

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

Santosh Kumar Mishra

SDE @Microsoft

Core Java (J2SE)

* Software:- Set of programs meant for performing 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 SW are development of drivers (printers, scanners, joystick, etc)

b) Development of language compilers, interpreters, linkers, assemblers, etc)

c) Development of real time operating systems (Windows, XP, UNIX, Linux, Mac OS, AIX, Solaris, etc)

According to industry standards to develop the system SW we use the languages like C and ALP (Assembly language programming).

These languages becomes popular for development of system software because they have high capabilities to interact with hardware devices directly.

- GUI Appⁿ = Look & feel based appⁿ.
- * 256 Header files in C lang.
 - * MNC uses C & ALP developing system slw. for (2)

System software:-

Definition:-

A System slw 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 appⁿ slw is to develop slw for the organizations.

In slw development to develop the appⁿ slw 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 appⁿ are called GUI appⁿ)

Username:	Sai
password:	sai
<input type="button" value="Login"/>	<input type="button" value="Exit"/>

To develop GUI appⁿ 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~~ SAIYAM
Low level lang ⇒ Interpreters used (8086) ⇒ slow.

If we develop any GUI app! by using the above technologies then such GUI app! 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 front end technologies.

⑥ AWT (Abstract windowing toolkit)

Applets → sun microsystems Java
Swings

If we develop any GUI app! 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 permanently (known as data persistancy).

In Industry data persistancy can be achieved in two ways they are

a) Through files.

b) Through database SQL.

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 sw.

so every database sw is treated as backend technology.

The following table gives database sw name and corresponding vendor

Database sw	Vendor
oracle	oracle corp.
MySQL	SUN MICROSYS.
DB2	IBM.
SQL server	microsoft.

Q. Define standalone applications and distributed app's.

Standalone applications:-

A standalone app' is one which runs in the context of local disks and whose results are accessible in the same disk but not accessible across the globe.

e.g: All the projects / app's of systems or app's sw are comes under standalone app's.

Distributed app':-

Are those which are running in the context of browsers / www whose results are accessible across globe.

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

9/7/13
③

Internet Application software:-

Internet sw basic aim is to develop distributed applications.

We know that distributed app's are those which are running in the context of browsers / www & whose results are shareable across the globe.

To develop the distributed app! we have following languages/ Technologies,

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

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

" Hence JAVA is one of the Internet SW developed by SUN microsystems 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.

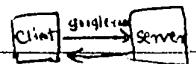
SATYAM	
PAI	VI
DAI	7

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

① 2-tier Architecture:-

This arch. contains 2 types of programs they are,

- ① set of client side programs.
- ② set of server side programs.

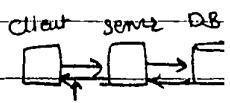


e.g.: Google.com

② 3-tier Architecture:-

This Arch. contains 3 types of programs they are,

- ① set of client side programs.
- ② set of server side programs.
- ③ set of database software.

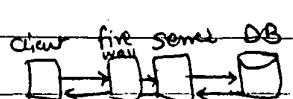


e.g.: mail.yahoo.com, gmail.com, ATM based appn, etc

③ n-tier architecture:-

This arch. contains 4 programs they are

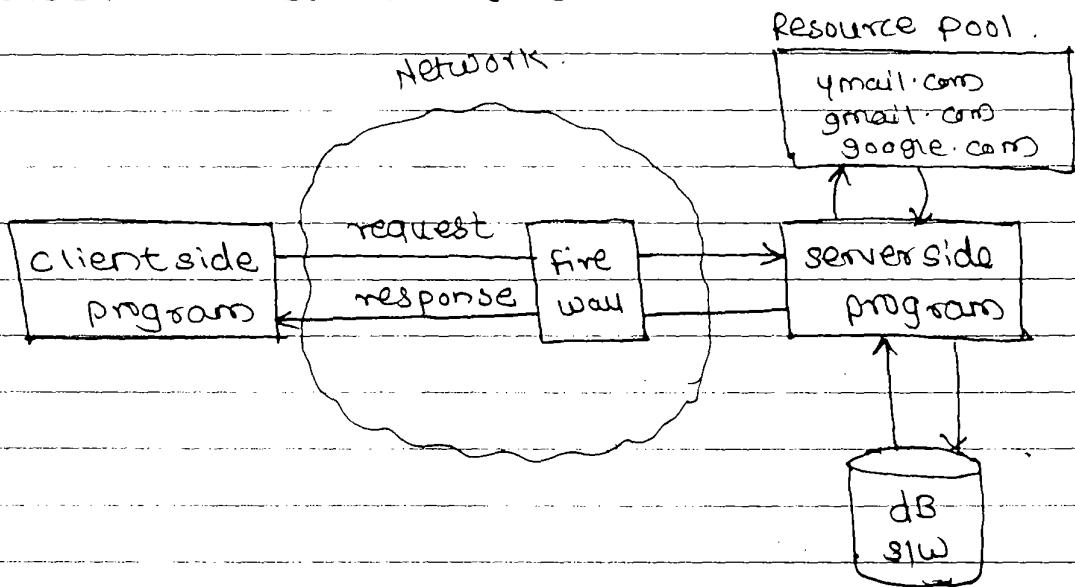
- ① set of client side programs.
- ② set of firewall programs.
- ③ set of server side programs.
- ④ set of database software.



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

- ① set of client side programmers
- ② set of server side programmers
- ③ set of firewall developers
- ④ set of database programmers.

Following diagram gives flow in the client server architecture.



* Definitions:-

- ① **Client**:- A client is a program which is always used for making request to the server side programs 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 I/O processing)

The basic advantage of server slw in the real world is to provide concurrent access.

The following table gives server slw names and their vendors.

Server SW Name	Server SW vendor name
Tomecat	Apache jakarta SW foundation.
Websphere Weblogic	BEA system
Websphere	IBM
Oracle	Oracle Corp.
Glassfish	SUN microsystem
IIS (Internet Information Server)	Microsoft.

⑧ Resource pool:-

Resource pool is one of the layer present in server SW & in which we place distributed applications.

⑨ Firewall:-

Firewall is one of the security programs which will validate whether client request is virus oriented or virus free.

⑩ Database:-

Database is one of the data persistence 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 world 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? development & it becomes more popular for development of distributed applications.

The java slw developed at a sun microsystem under the guidance of James Gosling & others.

The java slw 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 "true name" & OAK has taken 18 months to comp develop.

The slw OAK was popular for developing slw for electronic, Mechanical & automobile devices.

In other words the SW 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 SW 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 var-

The Java SW is available in the real industry in 3 categories They are,

- ① J2SE [Java 2 Standard Edition]
- ② J2EE [Java 2 Enterprise Edition]
- ③ J2ME [Java 2 micro/Mobile Edition]

In the industry J2SE is used for developing client side application, J2EE is used for developing server side app, and J2ME is used for developing mobile/wireless based app.

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

Java Tech. Version	Java Long. 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. (12)

JAVA 5	JDK 1.5 [Tiger SW]
JAVA 6	JDK 1.6
JAVA 7	JDK 1.7

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

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 b/w 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 ^{identity} session for certain time)	stateful protocol. (maintains identity of client forever)
Belongs to TCP	Belongs to UDP

Q. Define stateless & stateful protocol.

Ans:- Stateless protocol:-

A Stateless protocol is one which maintains an identity of a client for the limited period of time.

e.g: HTTP. (Best suited for ATM & logins)

Stateful protocol:-

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

e.g: FTP.

5/F/B

* 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 core compelt

without pointers :- app's development time is less.
app's execution time is more.

14

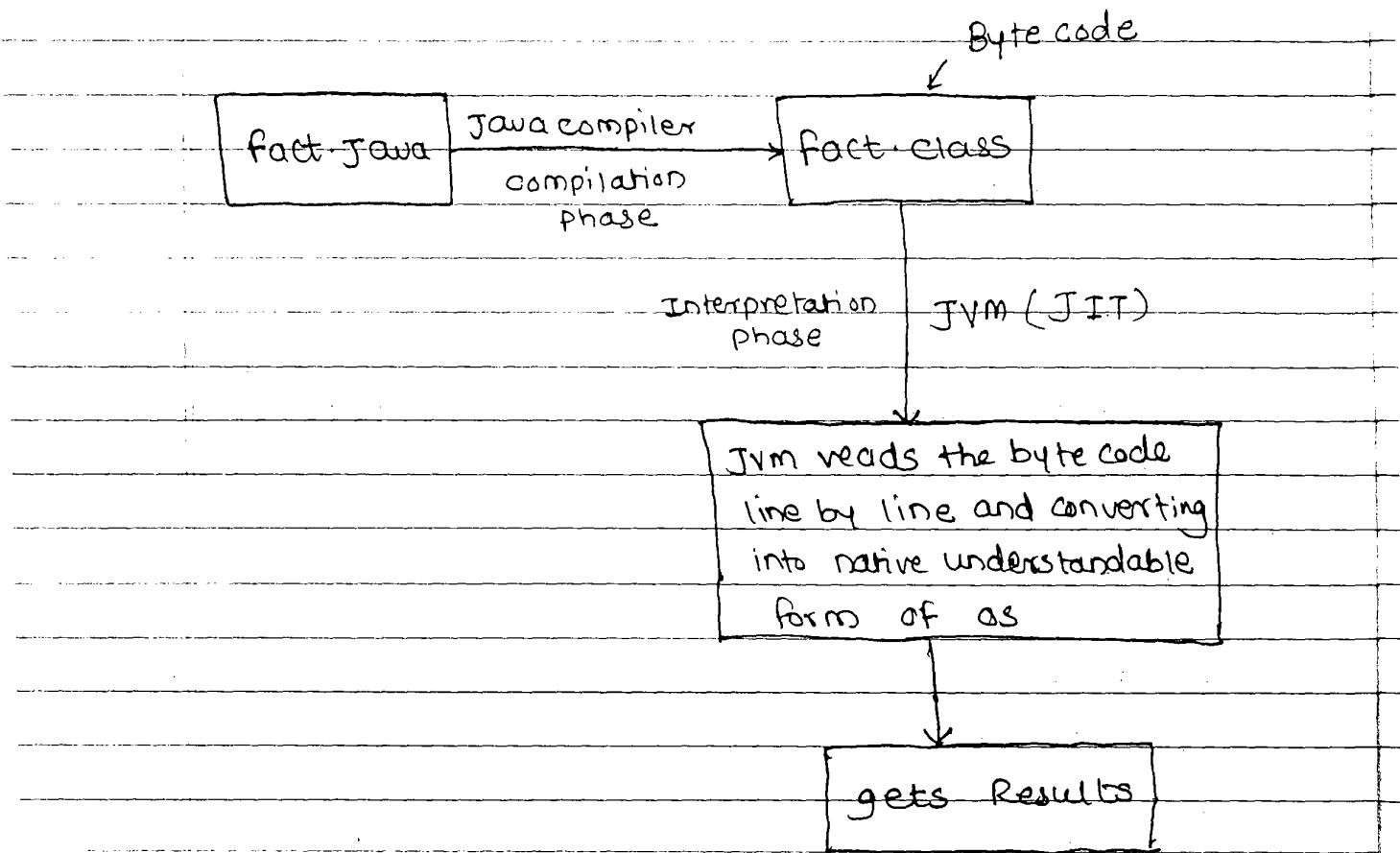
with pointers:- app's development time is more.
app's execution time is less.

① Simple:-

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

~~con~~ Java programming eliminates complex concept called pointers so that app's development time is less and app's execution time is less because of magic of byte code

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



* Definitions:-

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^{copy} & whose role is reading the byte code line by line & converting 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 programs 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 \Rightarrow memory alloc' at runtime
C, C++ \Rightarrow static \Rightarrow allocate memory at compile time

C++ can deallocate unused memory by using destructors.

16

alloc' = allocation

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's background Java programs which is running along with our regular Java programs for collecting unused/unreferenced memory space. and it improves performance of every Java application.

