constructors are not static (because constructors will be called each & every time whenever an object is created)

⑤ Constructors will not be inherited from one class to another class (coz every constructor is meant for initializing its own data members but not meant for initializing other class data members

⑥ The Access specifier of the constructor may or may not be private

i) If access specifier of constructor is private then we can create object of corresponding class in the same class context but not in other class context.

ii) If access specifier of constructor is not private then we can create an object of corresponding class in both same class context & other class context.

Types of constructors :-

Based on the object creation, constructors are classified into two types they are,

31/07/13

① Default / Parameterless / 0-argument constructors :

A constructor is said to be default if & only if it never takes any parameters.

The purpose of default constructor is to create multiple objects for placing same values.

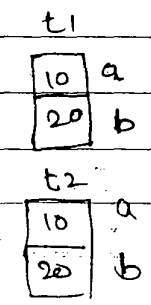
syntax:-

```
class <clsname>
{ -----
  -----
  < clsname () >   //Default constructor
  {
     Block of stmts; //Initialization
  }
  -----
}
```

Example: Write a Java program which listed the concept of default constructor.

```
class Test
{
   int a,b;
   Test ()
   {
      sop(" Test- Default constructor");
      a = 10;
      b = 20;
      sop("val of a = " +a);
      sop ("val of b = " +b);
   }
} //Test

class TestDemo
{
   psvm ( String s[])
   { Test t1 = new Test();
     Test t2 = new Test();
   }
}
```



Rules:-

① Defining the default constructor is optional. If we create an object of a specified class wrt default constructor then a programmer may or may not define default constructor.

If we are not defining the default constructor then JVM will call its own constructor known as system defined default constructor & places default values depends on the data members of the class.

If we define our own default constructor then JVM will execute programmer defined default constructor & it places our own values instead of placing default values.

② Parameterized Constructor:-

A constructor is said to be a parameterized if & only if it always takes parameters.

The purpose of parameterized constructor is to create multiple objects wrt same class for placing different values.

Syntax:-

```
class <clsname>
{ -----
    <clsname (list of formal parameters)>
    {
        Block of stmts;  //Initialization.
    }
    .....
}
```

parameterized constructor

Example: Write a Java program which listed the concept of parameterized constructor.

```
class Test
{
    int a,b;
    Test (int x, int y)
    {
        sop (" Test = Double parameterized const.");
        a=x;
        b=y;
        sop (" val of a=" + a);
        sop (" val of b=" +b);
    }
}

class TestDemo
{
    psvm (String s[])
    {
        Test t1= new Test (1,2);
        Test t2= new Test (10,20);
    }
}
```

Rules:-

① If we create an object with parameterized constructor then it is mandatory to the Java programmer to define parameterized constructor (otherwise we get compile time error).

(2) Whenever we create multiple objects with both default & parameterized constructors then it is mandatory to the Java programmer to define both default & parameterized constructors. otherwise we get compile time error.

NOTE:-
If we are not initializing any variables ~~of the class~~ or data members of the class in the current class constructors then they are automatically initialized by system defined default constructor with default values.

* Overloaded Constructor:-

A constructor is said to be overloaded if & only if constructor name is same but its signature is different

Signature represents the following points:-

a) Number of parameters
b) Type of parameters
c) order of parameters

} At least one thing must be differentiated.

e.g.:
```
Test t1 = new Test (10, 20);
Test t2 = new Test (100);
Test t3 = new Test (1.5f, 2.5f);
Test t4 = new Test (1.5f, 20);
Test t5 = new Test (20, 1.5f);
```

* Object parameterized constructor:-

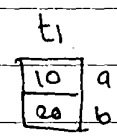
Defination:- A constructor is said to be object parameterized iff it always takes object as a parameter.

Purpose of object parameterized constructor is that to copy the contents of one object into another object where both objects belongs to same type

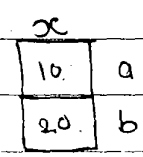
The functionality of object parameterized constructor is exactly ressembling the functionality of copy constructor of C++ (copy constructors are not present in Java)

Example:-

```
Test t1 = new Test (10,20);
```




```
Test t2 = new Test ( t1);
```




```
Test (Test x)
{
    a = x.a;
    b = x.b;
    sop("a="+a);
    sop(" b="+b);
}
```

Q. Write a Java program which illustrate Default Constructor, parameterized Constructor, Object parameterized constructor.

ANS:

```
// TestDemo.java.
Class Test
{
    int a, b;
    Test ()
    {
        Sop (" Test Default constructor");
        a=1;
        b=2;
        Sop (" val of a = " +a);
        Sop (" val of b = " +b);
    }

    Test (int x)
    {
        Sop (" Test: single parameterized constructor);
        a=x;
        b=x;
        Sop( " val of a = " +a);
        Sop( " val of b = " +b);
    }

    Test (int x, int y)
    {
        Sop (" Double parameterized constructor");
        a=x;
        b=y;
        Sop (" val of a=" +a);
        Sop (" val of b = " +b);
    }
```

```
        Test (Test x)
        {
            sop("Test: object parameterized constructor");
            a = x.a;
            b = x.b;
            sop("val of a = " + a);
            sop("val of b = " + b);
        }
    } // Test.

    class TestDemo
    {
        psvm(String s[])
        {
            Test t1 = new Test();
            Test t2 = new Test(1000);
            Test t3 = new Test(10, 20);
            Test t4 = new Test(t2);
        }
    } // TestDemo.
```

1/8/13

Observations:-

1) Each and every class of java can contain two types of constructors they are
   a) single Default constructor.
   b) Multiple parameterized constructor (known as overloaded constructors)

2) Bydefault each & every class of java contains only one constructor known as system defined default constructor.

Programmatically we can write / define programmer defined default constructor & programmer defined parameterized constructor.

3) Data members, constructors & methods of a class is known as profile. To view or to verify the profile of a class / Interface we use "Jawap" It is one of the tool / appln program or exe file present in Java-Home/bin directory.

This can be applied after compiling the Java program

syntax :- jawap <name of class file>.
e.g :- D:\Examples> javap Test.

4) private features of the class never participate in profiling process

NOTE :-
Some of the classes of Java does not contain constructor (True / False) ⇒ False
Every class of Java contains constructors but developer / programmer of class made those constructors as private.

5)

* this :-

'this' is a keyword | Implicit object created by JVM supplied to each and every Java program for Two purposes. They are,

① It always points to current class object.

② Whenever formal parameters & data members of the class are same then JVM gets an ambiguity (no clarity is provided betn data members & formal parameters) To differentiate between formal parameters & data members of the class, the data members of class must be preceded by 'this' keyword.

Syntax:- this.current class data member name.

Q. Write a Java program which illustrate the concept of 'this' keyword.

```
class Test
{
    int a, b;
    Test (int a, int b)
    {
        this.a = a;
        this.b = b;
        this.a = this.a+1;
        this.b = this.b+1;
        a = a+2;
        b = b+2;
        SOP (" value of a (formal param) = " + a);
```

Here Assigning formal parameter values to data members of class.

Here incrementing values of data members of class

~~Here refering~~ Here a & b refers formal parameters

Here we are incrementing values of formal parameters

```
        void disp()
        {
            sop("val of a = " + this.a);
            sop("val of b = " + this.b);
        }
}
```

Here writing 'this' is optional

```
class TestDemo
{
    psvm(String args[])
    {
        Test t1 = new Test(10, 20);
            t1.disp();
    }
}
```

Q. Define Time Complexity

ANS: The amount of time required by program from CPU for its complete execution is known as time complexity.

In order to determine time complexity of program we use following predefined method.

System → public static long currentTimeMillis()

```
long t1 = System.currentTimeMillis();
fact fo = new fact();
    fo.fet(4); fo.factcal(); fo.disp();
long t2 = System.currentTimeMillis();
sop("Execution time = " + (t2-t1));
```

2/8/13

③ Inheritance :-

It is one of the distinguished feature of OOPS.

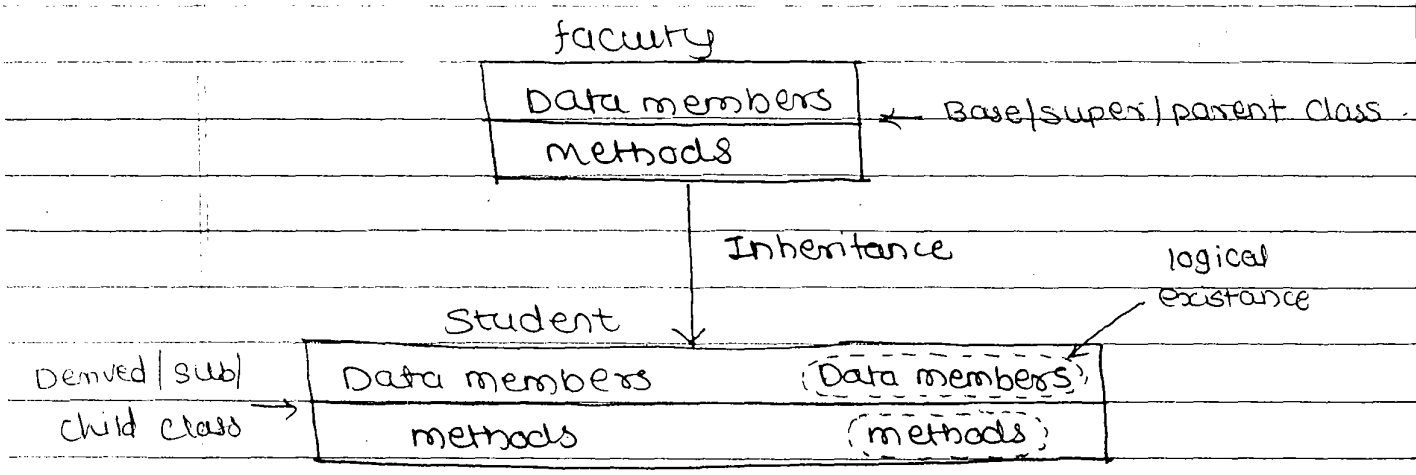
Defination:- The process of obtaining data members and methods from one class to another class is known as Inheritance.

The class which is giving datamembers & methods is known as base/super/parent class.

The class which is taking data members & methods is known as derived/sub/child class.

The data members & methods of a class are known as features.

The following diagram gives view about Inheritance.



The concept of Inheritance is also known as Reusability | Extendable classes | Sub classing | Derivation.

Advantages | Benefits of Inheritance:-

If we develop any Java appl^n with the concept of Inheritance then to that appl^n Inheritance will provide following advantages,

1. Appl^n development time is less.
2. Appl^n memory space (main memory) is less.
3. Appl^n Execution time is less.
4. Appl^n performance is enhanced (Improved).
5. Redundancy (Repeatation) of the code is reduced so that we get consistent results & Storage cost is reduced.
6. We are able to achieve slogan of Java that is WORA (write once Reuse Anywhere)

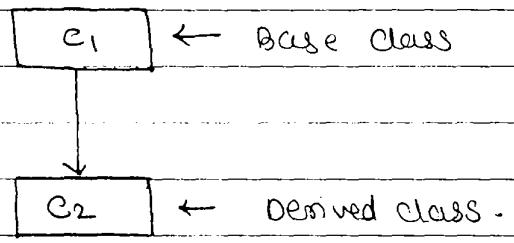
Inheritance Types | Reusable Techniques:-

Based on the number of The features are inherited from one class to another class, inheritanc types are classified into 5 types they are,

① Single Inheritance:-

Defination:- In single Inheritance there exists single Base class & single derived class.

Diagram:-

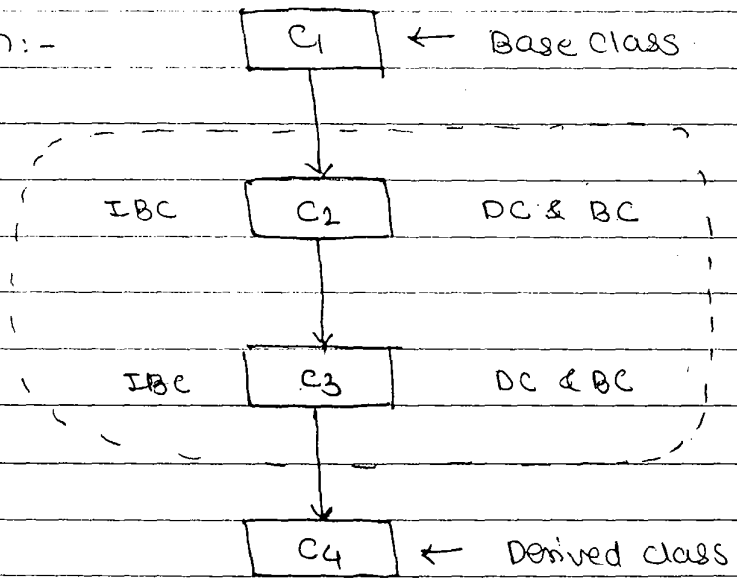


Inheritance path:- $C_1 \rightarrow C_2$

② Multilevel Inheritance:-

Defn:- In multilevel inheritance there exists single Base class, single derived class & multiple Intermediate Base classes (IBC)

Diagram:-

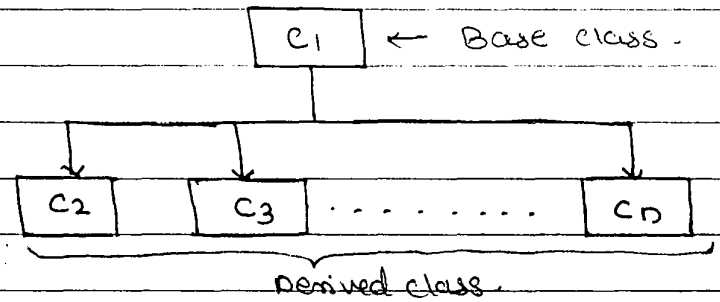


$C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$

③ Hierarchical Inheritance:-

Def^n:- In hierarchical Inheritance there exists single base class & multiple derived classes.