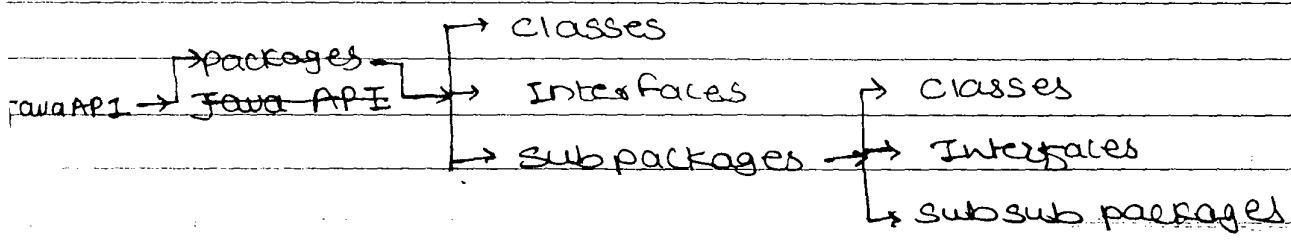
Garbage collector program running internally for regular periodical intervals of time.

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

(Just like header files .inc & .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 case having two times more space than DOS as except char it has same space

PAGE NO.	17
DATUM	, ,

(Date)

(d) Java programming language provides user friendly syntaxes which makes user to develop quicker app's development without errors.

② Platform Independent :-

A lang / Tech. based app's are said to be platform independent if & only if whose app's runs on each and every os 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 OS.

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

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

The language like Java & whose related app's are treated as platform independent because Java satisfies above 2 properties.

8/7/18 (Mon)

② 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 modeling languages because their execution environment provides single flow of control, They provide sequential execution whose appⁿ takes more execution time & does not contain any library for development of multithreading based appⁿ.

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

takes less execution time & provides an effective library for development of thread based appn.

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

(a) 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.

(b) Background / Parent Thread:-

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

The real time implementation of multithreading concept is that to develop real world servers softwares such as Tomcat, weblogic (BEA systems), websphere, etc. In other words most of the real world servers are 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.



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 app's)



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 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 or 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
#

⑥ 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.

Dynamic :-

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

They are,

- static memory allocⁿ
- Dynamic memory allocⁿ

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

Due to static memory allocation we get the following limitations.

- Waste of memory space.
- Loss of Data.
- 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, sun developers has populated that Java is one of the Dynamic programming language.

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 programs on their own & they can use readily available security programs 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

10/9/13



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.

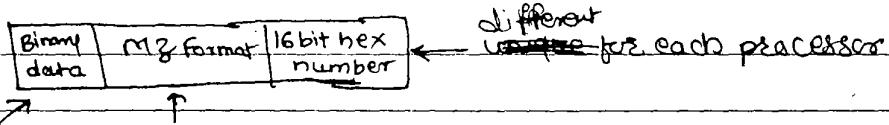
To say a lang / Tech. is arch. neutral, it has to satisfy the following property

i) The lang / Tech. must contains 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:-



The binary MZARTA format which is native understandable data of code 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 app's are treated as non portable. Whereas java lang. & whose app's are said to be portable.

Industry is preferring to use portable language for development of most of the distributed applications & not preferring 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 programs in the main memory of the computer either by following static memory allocation or Dynamic memory alloc.

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.

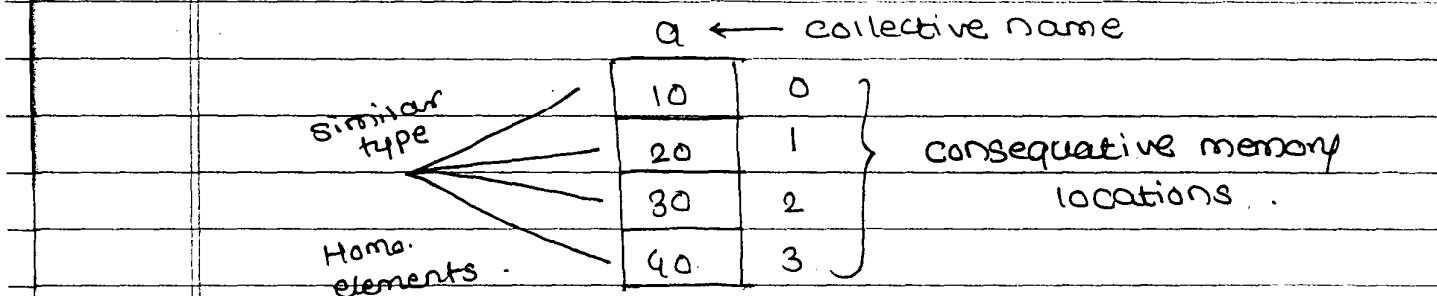
e.g: `int a[] = { 10, 20, 30 };` // valid.

`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 consecutive mem. locations which are all referred 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 structures (Struct), Unions (Union), Enumerations (Enum).

simillarly in 'java' programming we use classes (Class), Interfaces (Interface), Enumerations (Enum), etc. for development of user defined data types.

e.g:

Student so = new Student();

so	10	stro
	swop	sname
	85.5	marks
	'A'	grade

~~12/7/13~~

* Java Fundamental Data Types-

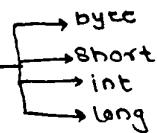
In Java programming we have 8 fundamental data types which are classified into 4 categories they are,

- 1) Integer category data types
- 2) float category data types.
- 3) Characters 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.



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 program 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.

$$\text{Range of any} = \text{data type}$$

The number of bits available in the long which is understandable by computer

number of bits occupied by a particular data type

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

$$16 \text{ bits} = 2 \times 8$$

e.g: Range of
short data type = (2)
16 bits

$$= -1 \text{ to } 65536$$

$$= 0 \text{ to } 65535$$

$$= +32767.5 \text{ to } -32767.5$$

$$= +32767 \text{ to } -32768$$

~~(15) $\neq 13 \pmod{2}$~~

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	$+n \text{ to } -(n+1)$	7
double	8	$+y \text{ to } -(y+1)$	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 $f_1 = 3.4f;$

3.4

f_1
011.0110011

7 decimal places.

for 0.4:-

$$\begin{array}{r} 0.4 \\ \times 2 \\ \hline 0.8 \\ \times 2 \\ \hline 1.6 \\ \times 2 \\ \hline 3.2 \\ \times 2 \\ \hline 0.4 \end{array}$$

for 3:

$$\begin{array}{r} 2 \\ \underline{\times} \\ 2 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 0 \\ \underline{\times} \\ 1 \end{array}$$

$$011$$

$$011.0110$$

Double:-

If we store any real constant value in a variable of double datatype then the double datatype stores the real constant value in such a way 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 d1 = 3.4d;

०८

```
double d1 = 3.4;
```

1

011·011001100110011

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 f1 = 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; IF (a == 3.4) double { pf("OK"); } else { pf("CANCEL"); } pf("OVER");	② float a = 3.5; IF (a == 3.5) double { pf("OK"); } else { pf("CANCEL"); } pf("OVER");	This happens due to floating point calculation. 3.4 :- 011.010011 float 011.011001100110011 double which are not equal.
ANS: CANCEL OVER	ANS: OK OVER		which are equal.

ANS: CANCEL OVER

ANS: OK OVER

To overcome this problem we need to write f for

⑧ Character category Data Type:-

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

A collection / sequence of characters enclosed within double quotes 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 programming we use a datatype called 'char'

Char data type in C, CPP takes 1 byte becoz C, CPP follows ASCII character set.

In Java programming char datatype takes 2 bytes becoz 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 bytes ($2^8 = 256$)
1 byte.

Languages like C, CPP, CAROL, 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 ($2^{16} = 65536$)
e.g: Java, .Net, etc. 2 bytes.

* NOTE:- (Drawback of Java)

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

① 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 programmers 1 byte of memory space gets wasted which is not collected by garbage collector or it is simple error in memory.

~~16/07/13~~

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

SARITA

86

* 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.

* Definition of Variable:-

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

Hence to write a meaningful programs 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 numbers 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 v1, v2, ..., vn;`

datatype represents either fundamental, derived or user defined datatype. v₁, v₂, ..., v_n represents java valid variable names.

Example:-

<code>int a, b, c;</code> <code>a = 10;</code> <code>b = 20;</code> <code>c = a+b;</code>	a 10 20 30	<code>- OR -</code> <code>int a;</code> <code>a = 10;</code> <code>int b;</code> <code>b = 20;</code> <code>int c;</code> <code>c = a+b;</code>	a 10 b 20 c 30
--	---	---	---

* 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 v1=val1, v2=val2, ..., vn=valn;`

Here val₁, val₂, ..., val_n represents list of values decided to initialize in the variables v₁, ..., 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.

^{common}
In general every fixed variable value must be made as final.

Syntax:- `final datatype v1=val1, v2=val2, ..., vn=valn;`

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 Hungarian notation)

e.g: itemStateChanged ()

actionPerformed ()

textValueChanged ()

} Java follows Hungarian notations as shown.

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

Block of stmts represents set of executable statement 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 SW development if we develop any method which is common for most of the Java programmers then such common methods definitions 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 overridden.

In other words final methods cannot be overridden.

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

final float simpleInterest (float P, float T, float R)

```
{  
    float si = P * T * R / 100;  
    return (si);
```

}

(Hence as final is having no one 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 definition 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 SW 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 retrieval language) etc.

In Industry procedure oriented prog. languages are always used for developing applications / projects like system SWs and application SWs. 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 app! (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 Pascal, Object Pascal, C++, Java, .Net, Riffil, Simula, etc

All the above OOPS principles are common for all object oriented programming languages but whose implementation or definitions 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 / programmers defined data types.

While we are using classes concept for development of user defined datatype we use a keyword ~~ext~~ 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 / programmers defined datatype.

Definitions:-

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

OR:

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

SATYAM

46

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 contains 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.

Data members 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 definition

}

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 definition 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 definition of the class must be incorporated within { and } and the definition 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 definition 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 ("Distinction");

}

} // Student.

Q. Write a class definition 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);

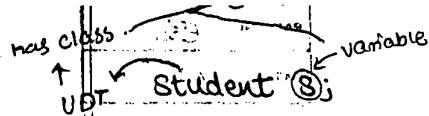
}

} // 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 data members of class by creating its object .



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

Definitions of Object :-

- ① Instance of a class is known as an object (instance is nothing but allocating sufficient amount of memory space for data members & methods of a class)
- ② Each & Every class variable is known as an object.
- ③ Each and Every Grouped items is known as an object.
- 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 data members and methods of a class.
- ② It takes an address / Reference 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 :-

`<classname> objname = new <classname()>;`

e.g: `Student so = new Student();`

o	sno
null	sname
o.o	m ₁
o.o	m ₂
o.o	m ₃

Syntax 2 :-

`<classname> Objname;`

`Objname = new <classname()>;`

Here `stmt1` represents object declaration whenever an object is declared, memory space is

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

Student s0; // student object declaration.

s0

null.

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 reference / Address of loaded class)

s0 = new Student(); // student object

referencing memory space

not null	s0	0	sno
reference of loaded class	xxxx	null	sname
		0.0	m ₁
		0.0	m ₂
		0.0	m ₃

Q. Define a class loader Subsystem.

It is one of the programs developed by SUN microsystems & 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).

13/07/13

Main memory

- (1) Heap memory - Objects
- (2) Stack memory - program logic
- (3) Associative memory - constant values.

S4

Q. What are the differences between classes & objects?

Class

A class is a collection of data members & methods

When we define a class no memory space is created for the data members & methods

All the classes will contain logical existence

The class definitions will exists only once

When we execute a java program, the definition of class loaded once in main memory from secondary memory

Instance of class is known as object.

When we create an object memory space will be created for data members & methods of a class.

All the objects will contain physical existence

w.r.t one class definition one can create multiple objects.

After loading class defⁿ 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

Instance data members are those whose memory space is created each & every time whenever an object is created.

Static Data Member

Static data members are those whose memory space is created only once when the class is loaded in main memory irrespective 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 v₁, v₂, ..., v_n; Syntax: static datatype v₁, ..., 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 members
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 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

Instance methods are those which are recommended to perform repeated operations such as reading records from file, database, user authentication process, etc.

Instance methods meant for performing specific operations .

Programmatically Instance methods definition should not be preceded by static

Static Methods

Static methods are those which are recommended to perform one time operations like opening files in read or write mode, obtaining database connection, etc.

Static methods are meant for performing common operations .

Static methods defn must be preceded by static .

Syntax:-

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

③ : Sop (" val of a "+a);

④ : 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 <ClassName>

{

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 in turn 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 defined datatype.
- ③ <classname> represents Java valid variable name treated as name of class. Each & every classname in Java programmatically treated as user defined datatype.
- ④ Data members 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 convention 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".

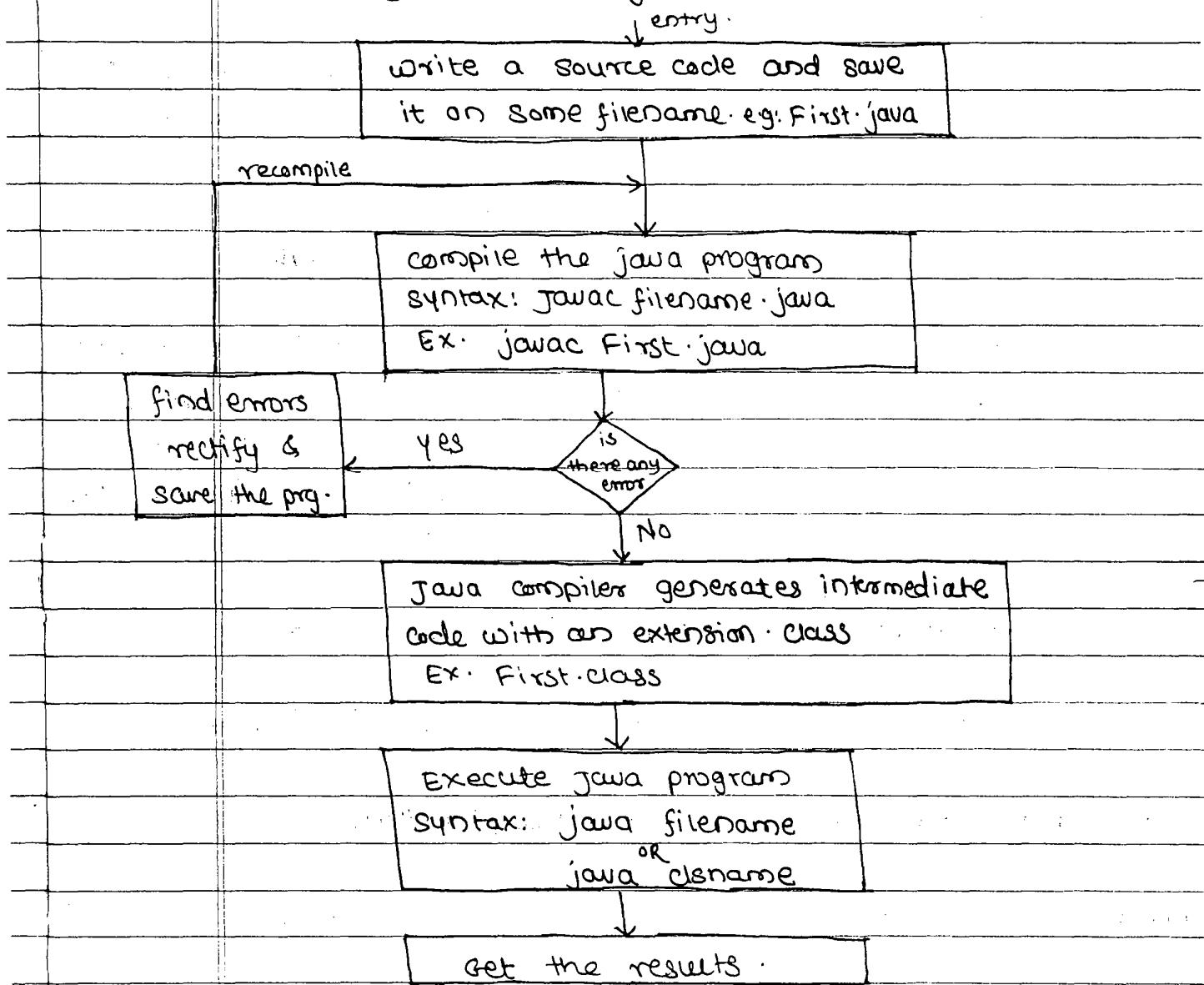
// First.java.

```
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 app? we require 3 requirements.

1) Java software requirement: Download jdk 1.5, jdk 1.6, 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 Environment)

IDE is one of the third party SW developed by third party vendors and released to the real industry for the development of Java applications by getting inbuilt compatibilities.

Various IDEs are,

I] Notepad, Dos Editor, Wordpad, Notepad++

II] EditPlus (developed by ES Enterprise)

III] NetBeans, Eclipse, MyEclipse, IntelliJ IDEA, 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 with 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

if (f1() == f2()) {
 f3(); // Rule ②

public static void main(String[] args) {
 f1();
 f2();
}

UWite kettu(70) sop("fz()<br") section 3.12 2013 2013

problems Questions of numbered 333 pages

6 80 0800 86728.00 091555 4901 182000

void f3()

201 200 Open Third (52%) as 11. Rule (4) 165.17

```
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 whether they belongs to same class or different class.
- ④ One instance method can call another static method wrt classname & without considering whether 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

{

PSUMC String s[])

{ SOP ("Starting of main");

sum so = new sum();

so.accept();

so.add();

so.~~disp~~.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 programs satisfies the above properties then such type of programming is called static programming / hard coded approach. Industry is not recommended to develop hard coded 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]);
 }



Accepting the values Dynamically through Command prompt :-

The numbers 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: E:\Swap\Prgs> java SumDemo 10 20

Command prompt

java prg

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])

10	0
20	1

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 wst 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 microsystems developers has withdrawn the decision of providing 8 main methods wst 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
```

```
{
```

```
    psum( String s[] )
```

```
{
```

```
    System.out.println("No of cmd line args: " + s.length);
```

```
    for (int i=0; i < s.length; i++)
```

```
{
```

```
    System.out.println(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.

2x10713

* 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.

2d

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)
byte	<u>Byte</u>	public static byte parseByte (String)
short	<u>Short</u>	public static short parseShort (String)
int	<u>Integer</u>	public static int parseInt (String)
long	<u>Long</u>	public static long parseLong (String)
float	<u>Float</u>	public static float parseFloat (String)
double	<u>Double</u>	public static double parseDouble (String)
char	<u>Character</u>	<u>parse</u>
boolean	<u>Boolean</u>	public static boolean parseBoolean (String)

3

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

↓ wrapper class name

public static xxx parseXxx (String)

4

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 parameters 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"};

10	20
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.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 \Rightarrow Business logic class.

class SwapDemo

{

psvm (String s[])

{ if (s.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(" original values:");
so.print 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));
        }
    }
} // mul BLC

```

```

class MulDemo {
    public void [sum] (String s[])
    {
        if (s.length != 1)
            SOP (" enter one value");
    }
}

```

every class name in java is a programmer defined datatype.

77

else

{

int n = integers.parseInt(SC0);

if (n <= 0)

SOP ("n + " Invalid input try Again");

else

{ mul m0 = new Mul();

m0.set(n);

m0.table();

}

}

} // main

} // MulDemo \Rightarrow ELC

29/7/13
mon.

Q Write a Java program which will decide whether 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(i==n)
        return "PRIME";
    else
        return "NOT PRIME";
}

```

} // prime BLC

```

class PrimeDemo
{
    public static void main(String args[])
    {
        if(args.length != 1)
            System.out.println("Enter only one value");
        else
        {
            int x = Integer.parseInt(args[0]);
            if(x<2)
                System.out.println("x + " is invalid input try again");
            else
            {
                Prime p0 = new Prime();
                p0.set(x);
                p0.display();
            }
        }
    }
}

```

Q. goto Stmt is not there in Java.

79

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 xc)

{

n = xc;

}

void convert()

{

if (n <= 0)

System.out.println("No Roman equivalent");

else

{

while (n >= 1000)

{ System.out.println("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. \Rightarrow BLC.

Class RomanDemo

```

{ psum (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 numbers to ordinary number.

ANS:

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

MMX 0

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

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

* Constructors In Java :-

Definition:-

A constructor is a special member method which will be automatically called by JVM (implicitly) 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 appn 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 <classname>
{
    ---  

    ---  

    < classname () > // Default constructor  

    {  

        Block of stmts; // Initialization  

    }  

    ---  

    }
}

```

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

```
class Test
```

```
{
```

```
int a,b;
```

```
Test ()
```

```
{
```

```
System.out.println("Test- Default constructor");
```

```
a = 10;
```

```
b = 20;
```

```
System.out.println("val of a = " + a);
```

```
System.out.println("val of b = " + b);
```

```
}
```

```
} // Test
```

```
class TestDemo
```

```
{
```

```
public void main (String s[])
```

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

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

```
}
```

t1

10	a
20	b

t2

10	a
20	b

Rules:-

① Defining the default constructor is optional.

If we create an object of a specified class w/o 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 w/o same class for placing different values.

Syntax:-

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

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

```

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

```

```

class TestDemo
{
    public static void main (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 Constructors:-**

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:-

Definition:- A constructor is said to be object parameterized if 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 resembling the functionality of copy constructor of C++ (copy constructors are not present in Java)

Example:-

Test t1 = new Test (10, 20);

t1	
10	a
20	b

Test t2 = new Test (t1);

x	
10	a
20	b

Test (Test x)

{

a = x.a;

b = x.b; "

System.out.println("a=" + a);

System.out.println("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 ()

{

System.out.println("Test Default constructor");

a=1;

b=2;

System.out.println("val of a = " + a);

System.out.println("val of b = " + b);

}

Test (int x)

{

System.out.println("Test: single parameterized constructor");

a=x;

b=x;

System.out.println("val of a = " + a);

System.out.println("val of b = " + b);

}

Test (int x, int y)

{

System.out.println("Double parameterized constructor");

a=x;

b=y;

System.out.println("val of a = " + a);

System.out.println("val of b = " + b);

}

Test (Test x)

```

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

```

class TestDemo

```

{
    public void psum (String s[])
    {
        Test t1 = new Test ();
        Test t2 = new Test (1000);
        Test t3 = new Test (10, 20);
        Test t4 = new Test (t2);
    }
}
// TestDemo.

```

Observations:-

- 1) Each and every class of java can contain two types of constructors they are
- single Default constructor.
 - multiple parameterized constructor (known as overloaded constructors)
- 2) By default 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 "javap". It is one of the tool / app" program or exe file present in Java-Home/bin directory.

→ This can be applied after compiling the java program

Syntax:- javap <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) \Rightarrow False

Every class of java contains constructors but developer / programmers of class made those constructors as private.

s)

* 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 b/w 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.currentclass.datamembername`

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;` } Here Assigning formal parameter
`this.b = b;` } values to data members of class.

Here incrementing values
of data members of class

Here we are incre-
menting values of
formal parameters

{ a = a+2;
b = b+2;

SOP (" value of a (formal param)= "+a); }

Here referring
Here a & b refers
formal parameters

```

void disp()
{
    System.out.println("val of a = " + this.a);
    System.out.println("val of b = " + this.b);
}

```

Here writing 'this'
is optional