Diagram:-

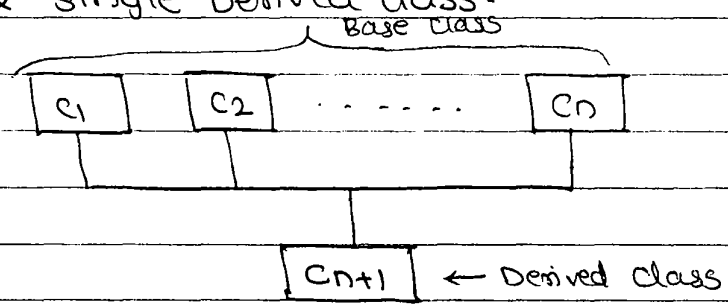


Inheritance path : $C_1 \rightarrow C_2$, $C_1 \rightarrow C_3$, ... $C_1 \rightarrow C_n$.

④ Multiple Inheritance:-

Def^n:- In multiple Inheritance there exists multiple Base Classes & single Derived class.

Diagram:-



Inheritance path:- $C_1 \rightarrow C_{n+1}$, $C_2 \rightarrow C_{n+1}$, ..., $C_n \rightarrow C_{n+1}$.

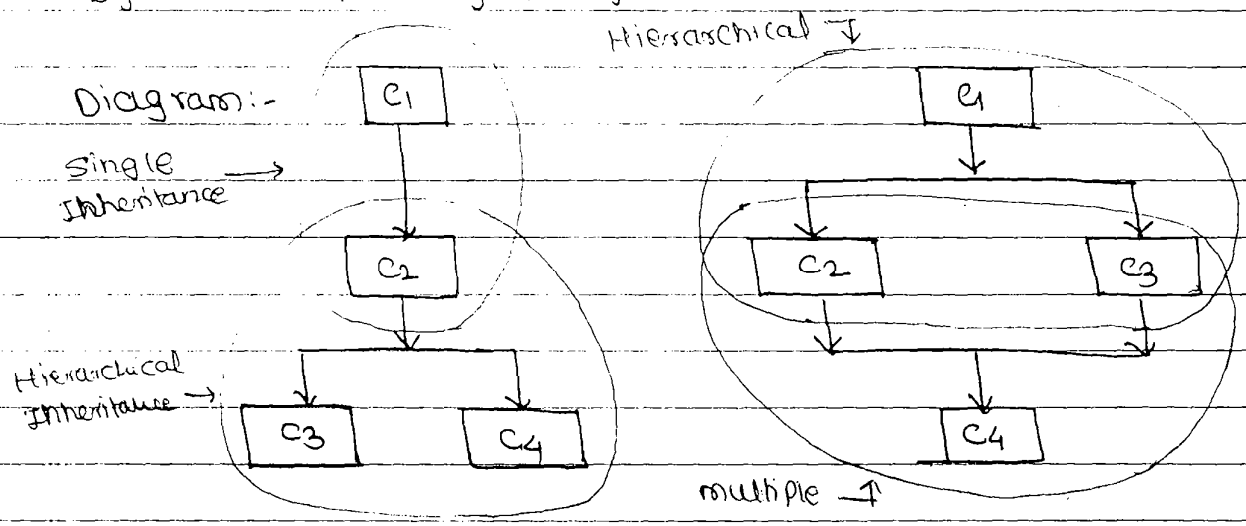
NOTE:- Java programming doesnot supports multiple Inheritance through the concept of classes but it can be supported by the concept of interfaces.

⑤ Hybrid Inheritance:-

Def^n:- Hybrid Inheritance = Combination of any available Inheritance type

In the combination if one of the combination is multiple inheritance then the entire combination is not supported by classes but it can be supported by the concept of interfaces.

Diagram:-



Valid by both classes & Interfaces.

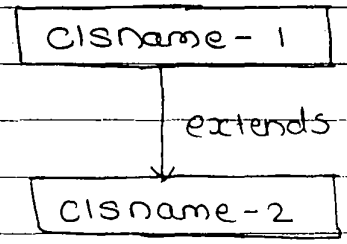
Invalid by classes & Valid by Interfaces.

8/8/13

✱ Inheriting features of Base class into Derived class:-

Syntax:- We use the following syntax for inheriting features of Base class into Derived class.

```
class <clsname-2> extends <clsname-1>
{
    variable declaration;
    method declaration;
}
```



Explanation:-

1) <clsname-1> & <clsname-2> represents name of the base class & derived class.

2) extends is a keyword used for inheriting features of base class into derived class plus it is improving functionalities of derived class

3) In Java programming one derived class can extends only one base class because Java programming never supports multiple inheritance throug the concept of classes but it can be supported through concept of interfaces.

4) Whenever we develop any Inheritance appl^n, it is always recommended to create an object of bottommost derived class because it is inheritin features of topmost base class & Intermediate (base class

5) When we create an object of Bottommost derivec class, first we get memory space for data membe of topmost base class, second we get memory space for datamembers of Intermediate base class & at last we get memory space for data members of bottommost derived class-

6) If we don't want to give features of base class into derived class then defn of base class must be made as final. Hence final base classes never participates in inheritance.

7) If we don't want to give some of the features of base class into derived class then such features must be made as private hence private features of base class never participates in Inheritance or never visible/accessible in the context of derived class.

8) In inheritance process, data members & methods of base class can be inheritted but constructors of base class can not be inheritted into the context of derived class because every constructor is meant for initializing the data members of its own class.

9) By using an object of base class we can access all the features which are present in the same base class but an object of base class cannot access those features which are specially defined in its subclass (This concept is known as <u>scope</u> of base class object)

10) For each & every class in java there exists an implicit pre-defined superclass called java.lang.object because it provides garbage collection facility for collecting unused memory space for improving performance of java applications.

Q. Write a Java program which illustrate the concept of Inheritance.

```
//Indemo.java
Class BaseClass
{
    int a;
    void disp()
    {
        Sop("disp: Base class");
    }
} // Base class.

Class DerivedClass extends BaseClass
{
    int b;
    void show()
    {
        Sop("Show: Derived class");
        a=100;
        b=200;
        Sop("value of a from BaseClass = "+a);
        Sop("value of b from DerivedClass = "+b);
    }
} // Derived class-

Class Indemo
{ psvm(String args[])
  { Sop("wrt Derived class");
      DC do1 = new DC();
      do1.disp(); do1.show();
      Sop("wrt Base class");
      BC bo1 = new BC();
} }   bo1.disp()
```

(Mon) 5/8/13

* Types of Relationships in Java:-

Types of relationships makes us to understand how to reuse the features of one class into another class.

In Java programming we have 3 types of relationships they are,

1. is-A relationship.
2. has-A relationship.
3. uses-A relationship.

1. is-A relationship:-

In is-A relationship one class is getting features of another class by using concept of Inheritance with the help of extends keyword.

e.g:

```
class C1 ~~extends C2~~
{
  ......
  ......
}
                    is-A relationship.
class C2 extends C1
{
  ......
  ......
}
```

Here the relationship betn C1 & C2 is 'is-A'

In is-A relationship features of C1 class & whose memory space available logically in its derived classes

② has-A || part-of | kind-of relationship:-

In has-A relationship, an object of one class can be created as a data member in another class

e.g:
```
class c1
{
  ......
  ......
}
class c2
{
    c1 o1 = new c1();
    ......
    ......
}
```
has-A relationship

In has-A relationship, we created an object of c1 class in the context of c2. In the context of c2, physical memoryspace is created for the features of c1 class. In general if multiple classes creates an object of c1 class then in all the classes context explicitly memoryspace is created for data members of c1 class.

③ Uses-A relationship:-

In Uses-A relationship a method of one class is using an object of another class

e.g:
```
class c1
{ .....
  .....
}
class c2
{ void disp()
  { c1 o1 = new c1();
    ......
  }
}
```
Uses-A relationship

In this relationship as long as disp() method of C_2 class will be executing object of C_1 will be live & when disp() method completes its execution an object of C_1 class will be destroyed.

NOTE:-

1) The default relationship in java is 'is-A relationship' Because for each & every class in java there exist an implicit predefined superclass known as ~~java~~ java.lang.Object.

2) System.out is the universal example for has-A relationship because out is an object of PrintStream class created as static data member in an another predefined class called System. The relationship betn system & PrintStream class is known as has-A relationship.

3) Each & every execution logic method (main()) of execution logic class is using an object of Business logic class. is known as uses-A relationship.

* 'Super' keyword :-

Super is one of the implicit keyword created by JVM & supplied to each & every java program for 3 purposes in other words super keyword playing an important role in 3 places (3 purposes) They are,

1) At variable level.
2) At method level.
3) At constructor level.

① Super at Variable level :-

Whenever we inherit the base Class data member into derived class, there is a possibility that base class data members are similar to derived class data members & JVM gets an ambiguty.

In order to differentiate betn base class data-members & derived class data members, In the context of derived class the base class data membe must be preceded by a 'super' keyword

Syntax :- super.(base class data member name);

If we don't write super keyword before the base class data member name then it is considered as current class data member & it is ~~preceded~~ hidden in the context of derived class.

e.g: Write a java program which illustrate the concept of super keyword at variable level.

```
class BC
{
    int a;
}

class DC extends BC
{
    int a,c;
    void set (int x, int y)
    {
        super.a = x;
        this.a = y;
    }
```

```
    void add()
    {
        c = super.a + this.a;
    }
    void disp ()
    {
        sop(" Val of a (BC): " + super.a);
        sop (" val of a (DC): " + this.a);
        sop(" sum = " + c);
    }
}

class sdemo
{
    psvm (String args[])
    {
        DC do1 = new DC();
        do1.set ( Integer.parseInt(args[0]), Integer.
                                parseInt (args[1]) );
        do1.add();
        do1.disp();
    }
}
```

Q. What is the difference between 'this' & 'super' keyword?

ANS: This kw always points to current class features
super kw always points to superclass features.

(2) Super at method level :-

Whenever we inherit base class methods into derived class, there is a possibility that base class methods are simillar to derived class methods & JVM gets an ambiguity.

To differentiate betn base class methods & derived class methods, in the context of derived class methods base class methods can be referred by using super keyword.

Syntax:- super.(baseclassmethod name);

If we don't write super keyword before the base class methodname then JVM will call the derived class method infinitely & base class method are hidden in the context of derived class.

NOTE:- Any method called by itself infinetly then stack memory becomes full & such type of error is known as StackOverflowError.

Q. Write a Java program which illustrate the concept of super keyword at method level.

```
// Sdemo.java
Class BC
{
    void disp()
    {
        sop("Base class:disp()");
    }
}
```

```
class DC extends BC
{
    void disp()
    {
        sop("Derived class: disp()");
        super.disp();  // call super class disp() from the
                          context of derived class.
    }
}

class sdemo
{
    psvm(String args[])
    {
        DC do1 = new DC();
            do1.disp();  // will call derived class disp().
    }
}
```

Super at Method overridden level :-

We know that method overridding is equal to method heading is same plus method body is different.

To implement method overriding concept we need to use inheritance concept.

Original methods always presents in base classes and overridden methods always presents in derived classes

Q. Write a Java program which illustrate the concept of super keyword at method overridden level.

```
// Sdemo.java
Class BC
{
    void op ( int x, int y)
    {
        Sop(" OP(): original: BC");
        int z = x+y;
        Sop("Sum in BC = " + z);
    }
}
Class IbC extends BC
{
    ~~class~~ void op (int x, int y)
        {
            super.op(x, y);
            Sop (" op(): overridden: IbC");
            int z = x-y;
            Sop(" Sub in IbC:" + z);
        }
}
class DC extends IbC
{
    void op ( int x1, int x2)
    {
        super.op(x1, x2);
        Sop(" op(): overridden: DC");
        int z = x1 * x2;
        Sop (" mul in DC:" + z);
    }
}
```

```
class sdemo
{
    psvm ( String args[] )
    {
        DC do1 = new DC();
            do1.op( 10, 20);
                :OR:
    // new DC().op (Integer.parseInt(args[0]),
                    Integer.parseInt(args[1]));
    }
}
```

In the above program new DC() is known as nameless/anonymous class object

Q. How do you call Base class methods from the context of Derived class methods where both the methods are same?

ANS: super.<base class method name> ~~to the~~

The above stmt must be written in the context of derived class.

Q. How do you call original method from the context of overridden methods?

ANS super. <original method name>

The above stmt must be written in the context of overridden method of derived class.

Whenever we apply super kw at variable & method level then it can be used betn Immediate base & derived classes but not between bottommost derived

Super keyword either at variable level or method level can be applied at only one level.

super.super.methodname is not valid.

8/8/13

③ Super at constructor level:-

Whenever we develop any Inheritance appl^n it is highly recommended to create an object of bottommost derived class because it obtains features from topmost base class & Intermediate base class.

When we create an object of bottommost derived class, first we get the memory space for datamemb of topmost base class, second we get memory space for datamembers of intermediate base class & at last we get memory space for datamembers of bottommost derived class.

In whatever the order memory space is created in the same order data members will be initialized (otherwise we get compile time error) i.e. first topmost base class data members will be initialized second intermediate base class data members will be initialized & at last bottommost derived class data members will be initialized. We know that initializing the data members requires constructors concept.

Hence in any Inheritance appl^n development, constructors are calling from bottom to top. & execution is from top to bottom.

In order to establish communication betn base class constructors & derived class constructors we have two implicit functions They are,

super ()

super (- - - -)

super () :- It is used for calling super class default constructor from derived class constructors.

super (- - - -) :- It is used for calling super class parameter as constructor from derived class constructors.

Rule :-

Whenever we use either super() or super(---) in the derived class constructors, they must be used as a first executable stmt. otherwise we get compile time error (because before executing derived class constructors first we must execute super class constructors).

Possibilities of using super() and super(---)

The following diagram gives possibilities of using super() & super(---)

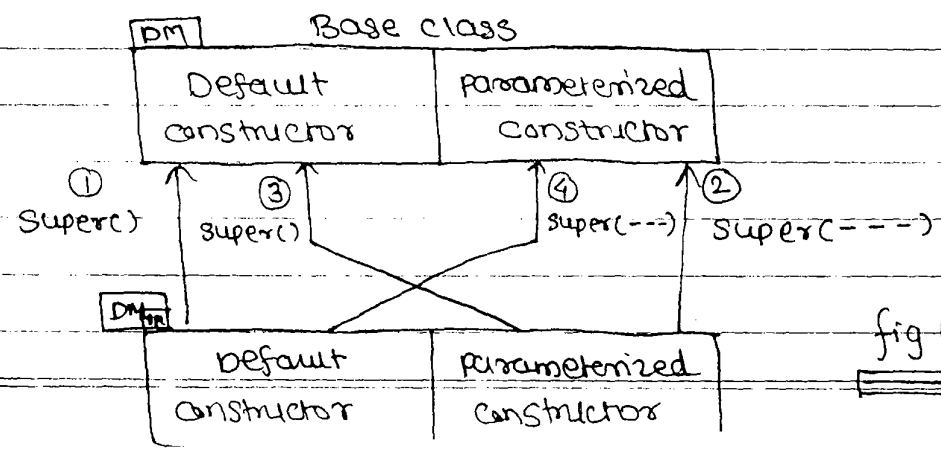


fig ①

Rules ① & ③ :-

Whenever we want to call base class default constructor from the context of derived class constructor in the context of derived class constructor we use super().

Using super() in the context of derived class constructor is optional because base class will contain single form of default constructor.

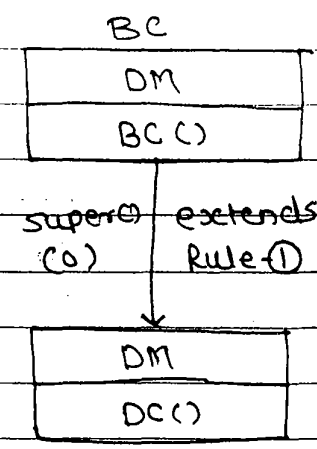
Rules ② & ④ :-

Whenever we want to call base class parameter as a constructor from the context of derived class constructor in the context of derived class constructor we must use super(---)

Using super(---) in the context of derived class constructors is mandatory because a base class may contain multiple parameterized constructor

Q. Write a java program which will illustrate Rule ① of fig. ①.

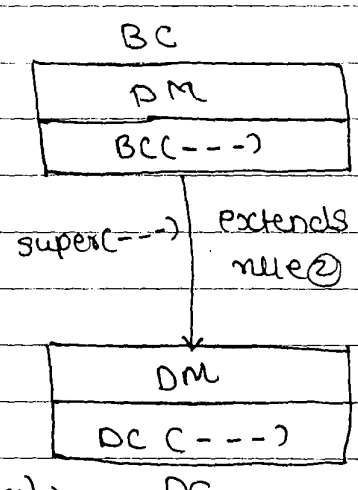
```
class BC
{
    int a;
    BC()
    {   // ........... ②
        SOP(" BC => Default Const");
        a=10;
        SOP(" val of a from BC:"+a);
    }
}
```



```
class DC extends BC
{
   int b
      DC()  //------------- ①
      { super(); // optional - control goes to ②
        sop(" DC => Default const");
        b=20;
        sop(" val of b from Dc:"+b);
      }
}

class Sdemo
{
   psvm(String args[])
   {
      new DC();    // control goes to ①
   }
}
```

Q. Write a java program which illustrate rule ② of fig ①.




```
class BC
{
   int a;
   BC(int a) ~~int y~~
   {
      sop(" BC => Default const");
      this.a= a;
      sop(" val of a from BC:"+this.a);
   }
}
```

```
class DC extends BC
{
    int b;
      DC ( int x, int y)
      {
          super( x);
          Sop( " DC: default Const");
           b=y;
          Sop( " val of b from DC: " + b);
      }
}

class sdemo
{
    psum ( String args [])
    {
    new DC ( 100, 200);
    }
}
```

Q. write a java program which illustrate rule ③ of fig ①

```
class BC
{
    int a;
    BC ()
    { Sop( " BC: Default constructor");
      a=10;
      Sop( " val of a from BC:" + a);
    }
}
```

```
Class DC extends BC
{
    int b;
      DC ( int x)
      { sop(" DC: Default param const");
          this.b = x;
      }
}

class Sdemo
{    psvm ( String args[])
      {
          new DC (20);
      }
}
```

Q. Write a java program which will illustrate Rule ④ of fig ①.

```
class BC
{
    int a;
     BC ( int a )
     { sop(" BC: Default param const");
         this.a = a;
         sop(" val of a from BC:"+ a);
     }
}

class DC extends BC
{
    int b; ~~sum~~; c = 20;
```

```
        DC()
            super();
        {   SOP(" DC: Default const");
            b = 30;
            SOP("val of b from DC1" +b);
        }
    }

    class sdemo
    {
        psum( string args[])
        {
            new DC();
        }
    }
```

9/8/13

④ Polymorphism :-

It is one of the distinct principle of OOPS.

Defination :- "The process of representing one form in multiple forms is known as polymorphism".

Polymorphism is not a programming concept, it is one of the principle.

In Java programming the polymorphism principle can be implemented by using method overridding concept.

We know that method overridding is equal to method heading is same plus method body is different.

Polymorphism principle is classified into two types They are,

1) **Static** | Compile time polymorphism.
2) Dynamic | Runtime polymorphism.

Java programming never follows static polymorphism but it always follows Dynamic polymorphism.

Consider the following diagram,

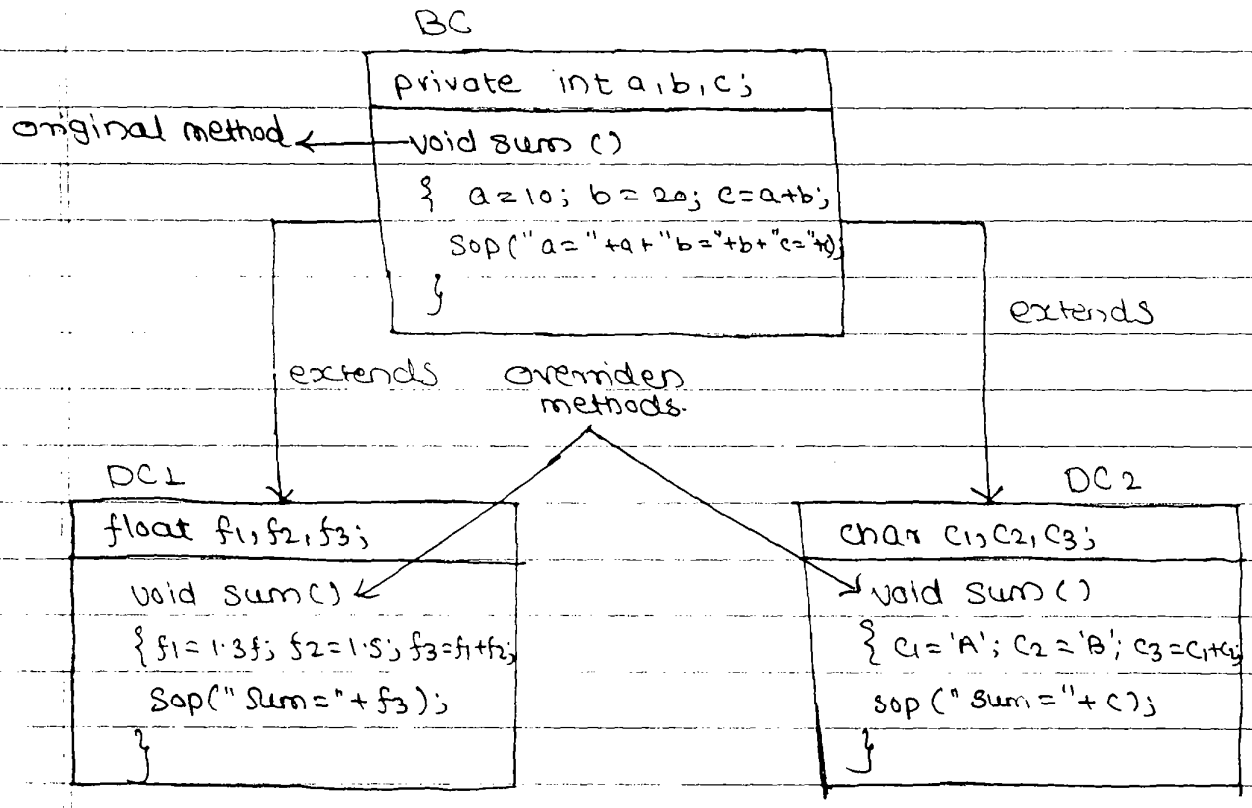


fig ②

In the above diagram BC, DC1 & DC2 are called dependent classes.

Originally in BC class we have one form of sum method (original form) & it is further

Implemented in multiple forms (overridden methods in the derived classes. Hence sum() method is called polymorphic method.

Hence in the industry most of the time Business logic will be developed in terms of business logic classes by following polymorphism along with method overridding.

10/8/13

(5) Dynamic Binding :-

Dynamic Binding always says " Don't create the objects of Derived classes But create an object of Base Class ".

Dynamic Binding principle always used for executing polymorphic based applications.

Advantages of polymorphism along with method overridding plus Dynamic Binding are,

1) Application will take less memory space
2) Application will take less execution time
3) Application gives more performance.

Defination:- The process of binding appropriate versions (overridden methods) of derived classes which are inherited from Base class with Base class Object is known as Dynamic binding.

If we use Dynamic Binding principle for executing Polymorphic based applications, internally JVM will consider following 2 points.

i) What type of object.
ii) What type of Reference/address object contains

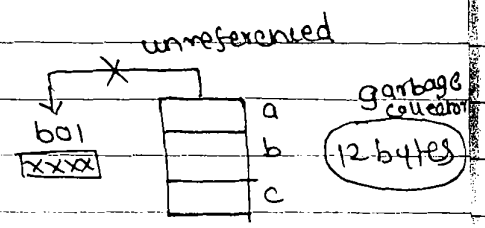
In other words If we create an object indirectly then JVM will consider above two points internally.

NOTE:- ① In order to create an object Indirectly then there must exists super-sub relationship / Base-Derived relationship / master-detailed relationship

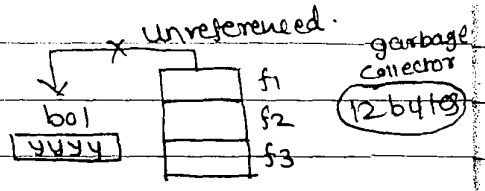
② If we create an object directly (Test t1 = new Test();) then such object creation is known as plain old execution logic & it is not necessary to have super-sub relationship.

e.g: The following set of executable statements which are used for executing the program represented in the fig ② (see page 120) by making use of Dynamic Binding principle.

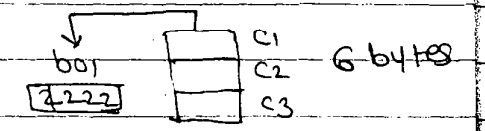
```
BC bo1 = new BC();
bo1.sum();
```




```
bo1 = new DC1();
bo1.sum();
```




```
bo1 = new DC2();
bo1.sum();
```



In the above example in Line no. 1, 3, 5 the object bo1 contains reference of BC, DC1, DC2. Hence bo1 object is known as <u>Polymorphic object</u>.

The method sum() is actually available in one form (original method) & it is implemented in multiple forms (overridden methods) hence sum() method is known as Polymorphic method.

The statement bo1.sum() is actually only one statement & it gives Intsum, floatsum & charsum hence this statement is known as polymorphic statement.

In real world to develop any Java appln we have 3 cases

Case I :- Business logic will be developed in Independent classes & execution logic will be developed with plain old execution logic (Direct Object Creation).

Case II :- Business logic will be developed in Dependent classes (polymorphism + method overriding used) & execution logic developed with plain old execution logic (Direct object creation)

If we develop any Java application with above two cases then we get the following limitations

1. Appln takes more memory space.
2. Appln ~~to~~ execution time is more.
3. Appln performance is reduced.

Case III :- Business logic will be developed in dependent classes (Contains polymorphism + method overridding) & execution logic will be developed by using dynamic Binding principle.