```

class TestDemo
{
    public static void main(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.set(4); fo.factorial(); fo.disp();
long t2 = System.currentTimeMillis();
System.out.println("Execution time = " + (t2 - t1));

```

(3) Inheritance :-

It is one of the distinguished feature of OOPS.

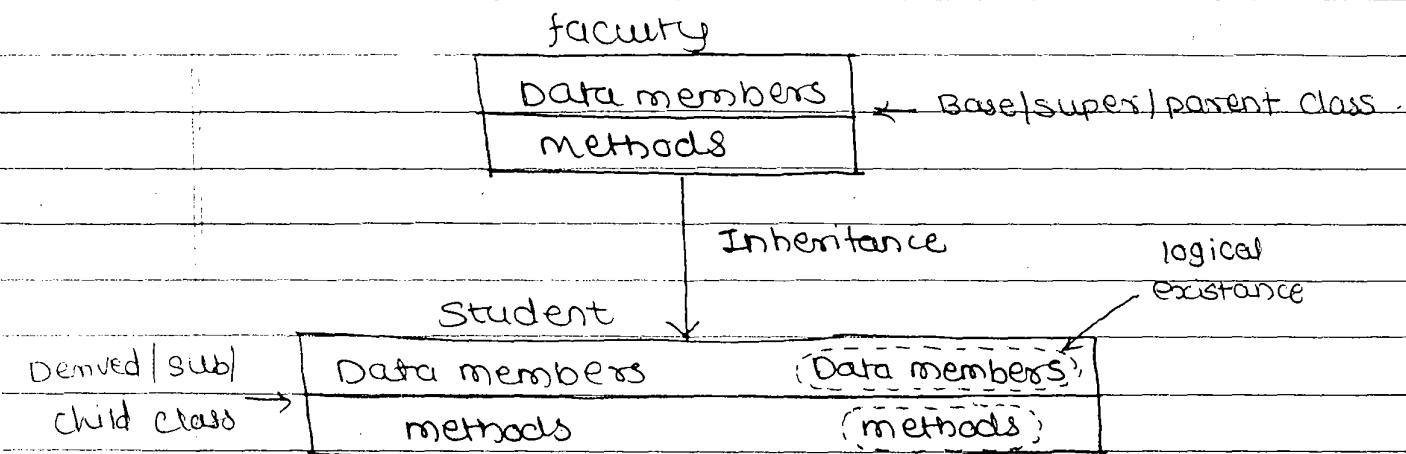
Definition:- The process of obtaining data members and methods from one class to another class is known as Inheritance.

The class which is giving data members & 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 app' with the concept of Inheritance then to that app' Inheritance will provide following advantages,

- ① App' development time is less.
- ② App' memory space (main memory) is less.
- ③ App' execution time is less.
- ④ App' performance is enhanced (Improved).
- ⑤ Redundancy (Repetition) 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)

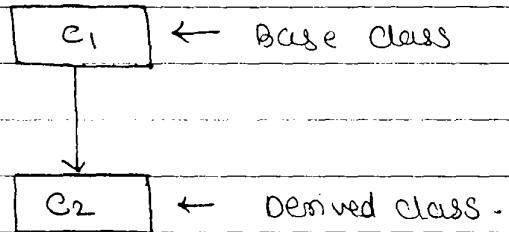
Inheritance Types / Reusable Techniques:-

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

① Single Inheritance:-

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

Diagram:-

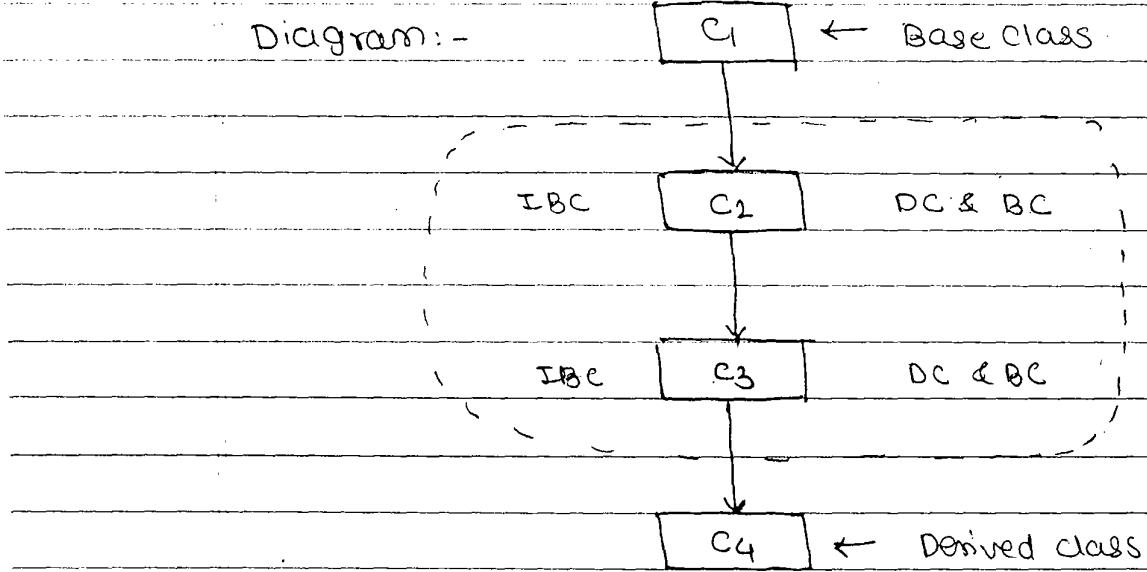


Inheritance path:- $C_1 \rightarrow C_2$

② Multilevel Inheritance :-

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

Diagram:-

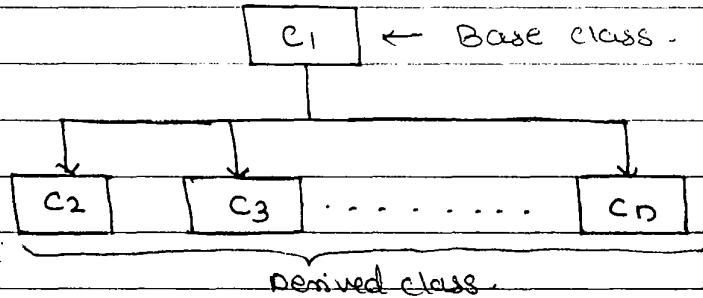


$C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$

③ Hierarchical Inheritance:-

Defⁿ: In hierarchical Inheritance there exists single base class & multiple derived classes.

Diagram:-

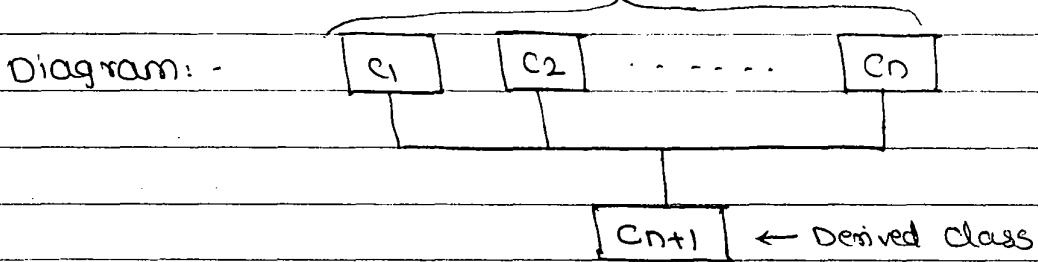


Inheritance path : $c_1 \rightarrow c_2, c_1 \rightarrow c_3, \dots, c_1 \rightarrow c_n$.

④ Multiple Inheritance :-

Def:- In multiple inheritance there exists multiple Base classes & single derived class.

base classes & single derived class



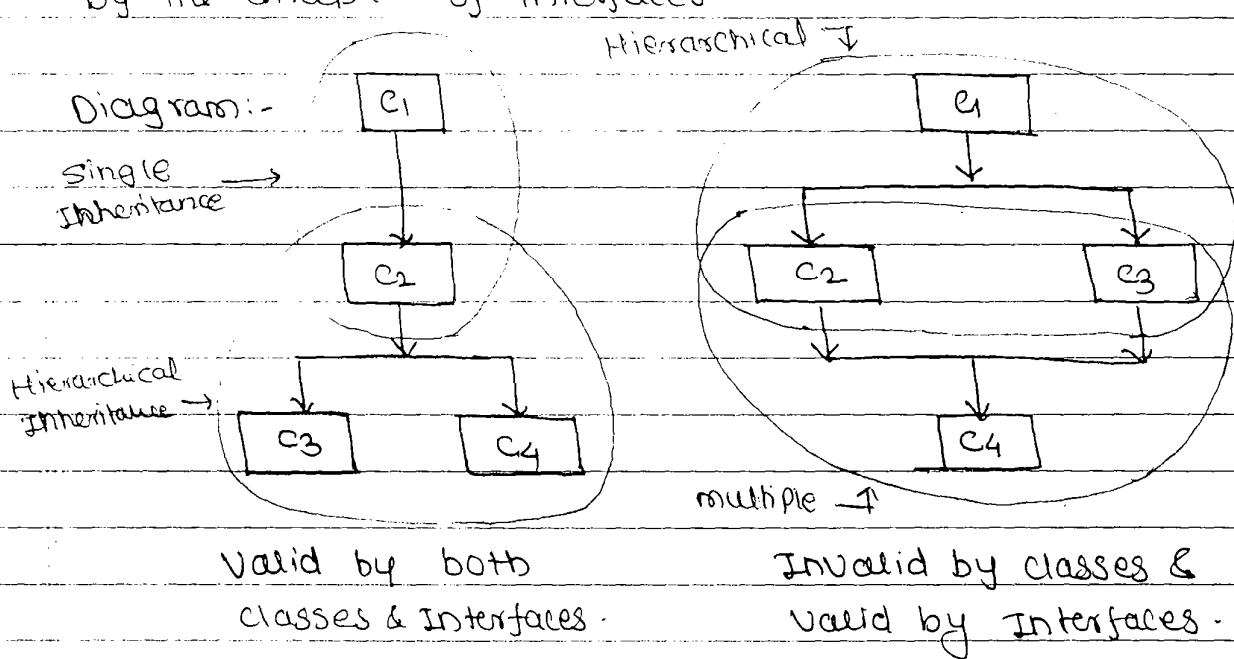
Inheritance path:- $C_1 \rightarrow C_{n+1}$, $C_2 \rightarrow C_{n+1}$, ..., $C_n \rightarrow C_{n+1}$.

NOTE:- Java programming does not support multiple inheritance through the concept of classes but it can be supported by the concept of interfaces.

⑤ Hybrid Inheritance:-

Def? :- 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.



8/8/13

* Inheriting features of Base class into Derived class:-

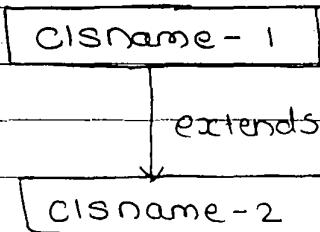
Syntax:- We use the following syntax for Inheriting features of Base class into Derived class.

Class <classname-2> extends <classname-1>

{

variable declaration;
method declaration;

3



Explanation:-

- 1) <classname-1> & <classname-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 through the concept of classes but it can be supported through concept of interfaces.
- 4) Whenever we develop any inheritance app!, it is always recommended to create an object of bottommost derived class because it is inheriting features of topmost base class & intermediate base class
- 5) When we create an object of Bottommost derived class, first we get memory space for data members of topmost base class, second we get memory space for data members 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 inherited but constructors of base class can not be inherited 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 scope 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()

{

 System.out.println("disp: Base class");

}

} // Base class

Class DerivedClass extends BaseClass

{

 int b;

 void show()

{

 System.out.println("show: Derived class");

 a = 100;

 b = 200;

 System.out.println("value of a from BaseClass = " + a);

 System.out.println("value of b from DerivedClass = " + b);

}

} // Derived class

class Indemo

{

 public static void main(String args[])

 {

 System.out.println("WRT Derived class");

 DC doi = new DC();

 doi.disp(); doi.show();

 System.out.println("WRT Base class");

 BC boi = new BC();

 }

}

(mon)
5/8/13

PAGE NO. 104

* 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 betw 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 memory space 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 memory space 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 C2 class will be executing object of C1 will be live & when disp() method completes its execution an object of C1 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.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 another predefined class called System. The relationship betw 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 ambiguity.

In order to differentiate betⁿ base class data members & derived class data members, In the context of derived class the base class data member 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
```

```
{
```

```
    public (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 similar to derived class methods & JVM gets an ambiguity ..

To differentiate b/w 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 method name 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 infinitely 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()

{

System.out.println("Base class : disp()");

}

}

class DC extends BC

{

void disp()

{

System.out.println(" Derived class: disp()");

super.disp(); // call super class disp() from the context of derived class.

}

}

class Sdemo

{

public static void main(String args[])

{

DC doz = new DC();

doz.disp(); // will call derived class disp().

}

}

Super at Method overridden level :-

We know that method overriding 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 doi = new DC();

doi.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>

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 b/w immediate base & derived classes but not between bottommost derived

920 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 app' 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 app' development, constructors are calling from bottom to top. & execution is from top to bottom.

In order to establish communication bet' 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.

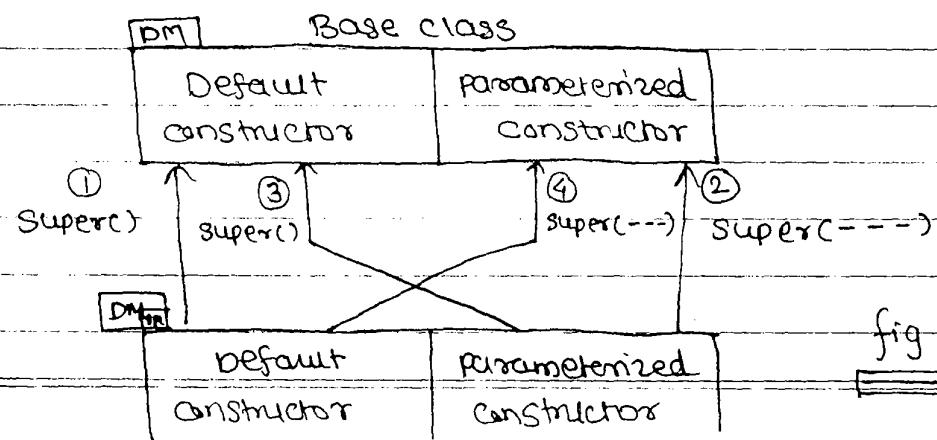
`Super(...)` :- It is used for calling super class parameterized constructor from derived class Constructors.

Rule 2-

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(...)`



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 parameterized 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

 System.out.println(" BC => Default Const");

 a=10;

 System.out.println(" val of a from BC :" + a);

}

}

BC

DM

BC()

super()
①

extends
Rule ①

DM

DC()

DC

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

{

psum(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 aa) ~~int a~~

{

sop(" BC => default const");

this.a = a;

sop(" val of a from BC: " + this.a);

}

}

BC

PM

BC --->

super(-->

extends
rule ②

DM

DC C --->

DC

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)

 { System.out.println("DC: Default param const");

 this.b = x;

}

}

class SDemo

{ public static void main (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)

 { System.out.println("BC: Default param const");

 this.a = a;

 System.out.println("val of a from BC: "+a);

}

}

class DC extends BC

{

 int b; ~~s~~; c = 20;

```

DC()
super();
{ SOP (" DC: Default const");
  b = 30;
  SOP (" val of b from DC" + 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 overriding concept.