If we develop any Java application by using Case III then Java appln will get following advantages

1. Less memory space
2. Less execution time
3. more performance.

C++ follows both static & Dynamic polymorphism.
C++ developers uses only dynamic polymorphism.

Hence Industry is always recommended to use case III for developing any Java appln but not recommended to follow case I & II due to their limitations.

Q. Define static & Dynamic polymorphism.

In Static polymorphism the overloaded methods binds with an object at compile time by differentiating its signature.

Due to Static polymorphism we get utilization of the resources are very poor. (especially main memory)
In Dynamic polymorphism The methods are binding at runtime by differentiating their signatures.

Even though C++ satisfies both Static & Dynamic polymorphism, C++ developers never follow static polymorphism because of its limitations & they always follows Dynamic polymorphism. In C++ static polymorphism principle implemented by function overloading & operator overloading. In C++ dynamic polymorphism will be implemented by virtual & pure virtual functions.

Even though function overloading concept is same in both C++ & Java, in C++ overloaded methods will be binding with an object at compile time whereas in Java programming overloaded methods binds with an object at runtime

Classes → Concrete classes.
→ Abstract classes

Due to the limitations of static polymorphism Java never follows static polymorphism but it always follows Dynamic polymorphism.

(Mon) 12/8/13

* Abstract Classes :-

We know that each & every Java program must starts with a Concept of class. In otherwords without class we can not write a single Java program.

In Java programming we have two types of classes they are,

1. Concrete classes.
2. Abstract classes.

1. Concrete classes :-

Defination:- A concrete class is one which contain fully defined methods.

In Java programming defined methods of a class are also known as concrete / implemented methods

Once the class is concrete we can instantiate its object directly.

e.g:
```
Class Test
{
    void disp()
    { sop(" Test: Disp"); }

    void show()
    { sop(" Test: show"); }
}
```

method prototype ⇒ idea about method.
prototype establish bridge betn funn call & funn defn.

Here the class Test contains 2 defined methods so that Test is one of the concrete class & we can instantiate its object directly.

```
e.g: Test t1 = new Test();
     t1.disp();
     t1.show();
```

★ Concrete classes always contains specific features and they always deals with specific requirements (suitable to individual programmers)

Concrete classes are not meant for dealing with common requirements (suitable for all programmers). If we choose concrete classes for dealing with common requirements then such appln will get following limitations,

1. Appln will take more memory space.
2. Appln execution time is more.
3. Appln performance is degraded.

To avoid the above problems of concrete classes in dealing with common requirements we use another type of classes concept called Abstract Classes.

(2) Abstract Classes :-

Defination:- An abstract class is one which is containing some defined methods & some undefined methods.

In Java programming undefined methods are called unimplemented / abstract methods

Abstract method (Defn):- An abstract method is one which contains only declaration / prototype but it never contains any defination / body.

In order to make any undefined method as abstract method then undefined methods declaration must be preceded by '<u>abstract</u>' keyword.

Syntax for Abstract method :-

abstract returntype methodname (list of formal parameters types if any)

eg: abstract void op (int, int);
    abstract void sum ();

☆ abstract keyword always makes JVM to understand following two points.

① What a method can do?
   But it never gives,
② How a method can be done?

We know that all the methods of java belongs to a class including abstract methods.

If a class of java contains abstract methods Then that class is known as abstract class & whose defination must be made as abstract by using abstract keyword.

Syntax for Abstract class :-

```
abstract class <clsname>
  {
      .......
      .......
      abstract ReturnType methodname( List of formal parameter type if any).
       {
         .......
         .......
       }
  }
```

example:
```
abstract class op
  {
      .......
      .......
      abstract void sum();
      ......
      ......
  }
```

Op o1 = new Op() // Invalid

Because Op is abstract.

In general once the class is abstract one can not create Instantiate its object directly But it can be created indirectly.

Advantages of Abstract classes:-

If we develop any Java appl^n with the concept of abstract classes then such appl^ns will get following advantages.

① Appl^n memory space is less (main memory)
② Appl^n execution time is less
③ Appl^n performance is more

Q. Write a Java program which will calculate area of different shapes.

```
abstract class Shape1
{
    abstract void area ();
}

class square extends Shape1
{
    int a;
    void area()
    {
        a=10;
        int area= a*a;
        sop(" area of square=" + area);
    }
} // square

class Rect extends Shape1
{
    int b, l;
    l= 10;
    b= 20;
```

```
        int area = l * b;
        sop(" area of Rect = " + area);
    }
} // Rect.

class TestArea
{
    public static void main ( String args[])
    { // ShapeL So = new ShapeL(); //error coz ShapeL
      ShapeL So = new Square();      is abstract.
            So.area();
            So = new Rect();
            So.area();
    }
}
```

For further Notes Refer NoteBook 2

* Points to Remember About Abstract Classes :-

1) Abstract classes always contains common features for dealing with common requirements.

2) Abstract class features are always reusable.

3) Since Abstract classes are always reusable, Their defination should not belongs to final hence Java programming doesnot contain final Abstract classes.

4) Abstract classes of Java are not recommended to contain programmer defined constructors because they are not directly referencing.

5) By default each & every Abstract class contains system defined default constructor.

6) Abstract methods of Abstract class are always belongs to Abstract Instance methods but not Abstract static methods because every Instance method meant for performing repeated operations & every static method is meant for performing one time operations.

7) Abstract classes of Java makes use of polymorphism & method overridding for business logic development & makes use of Dynamic Binding principle for execution logic.

8) An object of Abstract Class cannot be Instantiated directly but it can be instantiated indirectly.

i) An object of Abstract Class is equals to an object of its subclass.

ii) An object of Abstract Class is equals to an object of that class which extends the abstract class.

iii) An object of subclass of abstract class is object of abstract class.

9) An object of Abstract class contains the details about those features which are present in the same Abstract class but an object of Abstract class ~~ee~~ never contains the details about those features which are specially available in its derived classes. (Known as Scope of Abstract class object).

10) An object of Abstract class can be declared but it cannot be referenced directly & it can be referenced indirectly wrt its concrete subclass.

```
AC ao1;   // valid Abstract class Declaration
ao1 = new AC();  // Invalid coz Direct reference
                    of AC()

ao1 = new CC();  // valid coz referenced
                    indirectly.
```

Q. Write a java program which will calculate sum of any two numbers by using Abstract class.

ANS:

```
abstract class sum
{
    abstract void add ();
}

class Isum extends sum
{
   int a, b;
      ~~add~~ (Isum( int a, int b)
        {
            this.a = a;
            this.b = b;
        }
   void add ()
      {
         int K = a+b;
         Sop (" Sum of Int = " + K);
      }
}

Class Fsum extends sum
{
   float f1, f2;
      Fsum ( float f1, float f2)
      {
         this.f1 = f1;
         this.f2 = f2;
      }
```

```
void
~~float~~ add()
    {
        float f3 = f1 + f2;
        Sop("Sum of float = " + f3);
    }
}

class Ssum extend sum
{
    String s1, s2;
        ssum(String s1, String s2)
        {
            this.s1 = s1;
            this.s2 = s2;
        }
    void add()
        {
            String s3 = s1 + " " + s2;
            Sop("Sum of Strings =" s3);
        }
}

class SumDemo
{
    psvm(String args[])
    {
        Sum so1 = new ISum(Integer.parseInt(args[1]),
                Integer.parseInt(args[2]));
            so1.add();
        so1 = new FSum(Float.parseFloat(args[2]),
                Float.parseFloat(args[3]));
            so1.add();
```

```
        So1 = new Ssum(args[4], args[5]);
        So1.add();
    }
}
```

14/8/13

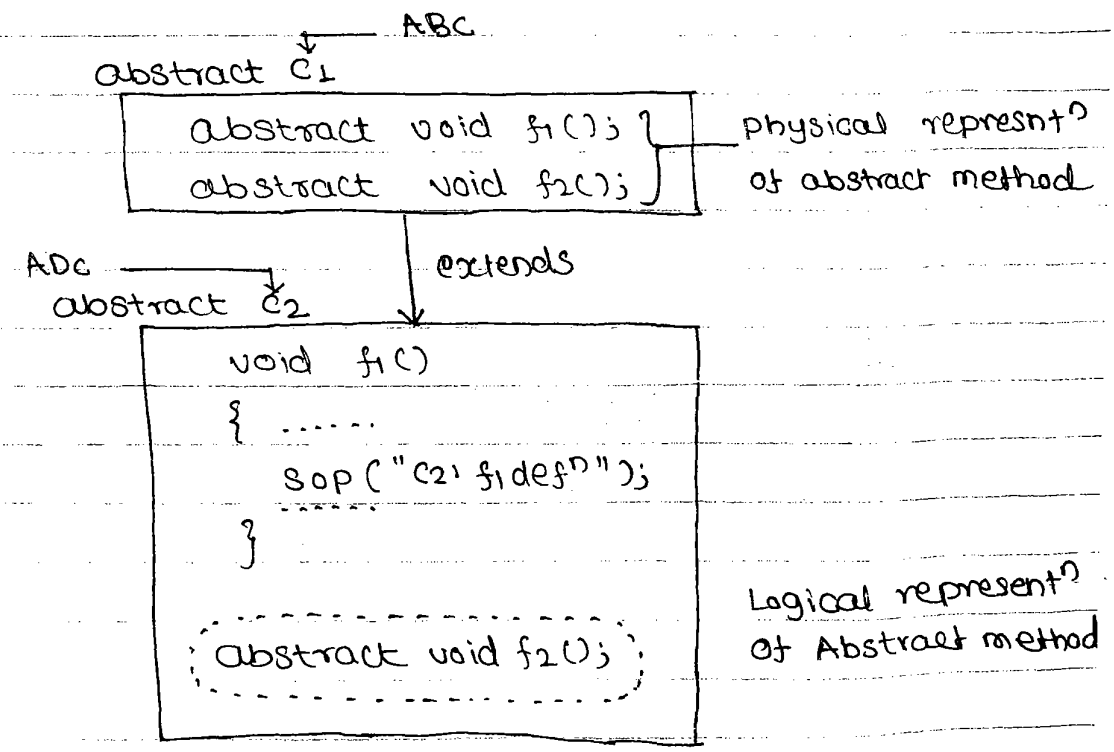
* Abstract Base Class & Abstract Derived class :-

Abstract Base class:-

Defination:- An abstract base class is one which is containing physical representation of abstract methods

Abstract Derived Class:-

Defination:- An abstract derived class is one which is containing logical representation of abstract methods which are inheritted from abstract base class.



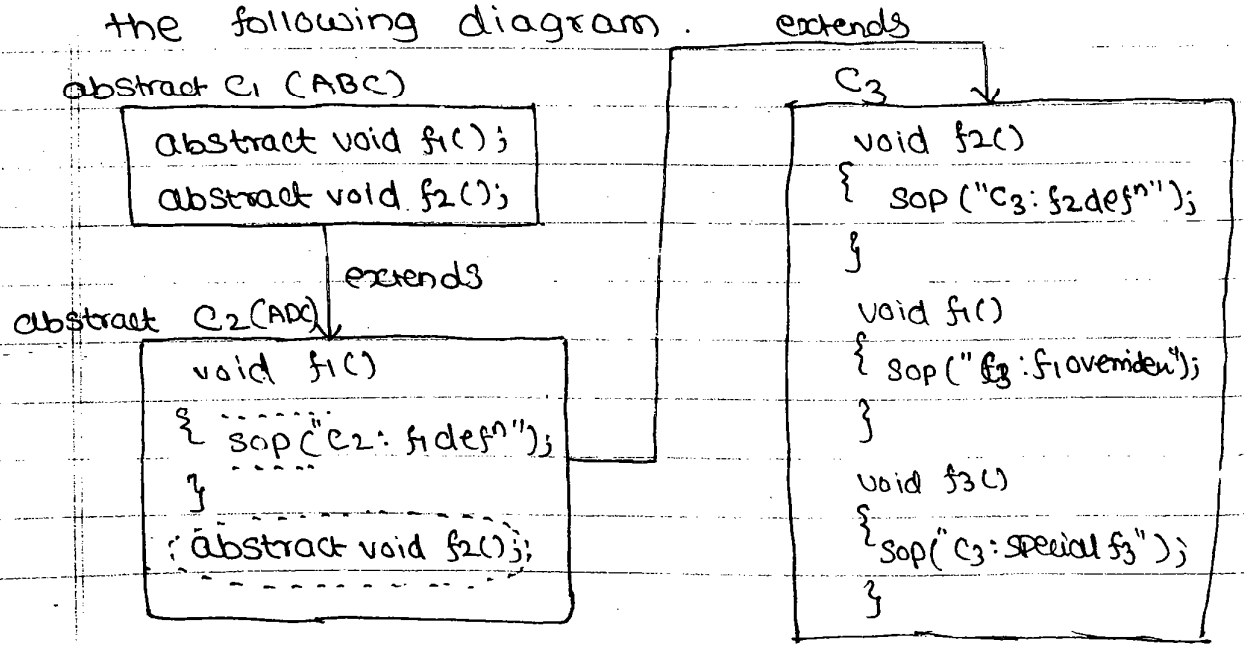
Since both these classes are abstract, they are always reusable.

Wrt ABC class & ADC class one can not create an object directly but we can create their objects indirectly.

When the derived class inherits multiple abstract methods from abstract base class & if the derived class is not defining atleast one abstract method then the current derived class is known as abstract derived class & whose defination must be made as abstract by using abstract keyword

If the derived class defines all the abstract methods which are inheritted from abstract base class then the current derived class is known as concrete derived class

Q. Write a Java program which will implement the following diagram.



ANS:

```
abstract Class ABC
{
    abstract void f1();
    abstract void f2();
}

abstract class ADC extends ABC
{
        void f1()
        {
            SOP (" ADC: f1 defn");
        }
}

class CDC extends ADC
{
    Void f2()
    {
        SOP(" CDC: f2 defn");
    }
    Void f1()
    {
        SOP(" CDC : f1 overridden");
    }
    void f3()
    {
        SOP ("CDC : f3 special");
    }
}
```

```
class AbTest
{
    psvm ( String [] args)
    {
        Sop(" wrt CDC ⇒ POBL");
            CDC  co = new CDC ();
                 co.f1 ();
                 co.f2 ();
                 co.f3 ();
        Sop(" wrt ADC ⇒ DB");
        // ADC  ao1 = new ADC();  // Invalid ... ADC is abstract
        ADC  ao1 = new CDC ();
             ao1.f1 ();
             ao1.f2 ();
        // ao1.f3 ();  // Invalid co2 f3() doesn't exist in ADC
        Sop(" wrt ABC ⇒ DB");
        // ABC  ao2 = new ABC();  // Invalid co2 ABC is abstract
        // ABC  ao2 = new ADC();  // invalid co2 ADC is also abstract
        ABC  ao2 = new CDC();
             ao2.f1();
             ao2.f2();
        // ao2.f3();  // Invalid co2 f3() doesn't exist in ABC
    }
}
```

Observation:-

An object of abstract class can not be instantiated either wrt abstract base class or indirectly wrt abstract derived class but it is always possible to create wrt its concrete sub class.

Consequence's of Abstract classes :-

If any concrete class of Java contains purely Null body methods. then calling those Null body methods wrt corresponding concrete class object is of no use because Null body methods never gives any result. Hence it is highly recommended to the Java programmer to make such concrete classes as abstract by using "abstract" keyword.

Null Body Method :-

Defination :- A null body method is a defined method without block of statements.

Syntax :- ReturnType method name (List of formal parameters if any)

```
{
                    // no block of stmt.
}
```

Null body methods always present in abstract concrete classes & always they are suppose to be overridden in the context of derived classes.

Q. Write a Java program which illustrate the concept of Null body methods of abstract concrete class.

```
// NullbTest.java
abstract class parent
{
    void goc()    // null body method.
    {
    }
}
```

```
class child extends Parent
{
    void goC()
    {
        SOP(" going to college each & every day");
    }
}

Class NullbTest
{
    psvm (String args[])
    {
        SOP(" wrt Parent => DB");
    // Parent Po = new Parent();  o/z it is abstract.
        Parent Po = new Child();  // Indirect object
            Po.goC();
    }
}
```

15-Aug-13

(2)

If any concrete class of java contains collection of concrete methods with block of statements & they are not at all providing any suitable solution to the client requirement. Then there is no use of calling such methods wrt corresponding concrete class object. Hence it is highly recommended to make such type of concrete classes as abstract by making use of abstract keyword.

Q. write a java program which illustrate the above Consequence

```
abstract class Op
{
    void sum()
    {
        Sop(" I am from sum() without doing any sum");
        Sop(" Don't call me");
    }
}

Class Isum extends Op
{
    int a, b, c;
    ~~void~~ Isum (int a, int b)
    {   this.a = a;
        this.b = b;
    }
    void sum()
    {
        c = a+b;
        Sop (" sum of "+a+" and " +b" is:" +c);
    }
}

Class AbTest
{
    psum (String args[])
    {
```

```
        if (args.length! =2)
         sop(" please enter two values");
        else
        {
          int x= Integer.parseInt(args[0]);
          int y = Integer.parseInt(args[1]);
          sop(" wrt op : DB");
        // op o1 = new op();   // Invalid op = abstract
          op o1 = new Isum (x,y);
             o1.sum();
        }
    }
}
```

Q. Can we make a concrete class as abstract class? what are the reasons?

'Yes' we can make concrete class as abstract.

Reasons:- See consequences of Abstract class consequence ① & ②

③ Industry is always recommended to override always abstract methods of abstract classes but not recommended to override defined methods of concrete classes because method overhead cost of overridding defined methods is more than method overhead cost of overridding abstract methods.

method overhead cost for overridding defined method = cost required for eliminating Existing body + cost required for construction of new body

method overhead cost ~~of~~ for overridding abstract method = cost required for constru-ctiing new Body only.

Method overhead cost reflects on execution time. Overridding of defined methods will take more execution time & overriding of abstract methods will take less execution time.

Q. Write a java program which will print fonts of the system.

```
import java.awt.GraphicsEnvironment;
class Fonts
{
    psvm ( String [] args)
    {
      GraphicsEnvironment ge = GraphicsEnvironment.
                              getLocalGraphicsEnvironment();
      String s[] = ge.getAvailableFontFamilyNames();
      sop(" no. of fonts: " + s.length);
      for( int i=0; i<s.length; i++)
      {
          sop( s[i]);
      }
    }
}
```

16/8/13

* Factory Method :-

A factory method is one whose return type must be similar to the classname in which class it present.

The purpose of factory method is to create an object without using new operator.

Rules for writing factory method:-

1) Return type of the factory method must be similar to the classname in which class it present.

2) The nature of the factory method must be static

3) Access specifier of the factory method must be public.

e.g: Java.awt.GraphicsEnvironment

```
public static GraphicsEnvironment getLocalGrap-
  ③      ②            ①          -hicsEnvironment();
```

Hence most of the predefined abstract classes in java contains factory methods.

* Interfaces :-

We know that concrete classes are always meant for dealing with specific requirements & they are containing specific features.

Concrete classes are unable to deal with common requirements. If we use concrete classes for dealing with common requirements then we get 3 limitations they are,

1. More memory space
2. More execution Time
3. Less performance.

To avoid these problems we use the concept of abstract classes

Abstract classes always meant for dealing with common requirements & it provides the following advantages,

1. Less memory space.
2. partially less execution time.
3. more performance.

While we are using abstract classes at the time of dealing with common requirements, along with their advantages, abstract classes are containing following limitations.

1. Abstract classes never participates in multiple Inheritance.
2. Abstract classes provides only common reusable features & they are unable to provide universal common reusable features
3. We know that abstract class is a collection of both defined & undefined methods Industry is highly recommended to override only abstract methods but

not recommended to override defined methods so abstract class containing both recommended & un-recommended features due to this abstract classes related appln gives partially less execution time & unable to give completly less execution time.

To avoid above limitations of abstract classes, we use the concept of Interfaces.

Interface :-

programmatically concept of Interfaces is always used for developing universal user defined data type.

To develop concept of interfaces, we use a keyword called Interface.

Each & Every Interface name in Java treated as universal user defined data type.

Defination :-

An interface is a collection of public static final xxx data members & public abstract methods.

Here xxx represents datatype, variable name & variable value.

An Interface is a collections of universal common reusable data members & universal common reusable methods.

Syntax for defining an Interface:-

```
interface <interfacename>
{
    var decl cum initialization;
    method declaration;
}
```

In the above syntax,

1) Interface is a keyword used for developing universal userdefined datatype

2) <interfacename> represents a Java valid variable name treated as name of the interface. Each & every interface name in Java is treated as universal user defined datatype. Wrt an interface we can not create its object directly but we can create its object indirectly.

3) var declaration represents data members. We know that every data member of an interface is meant for universally reusable so that they must be initialized otherwise we get compile time error. Since the data members of interface are universally reusable, those values will be used but they can not be changed (final), whose memory space to be created only once (static) & they can be accessed everywhere (public) programmatically,

public static final, float PI = 3.1417f;

4) Methods declaration represents type of undefined methods meant for performing universal operations. To make these undefined methods as abstract, explicitly we need not to write abstract keyword. To make these methods as universally accessible we need not to write public hence by default each & every method of interface belongs to public abstract methods.

ex:- $\underbrace{\text{public abstract}}_{\text{Appended by JVM}}$ $\underbrace{\text{void f1();}}_{\text{written by Java programmer}}$

Q. Define an Interface i1 by taking suitable datamembers & Methods.

```
interface i1
{
    int a=10;
    int b=20;
    void f1();
    void f2();
}
```

Compile the above program.

cmd> javac i1.java.

Ensure i1.java prg will be compiled & i1.class file must be generated.

View the profile of i1

cmd> javap i1

```
interface i1 {
  public static final int a;
```

```
    public static final int b;
    public abstract void f1();
    public abstract void f2();
}
```

Advantages of Interfaces:-

1) Interfaces always participates in multiple Inheritance
2) Interfaces concepts provides universal common reusable features.
3) Since interfaces are containing purely abstract methods sothat overridding of those abstract methods will take less time & results in completly less execution time.

NOTE:- Each & every datamember of interface must be accessed wrt interfacename because interface data members are belongs to static.

17/8/13

* Inheriting The features of Interface:-

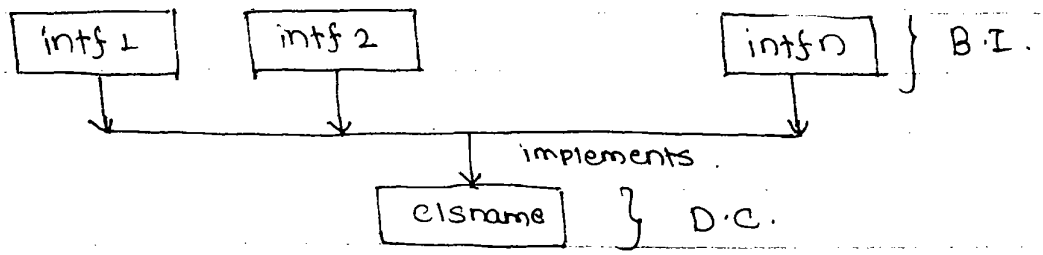
In order to inherit the features of base interfaces into derived classes we have 3 approaches They are,

Approach 1:-

This approach makes us to understand How to inherit the features of base interfaces into derived class.

Syntax:-

```
abstract class <clsname> implements <intf1>,.....<intfn>
        {
            variable declaration;
            method decl^n | Defination;
        }
```



In the above syntax <clsname> represents name of derived class.
intf1, intf2, ---- intfn represents name of the base interfaces.

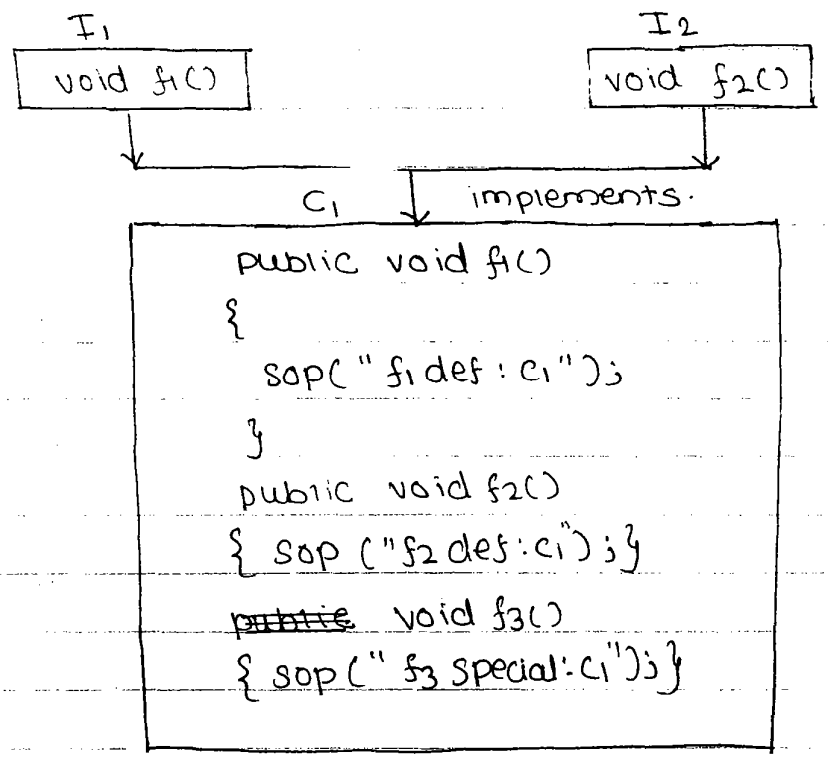
implements is the keyword used for inheriting features of base interfaces into derived class.

In Java programming one derived class can extends only one base class but one derived class can implements either one or more than one Interface because Java programming supports multiple Inheritance through the concept of interfaces.

Whenever the derived class in inheritting multiple abstract methods from multiple base interfaces then if current derived class is not defining

atleast one method then the current derived class is made as abstract by using "abstract" keyword.