We know that method overriding 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,

10/

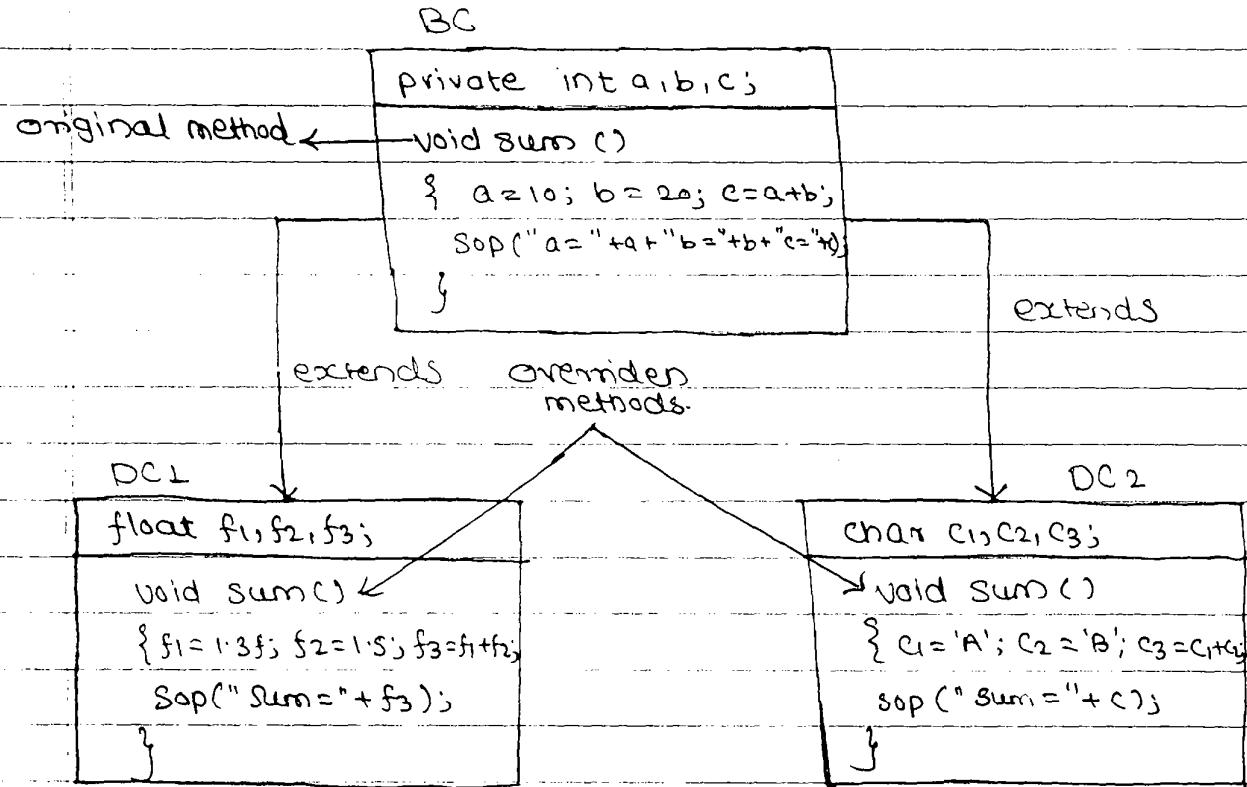


fig ②

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 overriding.

(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 overriding plus Dynamic Binding are,

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

Definition:- 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 client contains

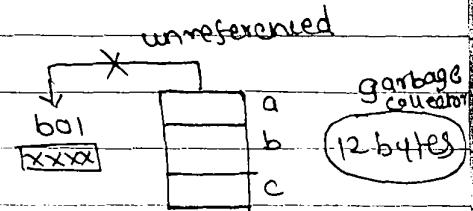
In other words If we create an object indirectly then JVM will consider above two points internally.

NOTE:- ① In order to create an object indirectly then there must exists super-sub relationship / base-derived relationship / master-detailed relationship

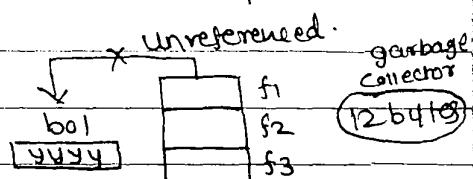
② If we create an object directly C Test t1 = new Test(); then such object creation is known as plain old execution logic & it is not necessary to have super-sub relationship.

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.

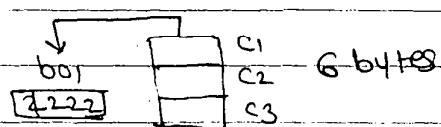
BC bo1 = new BC();
bo1.sum();



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



DC2 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 Polymorphic object.

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 `b1.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 appⁿ 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, & execution logic developed with (polymorphism + method overriding logic)
plain old execution logic (Direct object creation)

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

- ① Appⁿ takes more memory space.
- ② Appⁿ execution time is more.
- ③ Appⁿ performance is reduced.

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

If we develop any Java application by using Case III then Java appⁿ 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.

124

Hence Industry is always recommended to use Case III for developing any Java app? 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
 classes → abstract classes

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

(mon)
2/8/13

* Abstract classes:-

We know that each & every Java program must start with a concept of class. In other words 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:-

Definition:- A concrete class is one which contains 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 \Rightarrow idea about method.

126

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 appⁿ will get following limitations,

- ① Appⁿ will take more memory space.
- ② Appⁿ execution time is more.
- ③ Appⁿ 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:-

Definition:- 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 definition/ body.

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

Syntax for Abstract method :-

abstract returntype methodname (list of formal parameters 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 definition must be made as abstract by using abstract keyword.

Syntax for Abstract class :-

Abstract class <classname>

{

.....

.....

Abstract ReturnType methodname (List of formal
parameters 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 cannot
create Instantiate its object directly But it can be
created indirectly.

Advantages of Abstract classes:-

If we develop any java appn with the concept of abstract classes then such appn's will get following advantages.

- ① Appn memory space is less (main memory)
- ② Appn execution time is less
- ③ Appn performance is more

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

abstract class Shape

{

abstract void area();

}

class square extends shape

{

int a;

void area()

{

a=10;

int area = a*a;

sop(" Area of square = " + area);

}

} // square

class Rect extends shape

{

int b, l;

l=10;

b=20;

```
int area = l*b;
System.out.println("Area of Rect = " + area);
```

```
}
```

```
class TestArea
```

```
{
```

```
public static void main (String args[])
{
    Shape s1 = new Shape(); //error coz shape is abstract
    Shape s1 = new Square(); //is abstract
    s1.area();
    s1 = new Rect();
    s1.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 definition should not belongs to final hence Java programming does not 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 overriding 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 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 a01; // valid Abstract class Declaration

*a01 = new AC(); // Invalid coz direct reference
of AC

*a01 = 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;  
    System.out.println("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
```

```
{
```

```
PSum (String args[])
{
```

```
Sum s01 = new ISum (Integer.parseInt(args[0])
    s01.add(); Integer.parseInt(args[1]));
s01 = new Fsum (Float.parseFloat(args[2]),
    float.parseFloat(args[3]));
s01.add();
```

```
S01 = new Ssum(args[4], args[5]);  
S01.add();
```

14/8/13

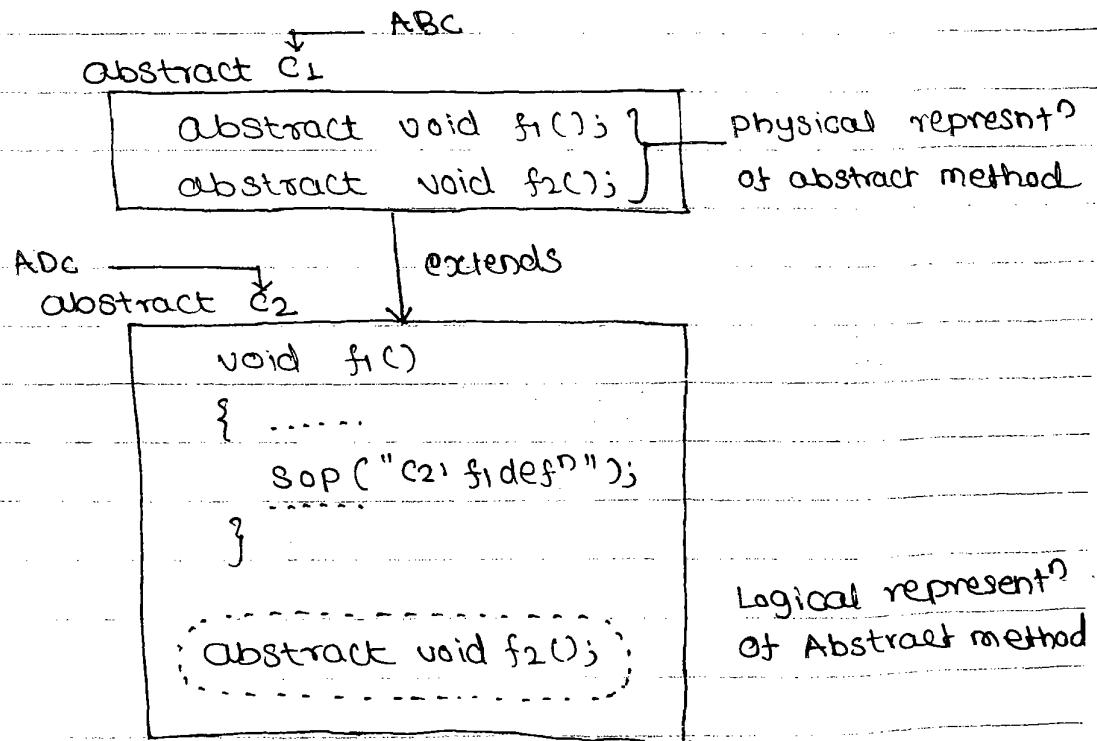
* Abstract Base Class & Abstract Derived class :-

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 inherited from abstract base class.