Q. Write a Java program which will implement the following diagram.



ANS:

```
interface i1
{
    void f1();    // public abstract void f1()
}
interface i2
{
    void f2();    // public abstract void f2()
}
```

(12)

```
Class C1 implements i1, i2.
{
    public void f1()
    {  sop (" f1() defn : C1"); }

    public void f2()
    {  sop (" f2() defn: C1"); }

    void f3 ()
    {  sop (" f3() special: C1"); }
}

Class InTest
{
    psvm( String args[])
    {
        sop(" wrt C1:POBL");
        C1 o1 = new C1();
            o1.f1();
            o1.f2();
            o1.f3();
        sop (" wrt i1: DB" );
        // i1 io1 = new i1();   co2 i1 is abstract  Invalid
        i1 io1 = new C1();  // indirect obj
            io1.f1();
            io1.f2();  } Invalid co2 f2() & f3()
            io1.f3()   } does not present in i1
        sop (" wrt i2: DB" );
        i2 io2 = new i2();  Invalid co2 i2 is abstract
        i2 io2 = new C1();  // indirect obj
            io2.f2();
    }
}
```

1. Interfaces always contains universal common reusable features.

2. Interfaces features are always participates in Inheritance.

3. Interface definations should not be final because they are always participating in Inheritance process.

4. Interfaces does not contain constructors because of the following reasons.
   a) Interface data members are already initialized
   b) Constructors are the special defined methods and such defined methods are not permitted in Interfaces.

5. While we are Defining Interface methods in its subclass, Interface methods must be always preceded by public (otherwise we get compile time error).

6. Interfaces of Java always contains public abstract instance methods only but not containing public abstract static methods because every instance method can be overridden for performing multiple operations whereas it is not recommended to override static methods. Because they meant for performing one time operation.

7. An object of an interface can not be created direct

But it can be created indirectly.

a) an object of an interface is equal to an object of its subclass.

b) An object of an interface is equal to an object of that class which implements the interface

c) An object of subclass of an interface is an object of an interface.

(8) The default modifier of an interface is abstract

(9) Interfaces of Java makes use of polymorphism plus method overridding for business logic development and makes use of dynamic binding principle for development of execution logic

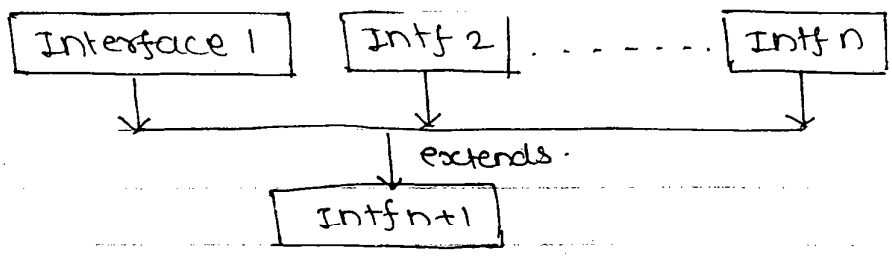
(10) An object of an interface contains details about those features which are present in the same interface but an object of interface does not contain details about those features which are specially available in other interfaces and in its subclass

(11) Main method of Java can not be written as a part of Interfaces But it can be written as a part of concrete classes and abstract classess

Approach 2:- (Interface Inheritance).

This approach makes us to understand how to inherit the features of base interfaces into derived interface.

The mechanism of inheriting the features of base interfaces into derived interface is known as interface inheritance



Syntax:-

```
interface <intf n+1> extends <intf 1>,..... <intf n>
 {
     variable decl^n cum initialization;
     method decl^n;
 }
```

Explanation:-

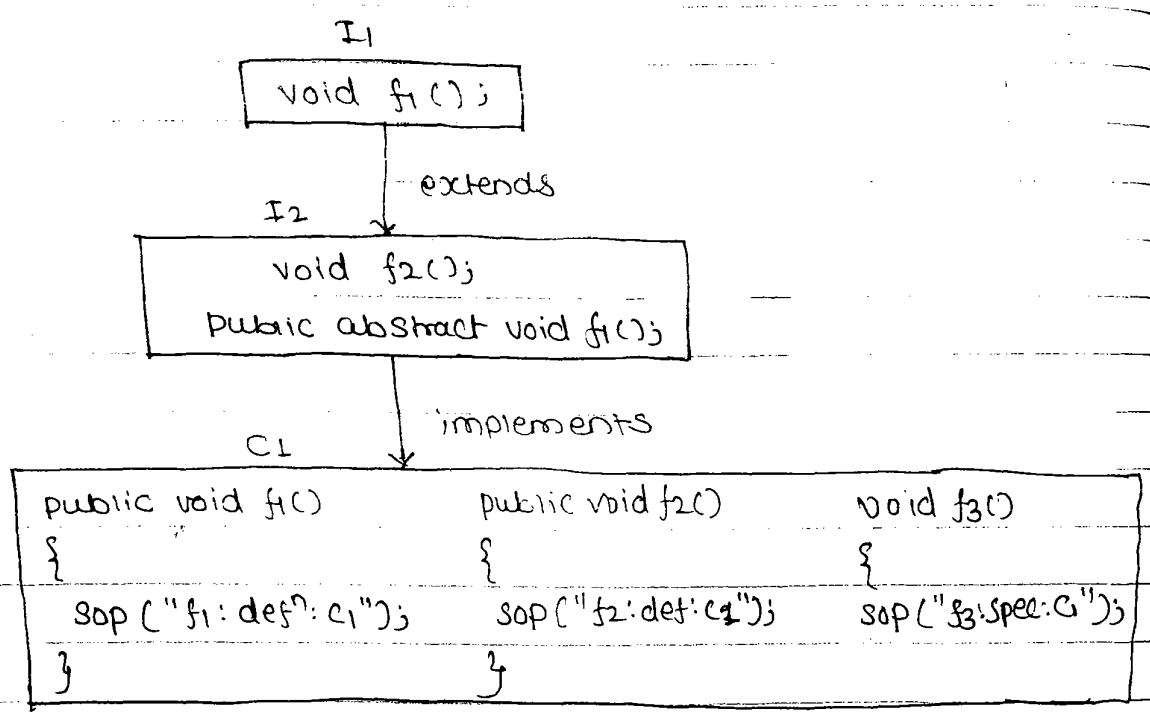
In the above syntax <Intf 1>,<Intf 2>..... <Intf n> represents name of base interfaces. <Intf n+1> represents name of the derived interface.

Extends is a keyword used for inheritting the features of base interfaces into derived interface

In Java programming one derived class can extend only one base class whereas one derived interfac can extends either one or more than one interface

because multiple inheritance is supported through the concept of interfaces.

Q. Write a Java program which will implement the following diagram



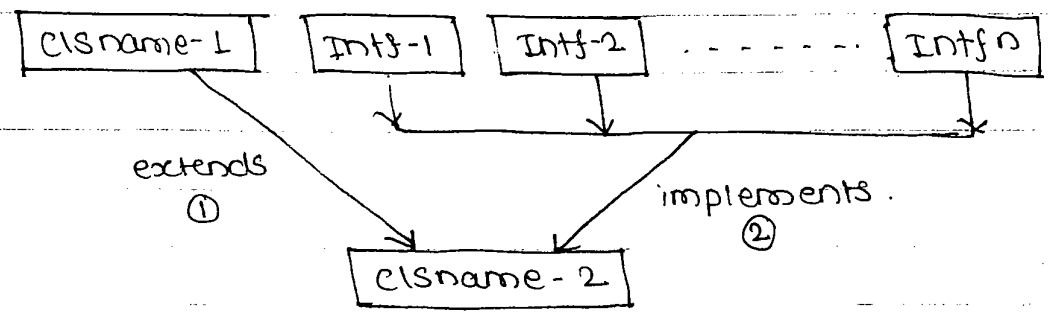

```
interface i1
{
    void f1();
}

interface i2 extends i1
{
    void f2();
}
```

```
Class C1 implements i2
{
    psvm (String args[])
    {
}
```

Approach 3:- (Parallel Inheritance).

This approach makes us to understand how to inherit the features of base class and from base interfaces simultaneously to the derived class.



Syntax:-

```
[abstract] class <clsname-2> extends <clsname-1>
                implements <intf-1> ..... <intfn>
    {
        variable declaration;
        method defn | Decln;
    }
```

Explanation:-

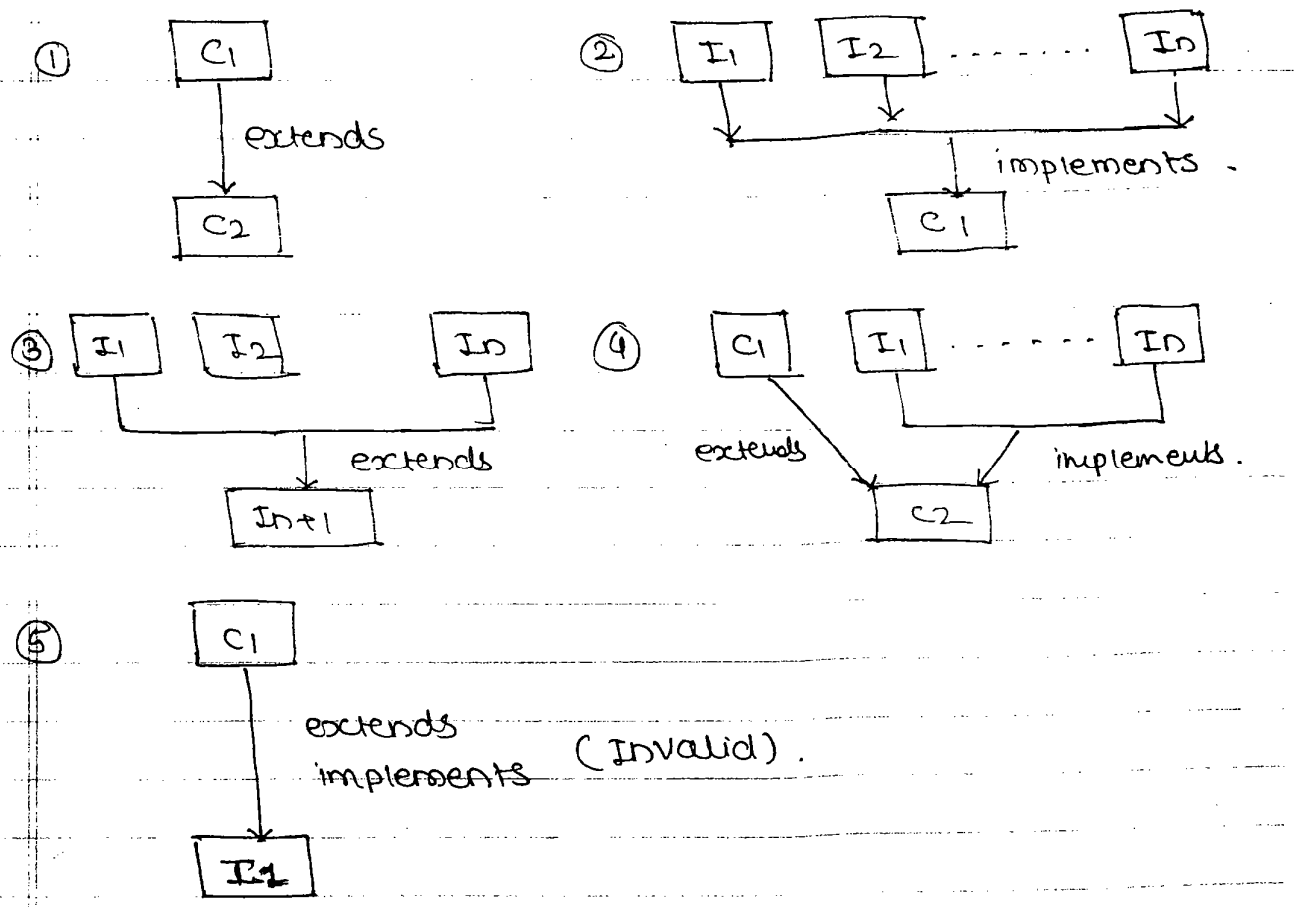
In the above syntax,

<clsname-1> & <clsname-2> represents name of the base & derived classes.

<Intf-1>, ...; <Intfn> represents name of the base interfaces.

extends & implements are two keywords used for parallelly inheriting features of single base class & from interfaces.

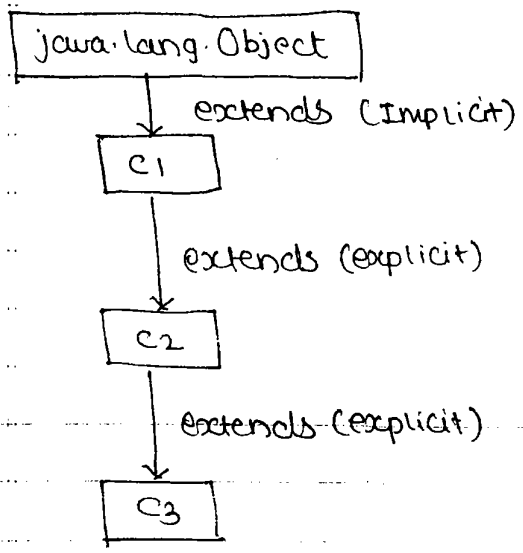
If we use extends & implements in a single syntax it is mandatory to the java programmer to write extends keyword first & later we write implements keyword otherwise we get compile time error.



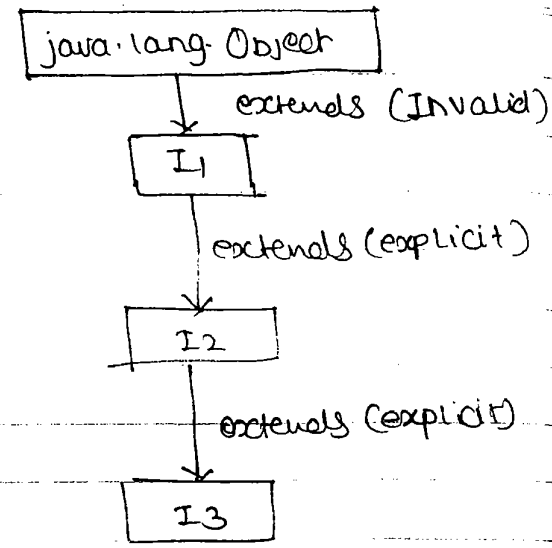
a) Because specific features of concrete class & common features of abstract class can not become as universal common features of interfaces.

b) Defined methods of both concrete & abstract classes will not become undefined methods.

(6)



(7)



(8) class C1 implements I1

```
{
    .....
    .....
}
```

write the profile of C1 (Jawap C1)

```
class C1 extends java.lang.Object implements I1
{ C1();
    ....
    ....

}
```

2018.13

* Packages :-

In general learning of any programming language is nothing but learning about its concepts, syntaxes & library.

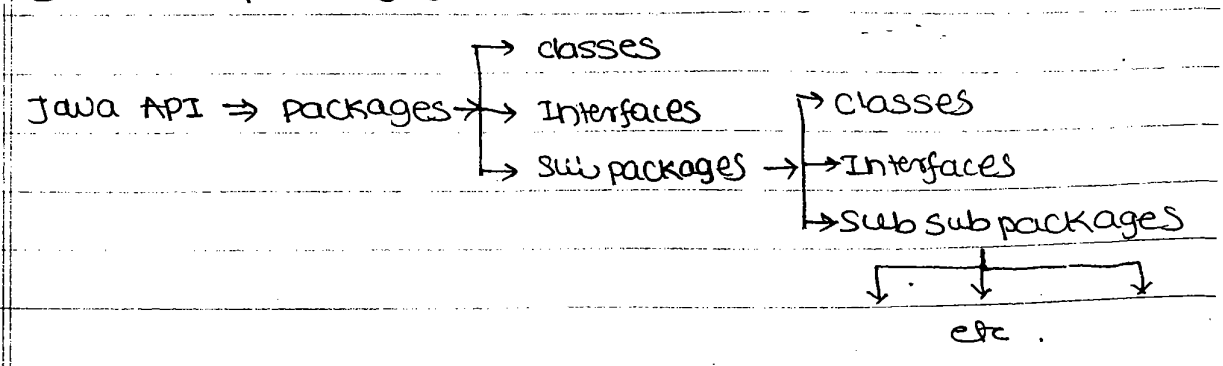
example:- Learning C language is nothing but learning about its concepts, syntaxes and library. We know that library of C language is a collection of Header files. A Header file is a collection of predefined functi

In C slw development If we want to develop any predefined function and it is common for most of the C programmers then they must be placed in a header file. In other words header files always contains common functions for many C programmers.

lly learning Java is nothing but learning about its concepts, syntaxes & oops features & its library / API

An API of Java is a collection of packages

Defination :- A package is a collection of classes interfaces & subpackages. A subpackage inturns contains collection of classes interfaces & sub-sub packages etc.



If any class or Interface is common for many number of Java programmers then such type of classes & interfaces we place in the packages. In other words all the classes & interfaces of a package are common for all the Java programmers

Advantages | Benefits of packages:-

If we develop any appl^n by using the concept of packages then we get the following advantages

1. Appl^n Development time is less
2. Appl^n memory space (main memory) is less
3. Appl^n execution time is less.
4. Appl^n performance is enhanced.
5. Redundancy of code is reduced so that we can get consistent result & less Storage cost.
6. We are able to get slogan of Java i.e. 'WORA'

Q. What is the difference bet^n Inheritance and package?

ANS: Inheritance concept always makes us to understand how to reuse the features within the program bet^n class to class, Interfaces to interface Interfaces to class. but not possible to reuse across the programs

Packages concept always makes us to understand how to reuse the features within the programs & across the programs bet^n class to class, Interfaces to interface & interfaces to class.

In Java packages are classified into 3 types they are ① predefined | Built in packages.
② user | programmer | custom defined packages.
③ Third party packages.

① Predefined packages:-
predefined packages are those developed by SUN ms and supplied as a part of JDK to deal with universal requirements.

② User defined packages:-
user defined packages are those which are developed by Java programmers & supplied as a part of their project to deal with common requirements.

③ Third party packages:-
Third party packages are those which are developed by third party s/w vendors and released to the real industry as a part of third party products (DB's, servers, etc) To communicate with third party products
Example:- $\underbrace{\text{oracle.jdbc.driver}}_{\text{Third party package name}}.\underbrace{\text{OracleDriver}}_{\text{classname}}$.

* Types of predefined Packages:-
predefined packages are classified into 3 types they are
① JSE packages (Core packages)
② JEE packages (Advanced packages)
③ JME packages (Micro | Mobile pckages)

JSE packages are used for developing clientside applications.
All the JSE packages starts with Java.packname*;
JEE packages are used for developing serverside applications and each & every JEE pkg starts with Javax.packname.*;
JME packages are used for developing mobile applications & they also starts with javax.packname.*;

(1) List of predefined JSE packages:-
As a part of JSE we have 8 essential packages they are,
① java.lang.* :-
The purpose of this packge is to provide language functionalities | services | facilities.

Some of the language functionalities are
ⓐ displaying the data on the console
ⓑ Accepting the command line arguments (String)
ⓒ providing the garbage collection facility
ⓓ Converting data from numerical strings into numeric values (wrapper classes, etc).

This is the package which is imported by default to each & every java prg. so that this pck is also known as default package.

② java.awt.*:- (Abstract window toolkit)
The purpose of this package is to design the ~~gui~~ GUI applications (look & feel based appl?)

In the real Industry to develop any GUI appl^n we require GUI components | controls such as labels, Buttons, checkBoxes, etc. In Java programming all the GUI components are available in the form of predefined classes which are present in java.awt.* pkg

Creating a GUI Component is nothing but creating an object of appropriate predefined class.
e.g: create a component Button with a name save.

```
Button b1 = new Button("save");
```

save

③ java.awt.event.*;
Here event is a subpackage of awt package. The purpose of this package is to provide functionality/life/behaviour to the GUI components.

In other words event package contains collection of classes & interfaces which are providing functionality to the GUI application.

Hence to develop complete GUI appl^n we must import java.awt.* (To design GUI appl^n) and java.awt.event.* (for providing functionality to the GUI application).

Event :-

Change in the state of the object is known as an event.
e.g: Button clicked, checkBox checked, mouse clicked, etc.

④ Java.io.* :- (file programming / Stream Handling)

The purpose of this package is to achieve data persistency (storing the data permanently) using files.

As on today Industry is not recommended to store the data permanently in the form of files because files of any programming language are not providing security in the form of username and passwords.

Hence Industry is highly recommended to store the data permanently in the form of popular DB products such as oracle, DB2, mysql, etc because these DB products provides effective security in the form of usernames & passwords

22/8/13

⑤ java.applet.* :- (Applet programming).

The aim of this package is to develop distributed applications.

In the initial days of SUN ms, SUN developers has developed a concept called applets with the intension of developing distributed applications To fulfill the concept of applets we have a predefined class called Applet & it is present in a package called java.applet.* This package contains only one class & whose fully qualified name is java.applet.Applet -

Applet:- An applet is a java program runs in the context of browser | www & whose results are shareable across the globe.

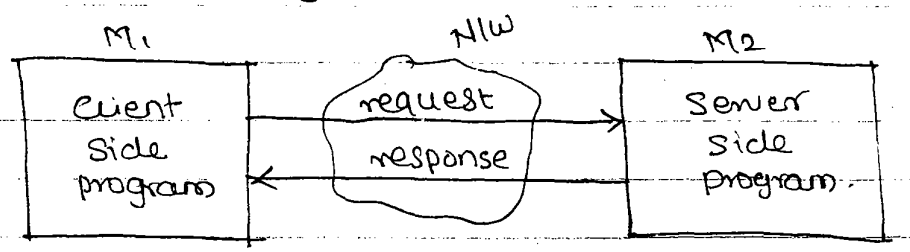
⑥ java.net.* :- (Network programming | socket prg.ing)

The aim of this package is to develop client-server applications.

In client-server appl^n development there exist two types of programs they are,

i) Clientside program.

ii) serverside program



Client side prg always makes the request to get the services from server side program.

The serverside prg receives the client request, process the client request & gives response back to the client.

for Development of both client & server side prgs we require some predefined classes & interfaces which are available in java.net.* package

⑦ java.util.* :- (collection framework).

The basic aim of this package is to improve the performance of all the java appl^ns

Collection framework is one of the additional service or add-on service developed by SUN MS & this mechanism allows us to group multiple values either of same type or different type or both the types in a single variable with dynamic size in nature. This single variable is known as collection framework variable.

Q. What are the differences betn Arrays & Collection framework?

Array	Collection framework
An array is a collective name given to a group of consegative memory locations which are all refered by Homogeneous/ simillar type of values.	Defination as given above.
Arrays concept allows us to organize only homogeneous elements but not Heterogeneous elements.	Collection flw concept allows us to organize both Homogeneous & Heterogeneous elements.
Arrays concept contains fixed size in nature	Collection flw concept contains dynamic size in nature

⑧ java.text.* :- (Text processing).

The aim of this package is that for formatting the dates, formatting the timings, numerical manipulations, misleneous information like finding the day number in an year, finding the week number of the year, etc

This package regourously used in report generation modules of every real world application.

② User defined packages :-

User defined packages are those which are developed by Java programmer & supplied as a part of their project & they deals with common requirements.

The purpose of packages concept is for placing common classes & Interfaces. In other words if any class or interface is common for many number of Java programmers then such type of classes & Interfeices must be placed in a package.

What are all guidelines developed by SUN MS for development of predefined packages, same guidelines will be followed by us for development of user defined packages.

Creating user defined package is nothing but creating it as a folder / directory in the current working m/c

Let us consider the following statement.

java.awt.event.ActionEvent.

In Java programming point of view,
java ⇒ upper package.
awt ⇒ sub package.
event ⇒ sub sub package.
ActionEvent ⇒ class.

In operating system point of view,
java is called root directory.
awt is called sub directory
event is called sub sub directory
ActionEvent is called one of .class file.

23/8/13

* Syntax for Creating a Package:-

To create a userdefined package we have the following syntax.

```
package pack1 [.pack2 [.......[.packn]]];
```

Explanation:-

- package is a keyword used for developing user defined packages.

- pack1, pack2, ...,packn represents java valid variable names treated as name of the user defined packages.

- pack2, ..., packn represents name of the sub packages & whose specification is optional because a package may or may not contain subpackages

- Pack 1 represents name of the upper/outer package & whose specification is mandatory.

- example: package P1;
  package P1.P2;
  } package stmt

Rule :-
When we are placing any class or interface in the package, the package statement must be always used as first executable statement (otherwise we get compile time error)

* Steps / Guidelines for placing Classes & Interface in the package :-

For placing classes & Interfaces in the package SUN developers has prescribed the following guidelines

① Choose an appropriate package name for placing common classes and interfaces. & ensure that the package statement must be first excecutable stmt.

② Choose an appropriate classname/Interface name and ensure whose modifier must be public

③ The modifier of the constructor of the class which is present in the package must be public (this Rule is not applicable in the case of Interfaces bcoz Interfaces doesnot contains constructors).

④ The modifier of the method of the class/Interface which is present in the package must be public (This rule is optional in case of Interfaces

⑤ Whatever the class | Interface we are placing in the package, it should be given as a filename with an extension .java.

⑥ At any point of time either we type a class defination in the package or Interface defination in the package but not recommended both the definations should be typed in the same window for placing in the package b coz we get file recognization problem.

Example 1) Create a package tp and place a class called Test by choosing suitable features.

```
// Test.java                                   //5
package tp;                                    // ①
public class Test                              // ②
{
    public Test ()                             // ③
    {
        Sop (" Test: Default constructor");
    }
    public void disp ()                        // ④
    {
        Sop (" Test: Disp method");
    }
}
```

2) Create a package tp & place an interface Itest

```
package tp;
public interface Itest
{
    void show();
}
```

* Syntax for compiling package related classes & Interfaces :-

In Real Industry for compiling package classes & Interfaces we use the following syntax

javac -d . filename.java

Explanation:-

1) Here -d is an directory option / switch which gives an indication to java compiler saying that " goto filename.java, take package name & create it as directory / folder" provided following conditions must be satisfied

ⓐ filename.java program should not contain any errors

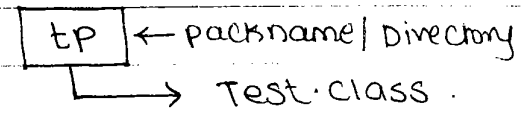
ⓑ Earlier the packagename should not be created as a directory (If it is already created as a directory then it will be considered as currently created directory without recreating it). The directory which is created now is considered as current directory and it is refferred by a symbol dot (.).

2) filename.java prg will be complied and ensure filename.class file will be generated.

3) dot (.) refers current directory. due to dot (.) currently generated class file in step 2 is automatically copied into

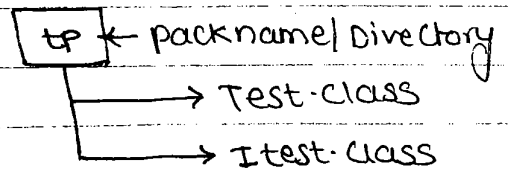
Example ① cmd> javac -d . Test.java.