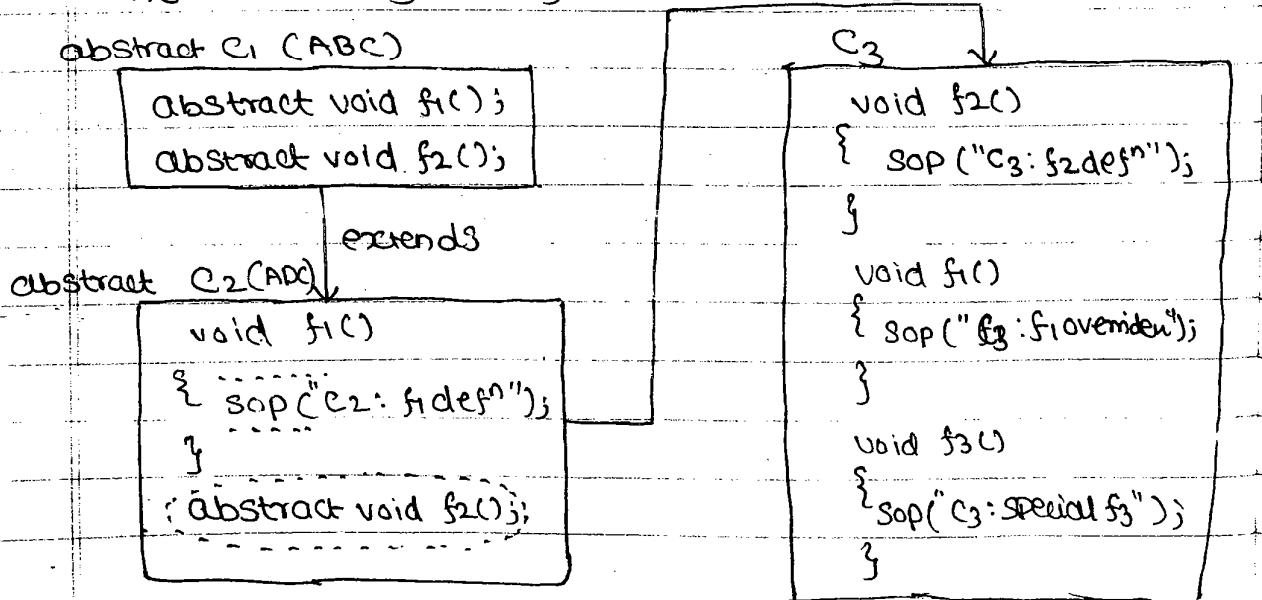
since both these classes are abstract, they are always reusable.

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 definition must be made as abstract by using abstract keyword.

If the derived class defines all the abstract methods which are inherited 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. extends



ANS: (Ans. Given)

abstract class ABC

{

 abstract void f1();

 abstract void f2();

}

abstract class ADC extends ABC

{

 void f1()

{

 SOP (" ADC: f1 def");

}

}

class CDC extends ADC

{

 void f2()

{

 SOP (" CDC: f2 def");

}

 void f1()

{

 SOP (" CDC: f1 overridden");

}

 void f3()

{

 SOP (" CDC: f3 special");

}

}

Class Ab Test

{

psvm (String [] args)

{

SOP(" wrt CDC → PSL");

CDC c0 = new CDC();

c0.f1();

c0.f2();

c0.f3();

SOP(" wrt ADC → DB");

ADC a01 = new ADC(); // Invalid as ADC is

abstract

ADC a01 = new CDC();

a01.f1();

a01.f2();

// a01.f3(); // Invalid as f3() doesn't
exist in ADC

SOP(" wrt ABC → DB");

|| ABC a02 = new ABC(); // Invalid as ABC is abstract

|| ABC a02 = new ADC(); // Invalid as ADC is

also abstract

ABC a02 = new CDC();

a02.f1();

a02.f2();

|| a02.f3(); // Invalid as 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 :-

Q 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 start

}

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 go() {} // Null body method.`

{

}

4

class child extends parent

{

 void go()

{

 Sop(" going to ^{college} each & every day ");

}

}

class NullTest

{

 psum (String args [])

{

 Sop (" wrt Parent => DB ");

(1) Parent Po = new Parent(); or 2 it is abstract

Parent Po = new Child (); // Indirect object

Po.go();

}

}

(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.

15-Aug-13

Q. write a java program which illustrate the above consequence

abstract class OP

{

 void sum()

{

 System.out.println(" I am from sum() without doing any sum");

 System.out.println(" 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;

 System.out.println(" sum of "+a+" and "+b+" is: "+c);

}

g

Class Abtest

{

 public static void main (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(" wif op: DB");
    if (op == new op()); // Invalid op = abstract
        op = new ISum(x,y);
    op.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 # & #

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 overriding defined methods is more
than method overhead cost of overriding
abstract methods.

method overhead	cost required for overriding	cost required for eliminating + construction of existing body	new body
defined method			

method overhead cost of = cost required for constructing new body only.
for overriding abstract method

Method overhead cost reflects on execution time.
Overriding 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
{
    public static void main (String [] args)
    {
        GraphicsEnvironment ge = GraphicsEnvironment.
            getLocalGraphicsEnvironment ();
        String s [] = ge.getAvailableFontFamilyNames ();
        System.out.println ("no. of fonts: " + s.length ());
        for (int i = 0; i < s.length; i++)
        {
            System.out.println (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 getLocalGraph-`
`(③ ② ① -icsenvironment());`

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

(16)

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

Definition :-

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 collection of universal common reusable data members & universal common reusable methods.

Syntax for defining an Interface:-

```
interface <interfacename>
{
    var Decl' Cms 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 userdefined datatype. Wt 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:- public abstract, void f1();

Appended by JVM written by java programmer

Q. Define an interface IL by taking suitable data members & methods.

interface IL

{

 int a=10;

 int b= 20;

 void f1();

 void f2();

}

Compile the above program.

cmd> javac IL.java

ensure IL.java prg will be compiled & IL.class file must be generated.

View the profile of IL

cmd> javap IL

interface IL {

 public static final int a;

```

public static final int b;
public abstract void f();
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 so that overriding of those abstract methods will take less time & results in completely less execution time.

NOTE:- Each & every datamember of interface must be accessed wrt interface name 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.

20
Syntax:-

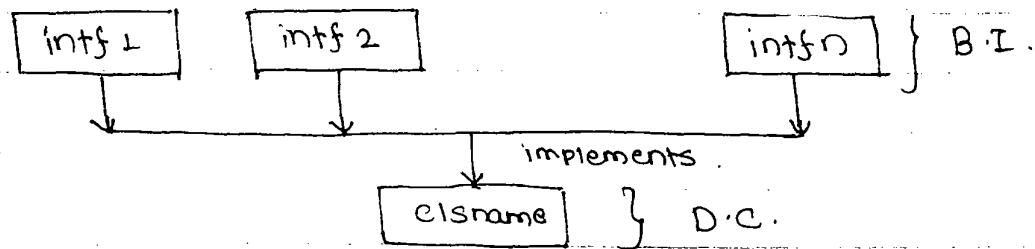
abstract class <clsname> implements <intf1>, ... <intfn>

{

variable declaration;

method decl' | definition;

}



In the above syntax <clsname> represents name of derived class.

<intf1>, <intf2>, ... <intfn> represents name of the base interfaces.

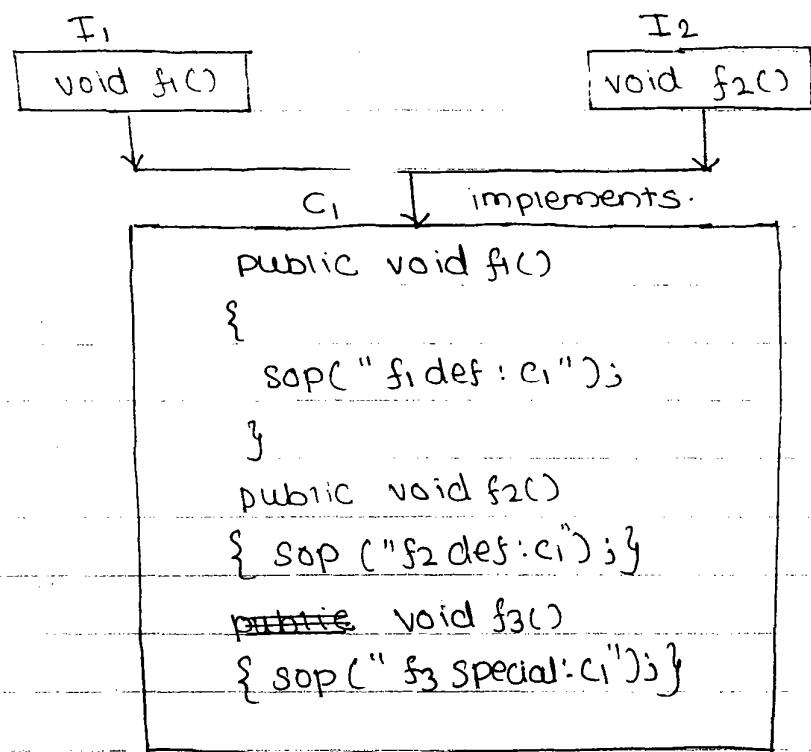
implements is the keyword used for inheriting features of base interfaces into derived class.

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 is inheriting multiple abstract methods from multiple base interfaces then if current derived class is not defining

at least 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()
}
```

(22)

Class C1 implements i1, i2.

```
{  
    public void f1()  
    { sop ("f1() def": c1); }  
  
    public void f2()  
    { sop ("f2() def": c1); }  
  
    void f3()  
    { sop ("f3() special: c1"); }  
}
```

Class InTest

```
{  
    psvm (String args[])  
    {  
        sop ("wrt c1: posl");  
        C1 o1 = new C1();  
        o1.f1();  
        o1.f2();  
        o1.f3();  
        sop ("wrt i1: DB");  
        || i1 o1 = new i1(); invalid co2 i1 is abstract  
        || o1 = new C1(); invalid co2 i1 is abstract  
        o1.f1();  
        o1.f2(); invalid co2 f2() & f3()  
        o1.f3(); does not present in i1  
        sop ("wrt i2: DB");  
        @ i2 o2 = new i2(); invalid co2 i2 is abstract  
        i2 o2 = new C1(); invalid co2 i2 is abstract  
        o2.f2();  
    }  
}
```

- ① Interfaces always contains universal common reusable features.
- ② Interfaces features are always participates in Inheritance.
- ③ Interface definitions 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 directly

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 overriding for business logic development and makes use of dynamic binding principle for development of execution logic

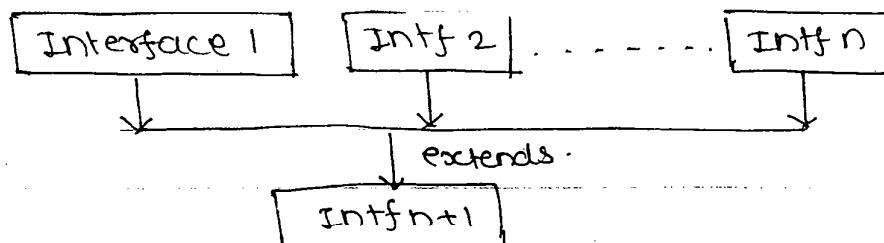
⑩ An object of an interface contains details about those features which are present in the same interface but an object of interface does not contain details about those features which are specially available in other interfaces and in its subclass

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 classes

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 <intfn+1> extends <intf1>,.....<intfn>

{

variable declⁿ cum initialization;
method declⁿ ;

}

Explanation:-

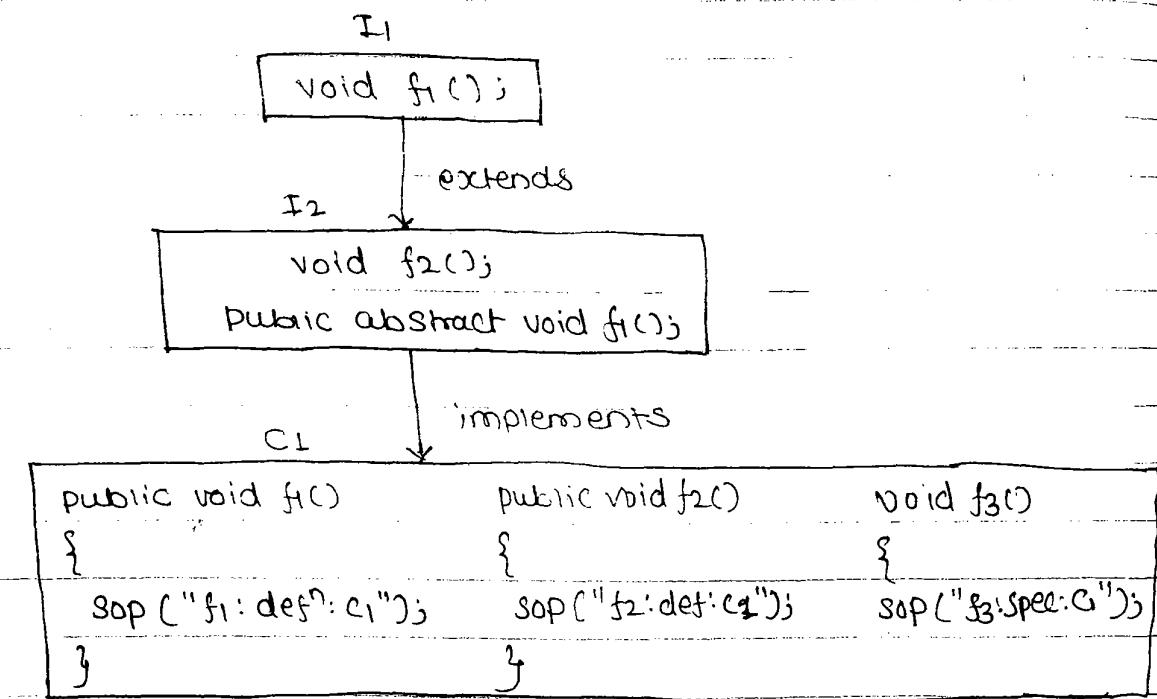
In the above syntax <intf1><intf2>.....<intfn> represent name of base interfaces. <intfn+1> represents name of the derived interface.

Extends is a keyword used for inheriting the features of base interfaces into derived interface.

In java programming one derived class can extend only one base class whereas one derived interface 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

{

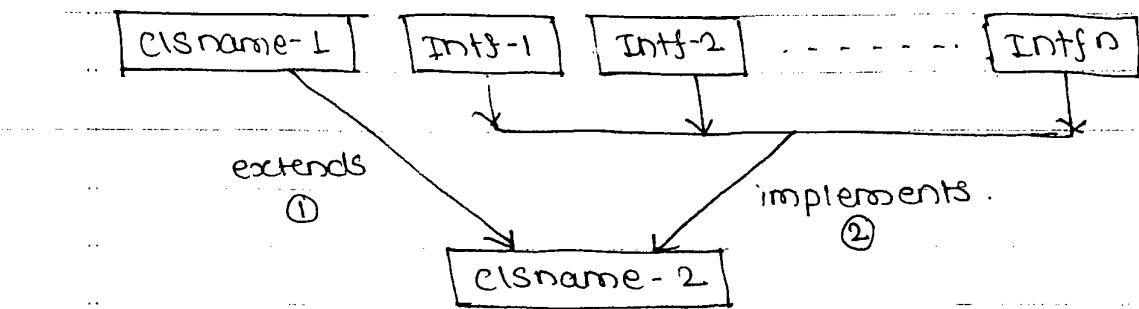
PSVVA (Strong Aggregate)

{

23

Approach 3:- (Parallel Inheritance) .

This approach makes us to understand how to inherit the features of base class & from base interfaces simultaneously to the derived class .



Syntax:-

[abstract] class <clsname-2> extends <clsname-1>
implements <intf-1> <intf-n>