Ensure the package tp will be created as a Directory, Test.java prg will be compiled, Test.class file will be generated and it will be automatically copied into tp package.



Example ② cmd> javac -d . Itest.java.

Ensure the package tp will not be created and the created directory will be considered as recently created or currently created, Itest.java prg will be compiled, Itest.class file will be generated & it will be automatically copied into tp package



24/8/13

Q. Write a Java program which makes use of Test class of tp package

```
// PackDemo1.java
import tp.Test;
class packDemo1 ~~java~~
{    psvm( String args[])
     {
        Test t1 = new Test()
            t1.disp();
     }
}
```

Q Write a java program which illustrate Itest interface of tp package

```
class swapnil implements tp.Itest   fully qualified name
{
    public void show ()
    {
        sop (" Show(): defined swapnil");
    }
}

class PackDemo02
{   psvm (String args[])
    {   sop ("wrt swapnil: POEL");
        swapnil so = new swapnil();
            so.show();
        sop(" wrt Itest: DB");

        tp.Itest io= new swapnil(); // Indirect approach
            io.show();
    }
}
```

* Number of approaches for referring classes & Interfaces of pkg

In Java programming we have two approaches for referring the classes & interfaces of a package they are,

1. By using import statement.
2. By using fully qualified name approach

① By using import statement :-

import is a keyword used for reffering either all classes & interfaces of a specific package or a specific class or specific interface of a specific package in the current Java prg.

An import keyword can be used in varities of ways

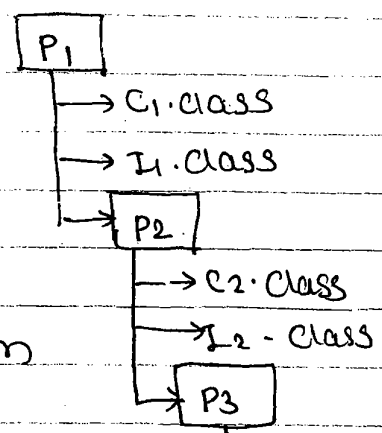
Approach 1:- This approach makes us to understand how to refer all classes & interfaces of a specific package

Syntax :-

import pack1[.pack2[.....pack n]].*;

Here * is a wild character which makes the Java compiler to verify all the classes & interfaces at compile time & makes the JVM to bring all classes and interfaces at the runtime.

① import P1.*;
② import P1.P2.*;
③ import P1.P2.P3.*;



If we use stmt ① as a part of our Java program then we can refer all the classes & interfaces of package P1 but we can not refer all the classes & interfaces of its sub packages P2 & P3.

When we use stmt2 as a part of our java program then we can refer all the classes & Interfaces of package P2 but not classes & interfaces of its upper package P1 & lower package P3.

Approach 2:- This approach makes us to understand how to make use of a specific class or specific interface of a package

Syntax:-

import pack1[.pack2[..... [.packn]]].<Isname>;

① import P1.C1;
② import P1.P2.C2;
③ import P1.P2.P3.I3;

If we use stmt① as a part of our java prg then we can refer only the class C1 of the package P1 but not other classes & interfaces of same package.

Static import Statements:-

static import is the new facility added in jdk 1.5 version onwards.

The benefit of static imports statement is to eliminate redundant refering of classnames and interface names before the static datamembers & ~~class~~ eliminating classnames before static methods.

Static import statement must be always supplied before static datamembers & static methods but not for instance features.

Approach 3:- This approach makes us to understand how to refer the static data members in our java program without using classnames & Interface names.

Syntax:-

import static pack1[.pack2.....[.packn]].class/Interfacename.staticdatamembernamen;

e.g:

```
import static java.lang.System.out;
    :
  out.println("helloworld");
```

Approach 4:- This approach deals with how to refer the static methods directly without using classname

Syntax:-

import static pack1[.pack2.....[.packn]].classname.staticmethodname;

e.g:-

```
import static java.lang.Math.pow;
    :
  SOP(" 2 to power 3 = " + pow(2,3));
```

Approach 5:- This approach makes us to understand how to refer all the static features of a class or interface in our java program without preceded by classname | interface name

Syntax:-

```
import static pack1[.pack2[.....[.packn]]].
                    Classname | Interfacename.*;
```

eg:

```
import java.lang.Math.*;
import static java.lang.System.out;
Class MathDemo
{
    psvm (String args[])
    {
        out.println ("val of PI=" + PI);
        out.println ("val of e =" + e);
        out.println ("2 to power 3 =" + pow (2,3));
    }
}
```

26/8/13

* Fully qualified Name Approach:-

This approach is also known as canonical form. It is one of the alternative approach for referring the classes & Interfaces instead of import statement.

With this approach one can refer either a class or interface but not possible to refer all the classes & interfaces of the package.

If we refer any class or interface throughout our java prg multiple times then in all the times we need to refer that classname or interface name with its fully qualified name.

syntax:-

pack1 [.pack2[.....[.packn]]].clsname / Intf name

example:-

① P1.C1 o1 = new P1.C1();

② class Test extends P1.P2.C2 fully qualified names.
{ .....
.....
}

③ Class Test2 ~~extends~~ implements P1.P2.P3.I3

Q. What is the difference between package keyword and import statement?

package is the keyword used for creating the user defined packages & placing common classes & interfaces.

import is a keyword used for reffering the classes & interfaces of a specific package.

* Access Specifiers :-

Access modifiers are those which are applied before the class def^ns & interface def^ns whereas access specifiers are those which are applied before the data members and methods.

In Java programming we have 4 types of Access specifiers they are,

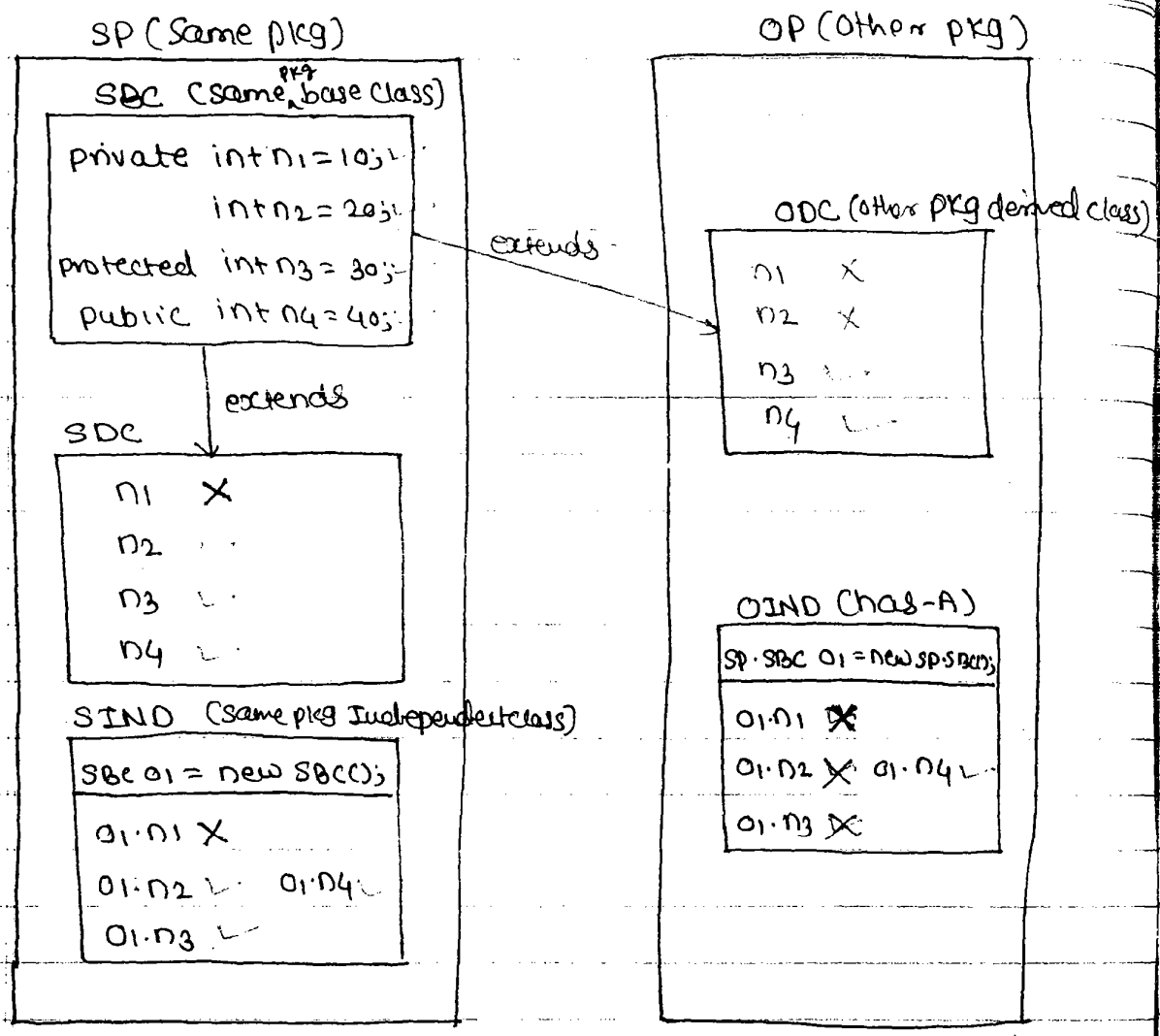
1. private
2. Default (not a keyword)
3. protected
4. public.

If we are not writing either private or protected or public then JVM is by defatult treated as 'default' access specifier.

The purpose of Access specifiers is how to Access the features within & across the packages. bet^n Class to class, interfaces to interface & interfaces to class. (or)
Access specifiers always makes us to understand where to access the features & where not to access the features (or) Access specifiers always provides features controlling accessing mechanism

Rules for Access specifiers:-

following diagram gives Rules for Access specifiers.



Access specifier protection Matrix:-

category of classes / cat. of ASP.	same package BC	same package DC	same package INDC	Different package DC	Different package INDC
private	✓	✗	✗	✗	✗
Default	✓	✓	✓	✗	✗
protected	✓	✓	✓	✓	✗
public	✓	✓	✓	✓	✓

✓: Accessible / Visible
✗: not accessible / Invisible.

27/8/13

Q. Write a Java program which illustrate the rules for access specifiers.

```
package sp;
public class sbc
{
    private int n1 = 10;
            int n2 = 20;
    protected int n3 = 30;
    public int n4 = 40;
    public sbc ()
    {
        sop ("val of n1 = " + n1);
        sop ("val of n2 = " + n2);
        sop ("val of n3 = " + n3);
        sop ("val of n4 = " + n4);
    }
}
```

sp → sbc.class

cmd > javac -d . sbc.java

```
package sp;
public class sdc extends sbc
{
    public sdc
    {   // sop ("val of n1 " + n1)
        sop ("val of n2 = " + n2);
        sop ("val of n3 = " + n3);
        sop ("val of n4 = " + n4);
    }
}
```

is-A relationship within the package

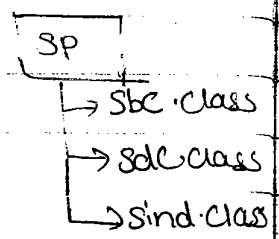
not accessible

sp → sbc.class
→ sdc.class

sind.java

```
package sp;
public class sind
{
    sbc s1 = new sbc();
    public sind ()
    {
        sop(" val of n2 = " + s1.n2);
        sop (" val of n3 = " + s1.n3);
        sop (" val of n4 = " + s1.n4);
    }
}
```

cmd> javac sind.java



spdemo.java

This program makes use of sbc, sdc & sind of sp package.

```
class spdemo
{
    psvm (-)
    {   sop(" wrt sbc");
        sp.sbc s1 = new sp.sbc();
        sop(" wrt sdc");
        sp.sdc s2 = new sp.sdc();
        sop (" wrt sind");
        sp.sind s3 = new sp.sind();
    }
}
```

cmd> javac spdemo.java

cmd> java spdemo.

```
package op;
public class odc extends sbc
{
  public odc()
  {
      sop
      sop ("val of n3 = " +n3);
      sop ("val of n4 = " +n4);
  }
}
```

op
→ odc.class

```
// oind.java
package op;
public class oind
{
    sp.sbc s1 = new sp.sbc();   // has A relationship
    public oind()
    {   // sop ("val of n1 = " + s1.n1);
        // sop ("val of n2 = " + s1.n2);
        // sop ("val of n3 = " + s1.n3);
        ~~sop ("val of n~~
        sop ("val of n4 = " + s1.n4);
    }
}
```

op
→ odc.class
→ oind.class

cmd> javac -d . oind.java

```
// opdemo.java
// This program [illegible] used to find of [illegible]
import op.odc;
import op.oind;
class opdemo
{
    psvm (-)
    {
        Sop (" wrt odc");
        odc o1 = new odc();  // 10 20 30 40
        sop (" wrt oind");
        oind o2 = new oind(); // 10 20 30 40 40
    }
}
```

```
cmd) javac opdemo.java
cmd) java opdemo
```

① private access specifier is also known as Native access specifier.

② Default access specifier is also known as package access specifier.

③ protected access specifier is also known as Inherited access specifier.

④ public access specifier is also known as universal access specifier.

Q. ① Which Access specifier is known as package ASP?
Default.

Q ② What are the package Access specifiers?
Private default (not a keyword) Protected public