```
{  
    Variable declaration;  
    method defn | Decln;  
}
```

Explanation:-

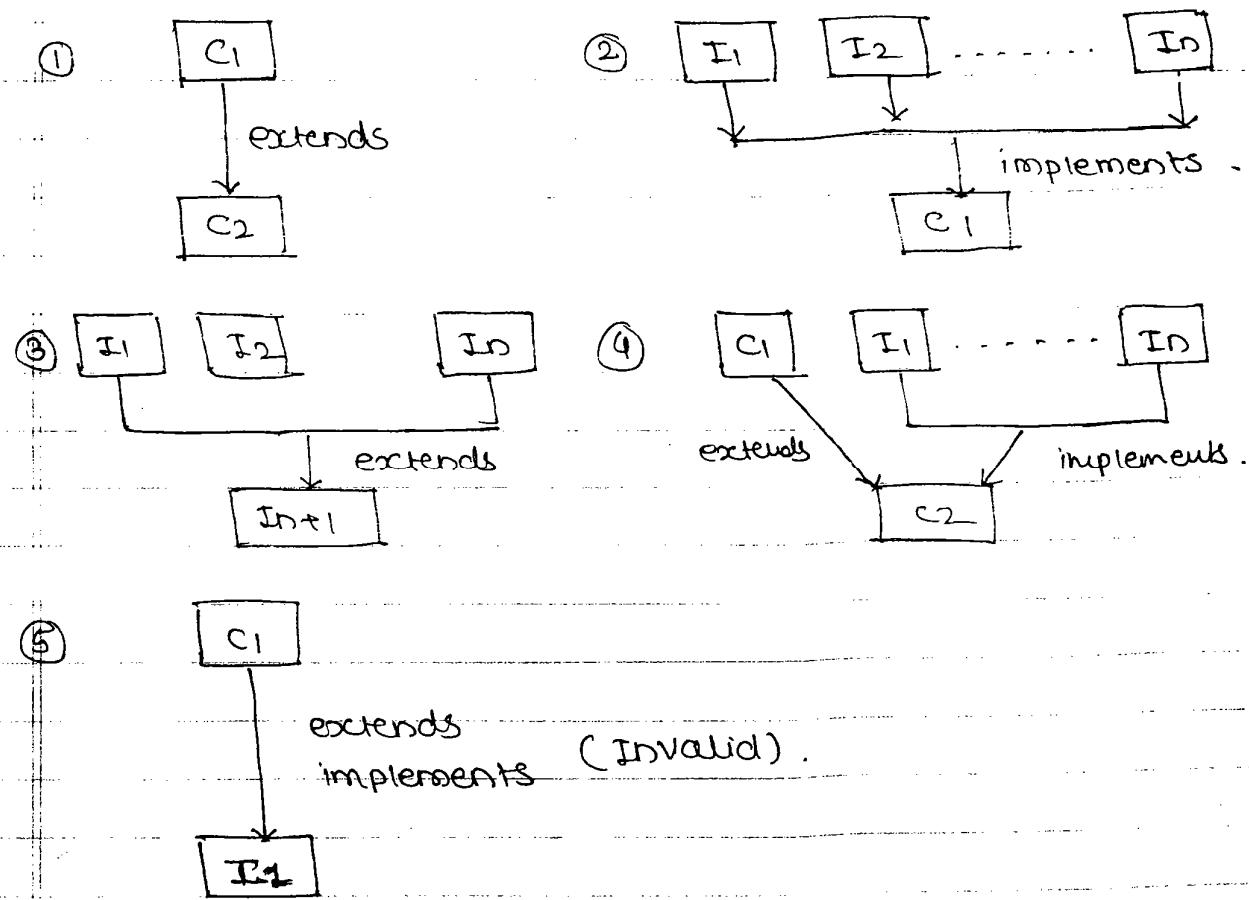
In the above syntax,

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

<Intf-1>,; <Intf-n> represents name of the base interfaces .

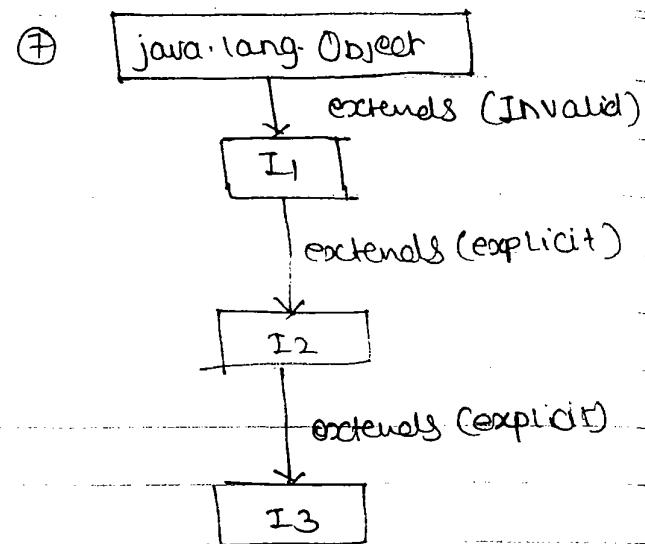
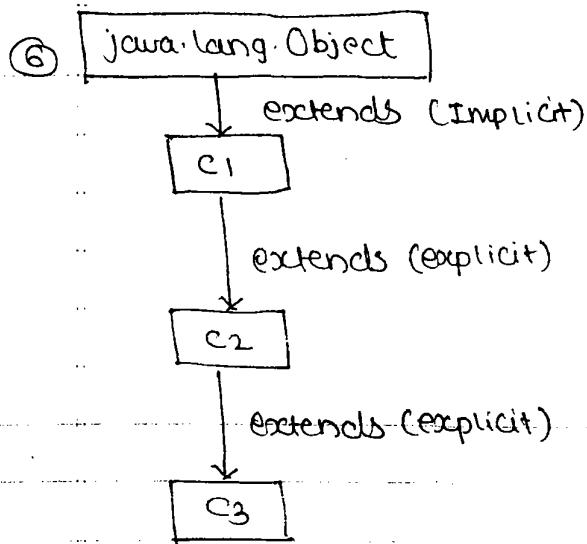
extends & implements are two keywords used for parallelly inheriting features of single base class & from interfaces .

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.



- Because specific features of concrete class & common features of abstract class can not become as universal common features of interfaces.
- Defined methods of both concrete & abstract classes will not become undefined methods.

3a



⑧ class C1 implements I1

{

}

write the profile of C1 (javap C1)

class C1 extends java.lang.Object implements I1

{ C1();

}

20/8/13

(3)

* Packages :-

In general learning of any programming language is nothing but learning about its concepts, syntaxes & library.

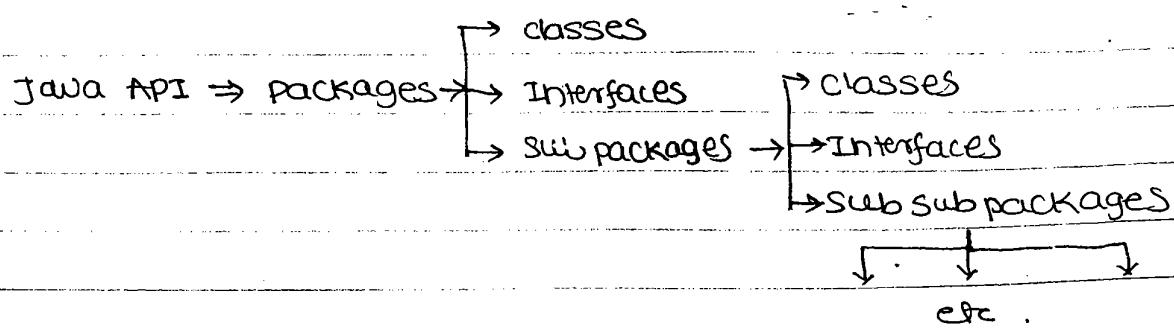
Example:- Learning C language is nothing but learning about its concepts, syntaxes and library. We know that library of C language is a collection of header files. A Header file is a collection of predefined functions.

In C SW 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.

My learning Java is nothing but learning about its concepts, syntaxes & OOPS features & its library/API

An API of Java is a collection of packages

Definition:- A package is a collection of classes, interfaces & subpackages. A subpackage in turn 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 appⁿ by using the concept of packages then we get the following advantages.

- ① Appⁿ development time is less.
- ② Appⁿ memory space (main memory) is less.
- ③ Appⁿ execution time is less.
- ④ Appⁿ 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. 'WORK'.

Q. What is the difference betⁿ Inheritance and package?

ANS: Inheritance concept always makes us to understand how to reuse the features within the programs betⁿ 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 betⁿ 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 | programmers | 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 software 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:- oracle.jdbc.driver.OracleDriver.

Third party package name Class names

* 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 packages)

JSE packages are used for developing clientside applications.

All the JSE packages starts with `java.*`
JEE packages are used for developing serverside applications and each & every JEE pkg starts with `javax.*`

JME packages are used for developing mobile applications & they also starts with `javax.*`

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 package is to provide language functionalities | services | facilities .

Some of the language functionalities are

- ① displaying the data on the console
- ② Accepting the command line arguments (`String`)
- ③ providing the garbage collection facility
- ④ Converting data from numerical strings into numeric values (wrappers classes, etc)

This is the package which is imported by default to each & every java prg. So that this pkc is also known as default package.

② `java.awt.*` :- (Abstract window toolkit)

The purpose of this package is to design the GUI applications (look & feel based appn)

In the real Industry to develop any GUI appⁿ 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 appⁿ we must import `java.awt.*` (To design GUI appⁿ) 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.

⑤ java.applet.* :- (Applet programming)

(22/8/13)
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 intention 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.

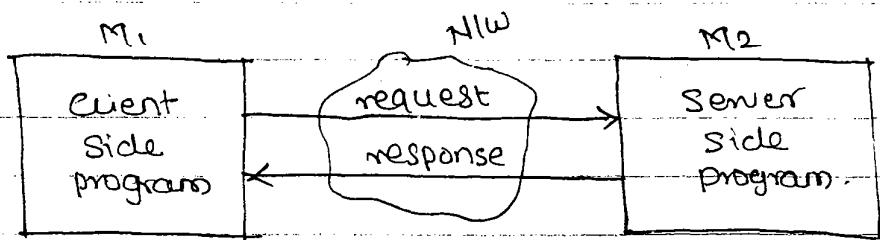
Applet:- An applet is a java program runs in the context of browser | www & whose results are shareable across the globe.

⑥ `java.net.*` :- (Network programming | socket prgning)

The aim of this package is to develop client-server applications .

In client-server app' 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 app's

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

An array is a collective name given to a group of consecutive memory locations which are all referred by Homogeneous/ similar type of values.

Collection framework

Definition as given above.

Arrays concept allows us to organize only homogeneous elements but not heterogeneous elements. Collection fw concept allows us to organize both Homogeneous & Heterogeneous elements.

Arrays concept contains fixed size in nature

Collection fw 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, miscellaneous information like finding the day number in a year, finding the week's number of ^{the} year, etc.

This package is regularly used in report generation modules of every real world application.

⑨ User defined packages :-

User defined packages are those which are developed by Java programmers & 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 & interfaces 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 dir.

Let us consider the following statement.

`java.awt.event.ActionEvent`.

In java programming point of view,

java \Rightarrow ^{upper} package.

awt \Rightarrow ρ sub package.

event \Rightarrow sub sub package.

ActionEvent \Rightarrow 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 L represents name of the upper/outer package & whose specification is mandatory.
- Example: package P1; }
 Package P1·P2; } package start

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 compiletime 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 executable stmt.
- ② Choose an appropriate classname/ Interface name and ensure whose modifiers must be public
- ③ The modifiers 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 modifiers of the method of the class / Interface which is present in the package must be public (This rule is optional in case of Interfaces bcoz Interfaces doesnot contains methods).

⑥ Whatever the class / Interface we are placing in the package, it should be given as a filename with an extension .java.

⑦ At any point of time either we type a class definition in the package or interface definition in the package but not recommended both the definitions should be typed in the same window for placing in the package bcoz we get file recognition problem.

Example 1) Create a package tp and place a class called Test by choosing suitable features.

```
public class Test {  
    public Test() {  
        System.out.println("Test: Default constructor");  
    }  
    public void disp() {  
        System.out.println("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

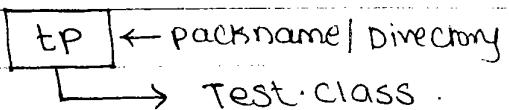
```
javac -d . filename.java
```

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 referred by a symbol dot (.)
- ② filename.java prg will be compiled 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 current created directory

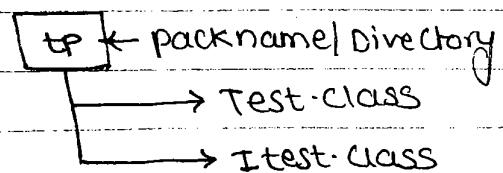
Example ① cmd> javac -d . Test.java.

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



~~26/8/13~~
Q. Write a java program which makes use of Test class of tp package

```

11 PackDemo1.java
import tp.Test;
class PackDemo1 {
    public static void main(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
{
    public void show()
    {
        sop (" Show(): defined swapnil");
    }
}

class PackDemo02
{
    psum (String args[])
    {
        sop (" wrt swapnil: P0E1");
        swapnil so = new Swapnil();
        so.show();
        sop(" wrt Itest: D3");
    }
}
tp.Itest io= new Swapnil(); // Indirect approach.
io.show();
}

```

* Number of approaches for referring classes & interfaces of package

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 referring either all classes & interfaces of a specific package or a specific class or specific interface of a specific package in the current Java program.

An import keyword can be used in various ways

Approach 1:- This approach makes us to understand how to refer ^{all} classes & interfaces of a specific package

Syntax:-

import pack1[· pack2[..... · packn]] · *;

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

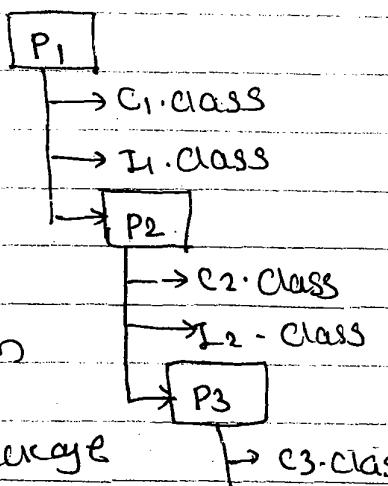
a part of our Java program

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 Stmt 2 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]]].<classname> (Inframe)

- ① import P1.C1;
- ② import P1.P2.C2;
- ③ import P1.P2.P3.I3;

If we use Stmt ① as a part of our java prog 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 referring of class names and interface names before the static data members & even eliminating class names before static methods.

static import statement must be always supplied before static data members & 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 class names & interface names.

Syntax:-

```
import static pack1.[pack2.....[packn]].class
```

Interface name • static datamember name

e.g.:

```
import static java.lang.System.out;
```

三

```
out.println("helloworld");
```

Approach 4:- This approach deals with how to refer the static methods directly without using class name

Syntax:-

`import static pack1.[pack2.....[packn]].classname
- static method name;`

e.g:-

```
import static java.lang.Math.pow;
```

4

```
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 class name | interface name

Syntax:-

import static pack1.[pack2 [.....[packn]].
Classname | Interfacename.*;

e.g:

```
import java.lang.Math.*;
import static java.lang.System.out;
Class MathDemo
{
    psum (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]]] · clasename / intf name

example:-

① P1.C1 o1 = new P1.C1();

② class Test extends P1.P2.C2
{
 ...
 ...
 ...
}

fully
qualified
names

③ class Test2 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 referring the classes & interfaces of a specific package .

* Access Specifiers :-

Access modifiers are those which are applied before the class def's & interface def's 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 default treated as 'default' access specifier.

The purpose of Access Specifiers is how to Access the features within & across the packages . bet' Class to class, interfaces to Interface & interfaces to class . (or)

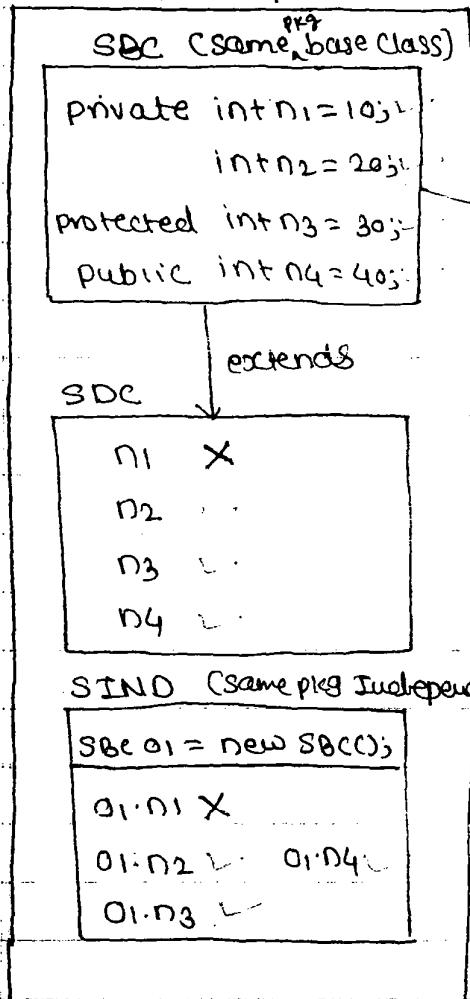
Access specifiers always makes us to understand where to access the features & where not to access the features (or) Access specifiers always provides features controlling accessing mechanism

Rules for Access specifiers:-

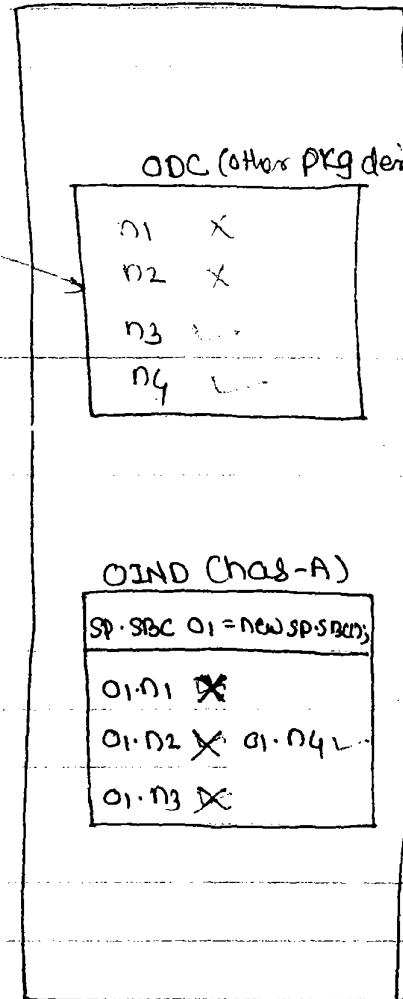
following diagram gives Rules for Access Specifiers .

(52)

SP (Same pkg)



OP (Other pkg)



Access Specifier protection Matrix:-

category car. of classes of ASP.	same package BC	same package DC	same package INDC	different package DC	different package INDC
private	✓	X	X	X	✗
Default	✓	✓	✓	X	✗
protected	✓	✓	✓	✓	✗
public	✓	✓	✓	✓	✓

✓: Accessible / Visible

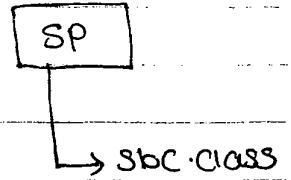
✗: Not accessible / Invisible.

27/8/13

(S2)

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()
    {
        System.out.println("val of n1=" + n1);
        System.out.println("val of n2=" + n2);
        System.out.println("val of n3=" + n3);
        System.out.println("val of n4=" + n4);
    }
}
```



Cmd> javac -d . Sbc.java.

```
package sp;
public class Sdc extends Sbc
```

is-a relationship
within the package

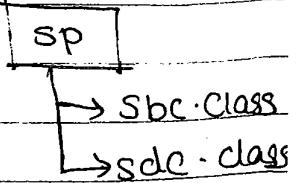
```
public Sdc
```

```
{ // System.out.println("val of n1=" + n1) not accessible
```

```
System.out.println("val of n2=" + n2);
```

```
System.out.println("val of n3=" + n3);
```

```
System.out.println("val of n4=" + n4);
```



Sind.java

```
package sp;
public class Sind
{
```

sbc s1 = new sbc(); // refer to previous class

public Sind()

{ // object of sub class can't be created, it is not possible.

sop("val of n2 = " + s1.n2);

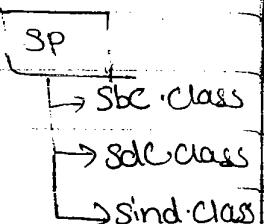
sop("val of n3 = " + s1.n3);

sop("val of n4 = " + s1.n4);

}

}

} ends your class Sind.java



Spdemo.java

This program makes use of sbc, sdc & sind of SP package.



Class Spdemo

{

psum(-)

{ sop("wrt sbc");

sp.sbc s1 = new sp.sbc(); // 10 20 30 40

sop("wrt sdc");

sp.sdc s2 = new sp.sdc(); // 10 20 30 40 20 30 40

sop("wrt sind");

sp.sind s3 = new sp.sind(); // 10 20 30 40 20 30 40

}

}

cmd> javac spdemo.java

and> java spdemo

```

package op;
public class odc extends sbc
{

```

```
    public odc()
```

```
{
    sop ("val of n3 = " + n3);
    sop ("val of n4 = " + n4);
}
```

```
cmd> javac -d . odc.java
```

[OP]

→ odc.class

? Child Class

```
package op;
```

```
public class oind
```

```
{
```

sp.sbc s1 = new sp.sbc(); n 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 n4")~~

sop ("val of n4 = " + s1.n4);

```
}
```

[OP]

→ odc.class

→ oind.class

```
cmd> javac -d . oind.java
```

11. Operator overloading

This feature makes use of overloaded operator.

```
import op::odc;
import op::oind;
class opdemo
{
```

```
    opsum (-)
```

```
{
```

```
    odc o1 = new odc(); // 16 22 32 42
```

```
    oop (" wrt oind");
```

```
    oind o2 = new oind(); // 16 22 32 42 46
```

```
}
```

```
}
```

```
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, protected, public